FEATURE-DRIVEN QUESTION ANSWERING WITH NATURAL LANGUAGE ALIGNMENT

by

Xuchen Yao

A dissertation submitted to Johns Hopkins University in conformity
with the requirements for the degree of Doctor of Philosophy

Baltimore, Maryland
July, 2014

To My Mother's Father

Abstract

Question Answering (QA) is the task of automatically generating answers to natural language questions from humans, serving as one of the primary research areas in natural language human-computer interaction. This dissertation focuses on English fact-seeking (factoid) QA, for instance: *when was Johns Hopkins founded?*.[1]

The key challenge in QA is the generation and recognition of indicative signals for answer patterns. In this dissertation I propose the idea of *feature-driven* QA, a machine learning framework that automatically produces rich features from linguistic annotations of answer fragments and encodes them in compact log-linear models. These features are further enhanced by tightly coupling the question and answer snippets via monolingual alignment. In this work monolingual alignment helps question answering in two aspects: aligning semantically similar words in QA sentence pairs (with the ability to recognize paraphrases and entailment) and aligning natural language words with knowledge base relations (via web-scale data mining). With the help of modern search engines, database and machine learning tools, the proposed method is able to efficiently search through billions of facts in the web space and optimize from millions of linguistic signals in the feature space.

QA is often modeled as a pipeline of the form:

question (input) $\rightarrow$ information retrieval ("search") $\rightarrow$ answer extraction (from either text or knowledge base) $\rightarrow$ answer (output).

This dissertation demonstrates the feature-driven approach applied through-

[1]January 22, 1876

out the QA pipeline: the search front end with structured information retrieval, the answer extraction back end from both unstructured data source (free text) and structured data source (knowledge base). Error propagation in natural language processing (NLP) pipelines is contained and minimized. The final system achieves state-of-the-art performance in several NLP tasks, including answer sentence ranking and answer extraction on one QA dataset, monolingual alignment on two annotated datasets, and question answering from Freebase with web queries. This dissertation shows the capability of a feature-driven framework serving as the statistical backbone of modern question answering systems.

Primary Advisor: Benjamin Van Durme

Secondary Advisor: Chris Callison-Burch

Acknowledgments

To my dissertation committee as a whole, Benjamin Van Durme, Chris Callison-Burch, David Yarowsky and Dekang Lin. Thank you for your time and advice.

I am very grateful to the following people:

Benjamin Van Durme, my primary advisor: Ben admitted me to Hopkins and completely changed my life forever. He wrote me thousands of emails during the course of my study, met with me every week, advised me, helped me, encouraged me and never blamed me a single time for my wrongdoing. He was always there whenever I needed help and he gave me a lot of freedom. Ben is a great person with extraordinary leadership, integrity, fairness, and management skills. I have learned more than enough from him. Thank you, Ben.

Chris Callison-Burch, my secondary advisor: Chris is extremely kind and generous with his students. He read the whole dissertation word by word front to back and marked every page with detailed comments. Chris has taught me things beyond research: networking, entrepreneurship, and artistic thinking. Thank you, Chris.

Peter Clark, who was my mentor when I interned at Vulcan (now his group is part of the Allen Institute of Artificial Intelligence). Pete is the one who inspired me to do a dissertation on question answering. His group also funded two and a half years of my PhD study. He is such a gentleman with an encouraging and supportive heart. Thank you, Pete.

Dekang Lin, who was my mentor when I interned at Google Research on their question answering project. Dekang reshaped my mind in problem solving and finding a balanced point between research and industry. He will have a profound impact on how I work in the future, just like what his research has influenced the entire community. Thank you, Dekang.

Jason Eisner, for whose Natural Language Processing class I was the teaching assistant at Hopkins for two years. Jason helped me with a deep understanding of log-linear models, which are the statistical backbone of the entire dissertation. He is a great person, an intellectual giant, and he treats everyone equally. Thank you, Jason.

David Yarowsky, for whose Information Retrieval class I was the teaching assistant at Hopkins. David's focus on research novelty and originality heavily influenced this dissertation. He also set a good example that finishing a PhD (with high quality) in less than four years was possible. Thank you, David.

Professors and researchers who taught me, mentored me or helped me at graduate school: John Wierman, Mark Dredze, Kyle Rawlins, David Chiang, Liang Huang, Adam Lopez, Matt Post, Sanjeev Khudanpur, Paul McNamee, Phil Harrison, Shane Bergsma, Veselin Stoyanov, and others. Thank you.

Colleagues and friends at the Center for Language and Speech Processing and JHU: Adam Teichert, Adithya Renduchintala, Andong Zhan, Ann Irvine, Aric Velbel, Brian Kjersten, Byung Gyu Ahn, Carl Pupa, Cathy Thornton, Chandler May, Chunxi Liu, Courtney Napoles, Da Zheng, Darcey Riley, Debbie Deford, Delip Rao, Ehsan Variani, Feipeng Li, Frank Ferraro, Hainan Xu, Hong Sun, Jason Smith, Jonathan Weese, Juri Ganitkevitch, Katharine Henry, Keisuke Sakaguchi, Matt Gormley, Michael Carlin, Michael Paul, Nanyun Peng, Naomi Saphra, Nicholas Andrews, Olivia Buzek, Omar Zaidan, Pegah Ghahremani, Peter Schulam, Pushpendre Rastogi, Rachel Rudinger, Ruth Scally, Ryan Cotterell, Samuel Thomas, Scott Novotney, Sixiang Chen, Svitlana Volkova, Tim Vieira, Travis Wolfe, Vijayaditya Peddinti, Xiaohui Zhang, and Yiming Wang. Thank you.

My very best Chinese friends at Hopkins: Cao Yuan, Chen Guoguo, Huang Shuai, Sun Ming, and Xu Puyang. All together we went through so much in grad school. Thank you.

Finally, thank you to my family. I wouldn't have been myself without your support.

Contents

List of Tables

List of Figures

1. Introduction

Question Answering (QA) is the task of automatically generating answers to natural language questions from humans. It provides a natural interface (often via text, image or speech) for human computer interaction (HCI), with the goal of satisfyingly answering as many questions as possible. Question answering is one of the few natural language tasks most humans perform daily, among other common tasks such as natural language understanding and generation. There are a wide variety of types of questions asked every day:

- fact-seeking (factoid), questions about general world knowledge. Usually they come with standard answers and can be judged as either correct or incorrect. For instance:
  - How old is the earth? (answered with a single short phrase)
  - What are the planets in the solar system? (answered with a list of phrases)
  - What is a planet? (answered with a definition)
  - Why is water essential to life? (answered with an explanation)
- opinion-seeking, questions about subjective belief. Usually they do not have definite answers but they can be judged as either relevant/acceptable or irrelevant/unacceptable. For instance:
  - What's the most epic photo ever taken? (most popular question on Quora.com, a community-based QA website)

  - What if the chicken didn't cross the road? (hypothetical)
  - Which dress should I pick? (gentlemen, be careful with the answer)
  - How to grow a garden? (with a process answer)

With gold-standard answers, factoid question answering can be quantitatively measured in terms of precision and recall values. Opinion-seeking questions on the other hand have no standard answers and often use the strategy of user voting to rank the whole list of answer candidates. Answers in fact-seeking questions can often be automatically extracted from existing sources, while answers in opinion-seeking questions are mostly usually produced from scratch or aggregated. We acknowledge the fact humans should play the most important role in opinion-seeking questions, especially in the process of generating original and intelligent content. Thus *this dissertation focuses on factoid question answering*, as it is still a challenging problem for machines, and it has a standard and widely accepted evaluation method.

Over the years, factoid QA systems have evolved in various ways:

- from closed-domain to open-domain: early QA systems such as BASEBALL and LUNAR (introduced more in § 2.3.4.1 on page 60) had very limited knowledge about the world: they only knew about baseball games and rocks on the moon; later QA systems are open to almost any questions.

- from text-based to speech-enabled: a QA system can be assisted with a speech interface with which questions can be transcribed by speech recognition software and answers can be pronounced by speech synthesis software. This is a popular setup on mobile devices or in hands-free environment.

- from out-of-context to contextual: questions used to be understood independently without previous context; now QA systems have a limited understanding of state-changing in conversational dialogues.

The current state-of-the-art factoid QA system is represented by IBM Watson (Ferrucci et al., 2010), which defeated two human champion-level contestants in the Jeopardy! game show in 2011. The Jeopardy! quiz show is a long lasting TV show on the air originated back in 1964. Jeopardy-style "questions" are presented in declarative sentences, commonly called "clues", such as: In 1903, with presidential permission, Morris Michtom began marketing *these toys*. The contestants must then identify the missing information in the clue, and propose a question that can be answered by the given clue, such as: what are Teddy Bears?. The Jeopardy-style clue-and-question pairs can be easily converted into conventional question-and-answer pairs.

By defeating the reigning human champion, Watson represents the most prominent success for factoid QA systems so far. Many questions naturally arose after Watson's success:

1. Did Watson solve Artificial Intelligence?
2. Can Watson really think?
3. What is left for QA research anyway after Watson beat humans?

A closer look at the technology behind IBM Watson will be provided in § 2.3.5.1 on page 69, based on some existing published articles of high-level descriptions. My own short answers to these questions are:

1. No, Watson did not solve AI.
2. Watson has a limited ability to reason among evidence, but not to the level of intellectual process in the human brain.
3. Did the existence of Google stop information retrieval research? [1]

[1] The act of answering a question with a question can be referred as maieutics (in a positive sense), quanswer (a coined word, neutral sense), or question dodging (in a negative sense).

The success of Watson marked the progress of QA (or in more general sense, AI, IR and NLP) research. It was a "showstopper" (in a positive sense) in the human's Jeopardy! quiz show, but it will not be a "show stopper" (in a negative sense) for QA research. Watson is a trained decision maker gathering judgements from hundreds of classifiers, each of which targets on one or several specific types of Jeopardy-style questions. For instance, it has an extractor for answering questions about the US presidents, and a pun detector for facts humans normally consider interesting. Watson was a highly optimized system to excel in the Jeopardy! quiz show with classifiers specifically designed for Jeopardy questions. Based on Watson's success at Jeopardy, IBM has secured contracts to adapt Watson for health care and government.

This dissertation instead focuses on more general techniques that can be easily applied towards questions of various domains. I propose the feature-driven architecture that can be established as the core and robust backbone of a general purpose QA system. It is able to model specialized routines for particular types of questions by incorporating corresponding features. Thus I am also interested in how this general purpose architecture compares with the expert system, Watson, on the Jeopardy data. To have a quick peek at the result, a comparison with the Watson performance curve is drawn in Figure 1.1 (technical details are presented in Chapter 3 on page 86 and experiments in Chapter 4.6 on page 180). Watson started with a re-implementation of research papers but ended up with a lower than expected baseline, dated "12/06" in Figure 1.1 (a)). With a major breakthrough utilizing the DeepQA architecture in 12/07 (the "v0.1" system), the Watson team was able to iteratively push the curve up towards human performance (colored dots in the figure).

The QA system described in this dissertation, jacana-qa (blue line in Figure 1.1(b)), roughly overlaps with the v0.1 version of Watson. jacana-qa does not contain any classifiers specifically designed for Jeopardy questions.[2] If the baseline Watson was an authentic reimplementation of previous years' QA research during the TREC era, then the breakthrough to v0.1 would also mark an equivalent breakthrough of jacana-qa over previously published work. I consider jacana-qa a "lightweight Watson" due to its simplicity and high potential to be easily adapted towards various domains, closed or open.

In the era of mobile computing, question answering has been widely applied as a natural extension to search, especially in conjunction with speech technologies on mobile devices. To name a few commercial systems: Siri (Apple), Google Now (Google), Graph Search (Facebook) and Cortana (Microsoft). These systems start with a concentrated focus on the user, thus deliver an experience in the flavor of personalized search. To deal with the awkward moment of failing to answer a question, different strategies are employed. For instance, Siri often prepares humorous response; Google Now and Cortana fall back to text search; Facebook Graph Search prompts and constrains the users with only questions that it is able to answer. The fundamental research question in QA: increasing recall at a high precision, still persists. Thus even though QA has been in existence as a natural language understanding task since the 1960s, research is still very active in this area.

I predict that the focus of current commercial QA services will gradually shift from *user-aware* towards more *world-aware*. Imagine that a user asks his or her phone for directions to a museum or a cinema. The personal assistant on the phone should not only be able to calculate the optimal route, but also answer questions regarding various facts during the museum visit or prior to the movie. IBM Watson has shown that this kind of factoid question answering, at least in the Jeopardy domain, can be successfully accomplished. This dissertation explores and presents one encouraging technique, entitled "feature-driven question answering", that is

[2] As a matter of fact, when I designed the algorithms for jacana-qa, my references were all TREC questions.

free of any painful manual effort, gives promising result, and does not utilize too much heavy machinery.

1.1. Motivation

Question answering is such an amazing application for AI and NLP that people have imagined creating knowledgeable and/or conversational robots in so many fictional novels and movies. But it is still unsolved. This is the most fundamental motivation. Previous years' QA research has been mostly stimulated by the following three high-profile campaigns:

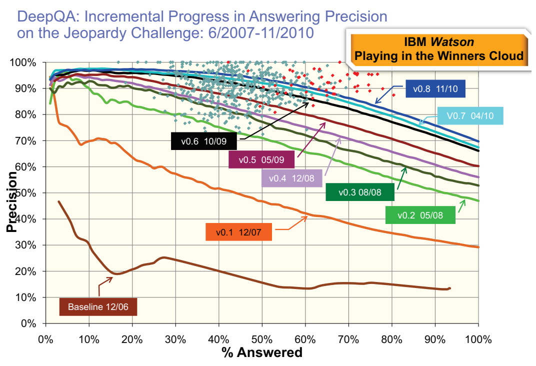
- TREC QA (1999-2007, details in § 2.2.1 on page 31). Almost 10 years of TREC QA evaluation promoted the task of question answering, with a major focus on short, fact-based questions. By current standard, TREC QA used a static and small corpus, and fixed types of questions (mostly wh-questions plus a few how/why questions). One example was: What year was Alaska purchased?. TREC was funded by the ARDA (Advanced Research Development Agency) AQUAINT (Advanced Question Answering for Intelligence) program.[3]

- CLEF Multi-language QA (2003-present, details in § 2.2.2 on page 35). This European QA campaign is similar to TREC-style QA but has a broad reach of major European languages. It also contains bilingual QA, i.e., query in one language and search answers from sources of another language.

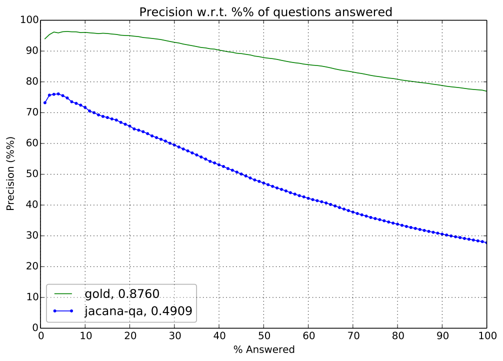
- The Jeopardy! quiz show (1964-1975, 1984-present).[4] This American TV show has always hosted human contestants until the IBM Watson computer

[3]`http://www-nlpir.nist.gov/projects/aquaint/`

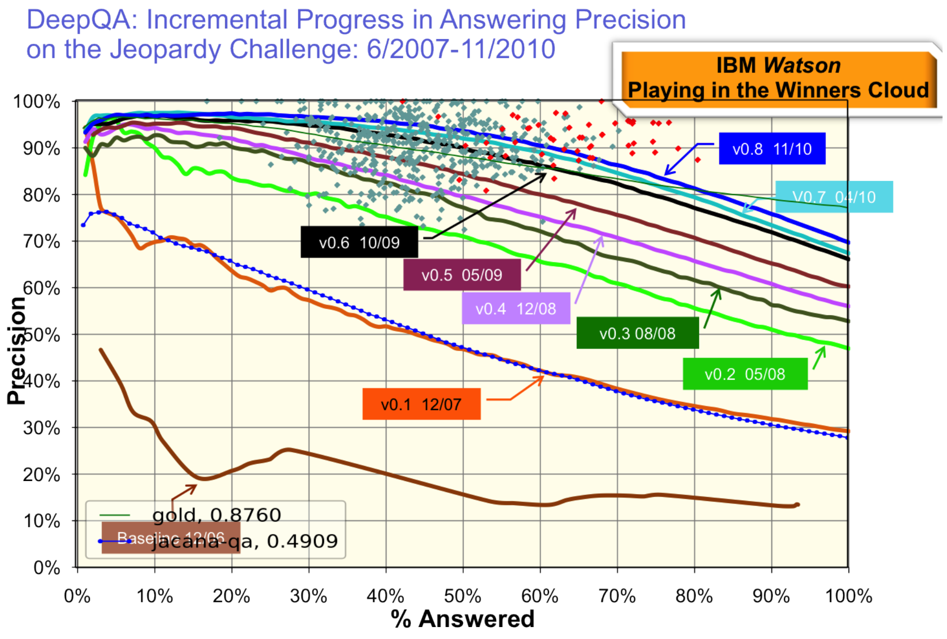
[4]`http://www.jeopardy.com/` all questions and answers can be found at `http://j-archive.com/`



(a) IBM Watson. Curves are from Watson and dots are from human contestants.



(b) jacana-qa (blue line, second from the top), using only web text as the answer source.



(c) a crude overlay of (a) and (b). Note the overlap between blue line of jacana-qa and v0.1 of Watson.

Figure 1.1.: Performance curve comparison (c) on Jeopardy data between IBM Watson (a) and the techniques described in this dissertation (b), with written permission to reuse the Watson curve published in the IBM Journal of Research and Development article (Ferrucci, 2012). Note that this is not a strictly direct comparison: the Watson curve was drawn on about $12,000$ questions from 200 games while the jacana-qa curve was drawn on $18,725$ randomly sampled questions from my own re-creation of the Jeopardy data set.

system made an entry and "beat" the former human winners. The show covers a very wide aspect of world knowledge, sometimes involving inference making, pun and irony detection, beyond fact seeking. One example question under the category of toys is: In 1903, with presidential permission, Morris Michtom began marketing these toys.[5]

These events have provided valuable resources and motivated various techniques (such as question analysis, answer typing and information retrieval) for QA, but not without limitations. For instance, earlier years' QA systems heavily relied on question and answer templates, which were mostly manually constructed and very time consuming. Research in this area prospered for nearly a decade, then cooled off following the closing of TREC QA.

However, technologies have never stopped evolving. The very recent and rapid development of other closely related disciplines, such as AI, knowledge discovery, machine learning, and automatic speech recognition, have created new data and tools for QA research. The feature-driven question answering techniques described in this dissertation are due to these new changes, specifically, motivated by the following factors:

- Large amount of training data. The three QA campaigns have created hundreds of thousands of questions with standard answers, perfect for supervised learning. Amazon Mechanical Turk has also made labeling data in a speedy and large-scale way more affordable than before.

- Machine learning advances. Log-linear models and Markov networks have shown the power in NLP, especially with the maturity of highly optimized tools for large-scale discriminative training.

- Knowledge base creation. Knowledge has been either automatically curated or manually organized into databases for wider coverage and leaner presen-

[5]Answer: What are Teddy Bears?

tation of information. Techniques utilizing KBs should find answers more precisely.

- Increased demand from users. The joint progress of speech technology, mobile device, wireless Internet and search has gradually changed people's habit of typing individual keywords into speaking a whole natural sentence for seeking information. Search engines now receive more queries in natural language questions than ever before.

Thus I revisit the QA challenge by incorporating new machine learning algorithms and tools in current QA research.

1.2. Main Idea: Feature-driven Question Answering

Treating question answering as a machine learning problem, *the most fundamental challenge is capturing the most useful signals of the answer to the question.* Or put it another way, question answering is a patten recognition problem for answers. However, answer patterns do not come out of the box automatically: they need to be produced first. This production process usually requires a lot of linguistic insight, and years of experience. *One central challenge for this dissertation is to design methods for generating these answer patterns, then recognizing them, both in an automatic way.*

We hypothesize that answer signals can be learned from basic linguistic annotations of the QA pairs. Specifically, given a question and related search snippets from text or KB, the annotations (ANNO) used in this dissertation are:

- Q: a simple analysis of the question, such as question word, focus, topic, and main verb (detailed examples are shown in § 2.3.2.1 on page 48);

- if the search snippets are sentences:

  - POS: part-of-speech tags of the words in the snippet;
  - NER: named entity labels in the snippet;
  - DEP: dependency parse of the sentences in the snippet;
- if the search snippets are from a knowledge graph:
  - REL: incoming and outgoing relations of an answer candidate node in the graph, such as people.person.spouse_s;
  - PROP: property of an answer candidate node in the graph, such as *type*:person;
- ALIGN: (soft or hard) alignment between the question and the snippet.

These annotations are then combined as *features* to the pattern recognition problem of QA. It is usually very difficult to decide which features are useful. Thus we first over-generate them, then use machine learning to weigh all the features. We apply this idea to both text (§ 1.2.1) and a knowledge base (§ 1.2.2). Furthermore, we devise our own devices to align questions with the answer source, and show that features based on alignment can significantly improve QA performance, no matter whether it is from hard alignment on text (§ 1.2.3), or soft alignment on KB (§ 1.2.4).

Throughout this section, we illustrate with the following example with two questions and two snippets (one from text and one from KB):

- Q1: Who was President Cleveland's wife?
- Q2: When did President Cleveland marry?
- S1 from text: President Cleveland married Frances Folsom in 1886.
- S2 from Freebase (c.f. Figure 1.2 on page 13(b)):
  - REL: */people/person/spouse_s*(Grover Cleveland, marriage_super_node)

– PROP: marriage_super_node{*Spouse*: Frances Folsom, *From*: 1886, *To*: 1908}

– PROP: Frances Folsom{*Type*: Person, *Gender*: Female, ...}

– PROP: Cleveland{*Type*: Person, *Gender*: Male, ...}

1.2.1. QA on Text

We frame answer extraction with text as a sequence tagging task: a linear-chain Conditional Random Field (Lafferty et al., 2001) is used to tag answer segments in retrieved sentences (Figure 1.2(a)). It inspects each token in a sentence and tag it with either: B-ANS (beginning of answer), I-ANS (inside of answer), or O (outside of answer) – a task very similar to chunking with a CRF. Annotations of each token in the snippet are combined with their local context to generate features. Suppose 0 is the current token position, then the chunking-like features are expanded with unigram, bigram and trigram annotations: ANNO_0, $\text{ANNO}_{-1}|\text{ANNO}_0$, $\text{ANNO}_0|\text{ANNO}_1$, $\text{ANNO}_{-2}|\text{ANNO}_{-1}|\text{ANNO}_0$, $\text{ANNO}_{-1}|\text{ANNO}_0|\text{ANNO}_1$, $\text{ANNO}_0|\text{ANNO}_1|\text{ANNO}_2$, where ANNO is either POS, NER or DEP, or a mixture of the three (examples immediately follow).

Chunking-like features are combined with the question word to capture a general pattern of answer typing. The basic intuition is, for instance, when the question asks for a "`who`", we expect the annotation of POS=NNP (proper noun) or NER=PERSON from the answer. This form of answer typing is reflected through the learned feature weight: the larger the learned weight, the stronger the typing. For instance, compare the following three features and their learned weights for predicting B-ANS on the answer candidate at position 0:

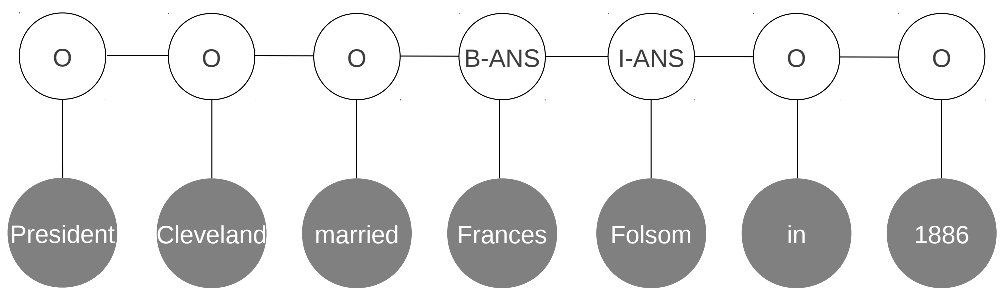
	feature	**token**	**weight**
1	$q=\text{who}|\text{NER}_0=\text{PERSON}|\text{NER}_1=\text{PERSON}$	Frances	0.5
2	$q=\text{when}|\text{POS}_{-1}=\text{IN}|\text{POS}_0=\text{CD}$	1886	0.6
3	$q=\text{when}|\text{NER}_0=\text{PERSON}|\text{NER}_1=\text{PERSON}$	Frances	-0.4

These features are combination between the question word and bigram chunking-like features. A literal translation of the feature in line 1 reads: if the question is a `who` question, and the current token's entity type is a person (NER$_0$=PERSON), followed by another person name (NER$_1$=PERSON), then there is a positive feature weight (0.5) learned. This feature actually fires on the token "`Frances`" in the example of Figure 1.2(a), since "`Frances`" as a first name would be tagged as a person name and the token "`Folsom`" following it is also a person name.

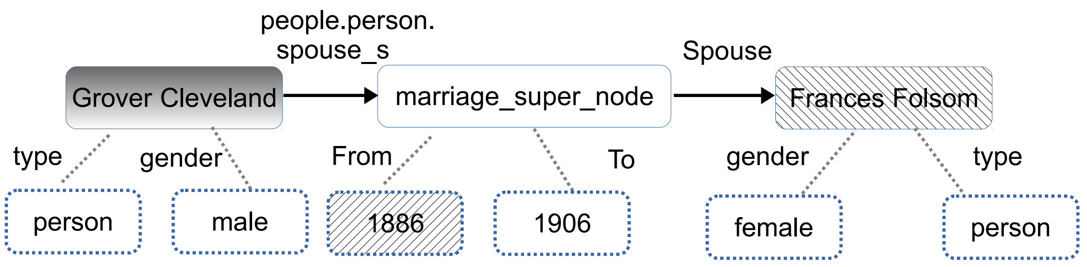
Similarly, in line 2, the CRF learned a good positive weight (0.6) for a feature extracted on the current token, describing the case that there is a `when` question (q=when), and the current token's POS tag is a cardinal number (POS$_0$=CD), preceded by a preposition (POS$_{-1}$=IN). This feature would fire on the token "`1886`" from the string "`in 1886`" if the question was a `when` question, such as "`when did President Cleveland get married?`".

On the contrary in line 3, when the question asks for a time (`q=when`) but the token and its context is about a person (`NER`$_0$`=PERSON|NER`$_1$`=PERSON`), then a negative feature weight is learned. For instance, in Figure 1.2(a), if the question was the same `when` question introduced above, then the feature would fire on the token "`Frances`", discouraging it to be an answer. Negative feature weight can be intuitively understood as discouraging the ANNO pattern from being classified the right answer type for the question.

The automatically generated features can have millions of distinct types given a large training corpus. We applied L1 regularization during training to reduce the feature space. Still, the final model could use thousands of both positive and negative features. A lot of these features are fairly intuitive. Manually writing rules for these scale of model training immediately becomes painfully impossible in this task. But we instead let the automatically generated and optimized features speak for the answer patterns. That is the basic idea of *feature-driven question answering*.



(a) An example of linear-chain CRF for answer sequence tagging in response to the question "`Who was President Cleveland's wife?`". The answer tokens "`Frances Folsom`" are labeled as B-ANS/I-ANS (beginning/inside of answer) and other tokens are labeled as O (outside of answer).



(b) An excerpt of related Freebase graph nodes about Grover Cleveland for answering the questions "`Who was President Cleveland's wife?`" and "When did President Cleveland marry?". Hatched nodes are answers. Solid arrows: relations; solid nodes: entities; dotted lines: properties; dotted nodes: values.

Figure 1.2.: Question answering from either text (a) or knowledge base (b).

1.2.2. QA on KB

The same feature-driven idea can be applied on a knowledge base for extracting answer nodes. Question answering now is treated as a binary classification problem on whether each node in the knowledge graph is the answer, using logistic regression. For each node, we still generate features based on the question and the relation/property of that node, for instance:

	feature	**node**	**weight**
1	q=who|focus=wife|TYPE=person|REL=Spouse	Frances Folsom	0.5
2	q=when|PROP=From	1886	0.6
3	q=when|TYPE=person	Frances Folsom	-0.4

Line 1 reads: if the question asks for the name of a wife (q=who|focus=wife), and the node is a person node (TYPE=person) connected with the spouse relation (REL=Spouse), then there is a high chance that this node (Frances Folsom) is the answer.

The feature at Line 2 would fire on the 1886 node, where we learn a close correlation between the when question word in Q2 and the from property to a node. Normally it is not obvious to the eyes what the From property in the KB means: a date or a location or something else? We just let the logistic regression learner figure it out.

Finally, a negative weight is learnt for the feature q=when|TYPE=person: normally we would not expect a person node to answer a when question. This feature weight makes sense and votes down person nodes.

We present more details in Chapter 5 on page 198 and show that this simple and intuitive technique works on par with more sophisticated semantic parsing approaches. Detailed performance analysis and comparison also make it clear that these two techniques are essentially learning the same answer patterns from the data, except that our feature-driven method is much less complicated.

1.2.3. With Hard Alignment on Text

Combining question features and snippet features provides a loose coupling between the question and the snippet. However it is not precise enough. For instance, we have previously learned that the following feature is a good signal for answers:

q=who|NER_0=PERSON|NER_1=PERSON: 0.5

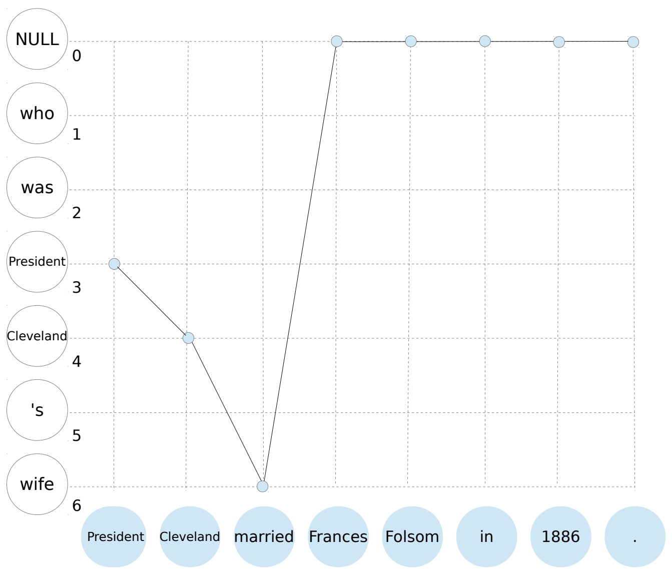
This feature would fire on Frances, Ruth and Esther in the following sentence:

President Cleveland married Frances Folsom and they had two children: Ruth Cleveland and Esther Cleveland,

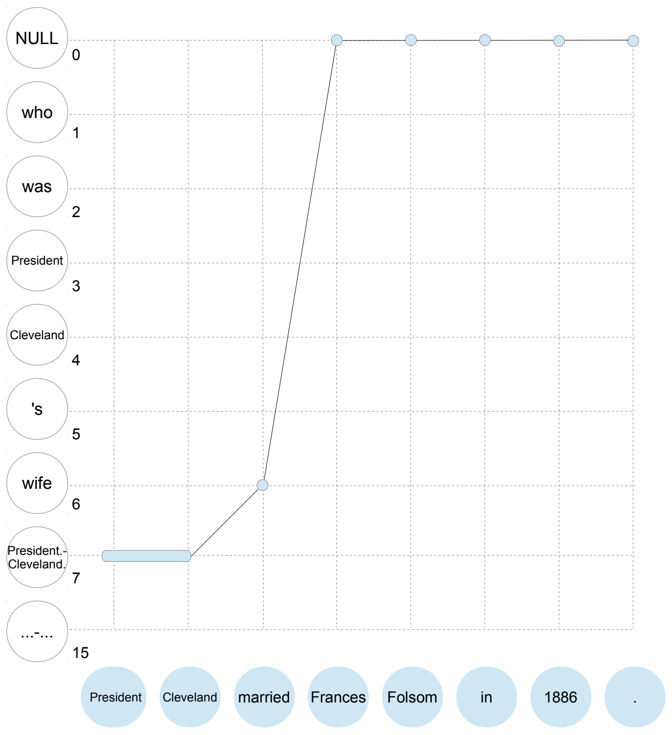
given the question:

Who was President Cleveland's wife?

If we had a mechanism to precisely align President Cleveland↔President Cleveland and married↔wife, then we would know that Frances Folsom has a much higher



(a) Markov token-based alignment. NULL state represents deletion.



(b) Semi-Markov phrase-based alignment. Semi-Markov state aligns phrases on the source (horizontal) side; Phrasal state aligns phrases on the target (vertical) side.

Figure 1.3.: Alignment between QA pairs. The graph shows the desired Viterbi decoding path.

chance to be the answer than Ruth Cleveland and Esther Cleveland, since it is both adjacent to married in word order and depended on married according to the dependency parse.

This mechanism is monolingual alignment. We devise two monolingual aligners, one for token alignment with CRF, the other for phrase alignment with semi-Markov CRF, to connect semantically equivalent words in a pair of sentences. They are able to align lexically similar words, and also make shallow inference through various lists of synonyms, hypernyms, paraphrases etc. Figure 1.3 shows an example.

The two aligners are the first open-source discriminative aligners specifically designed for the task of monolingual alignment. Unlike other aligners based on unsupervised learning or generative models, these aligners are able to optimize over multiple lexical resources, extending the types of alignment beyond word similarity and common co-occurrences.

The major intuitions of using alignment in Question Answering are:

- Aligned tokens (known information w.r.t. the question) are most usually not the answer: since they appear in the question as well.
- Deleted tokens (missing information w.r.t. the question) from the answer sentence might be the answer: but we need to learn the patterns and differentiate them from deleted and irrelevant tokens.

Then these alignment-based features are also fed into the feature-driven framework and weighted on the training data. We show significant improvement with these features later in § 4.6 on page 180.

1.2.4. With Soft Alignment on KB

One challenge for asking questions from a knowledge base is to align knowledge base relations with natural language text. Recall the previous example:

- **Sandra** then was *cast in* **Gravity**, a two actor spotlight film.
- **Sandra Bullock** *plays* an astronaut hurtling through space *in* new blockbuster **Gravity**.
- **Sandra Bullock** *stars/acts/performs in* **Gravity**.
- **Sandra Bullock** conquered her fears to *play the lead in* **Gravity**.

Figure 1.4.: A made-up example of the Freebase annotation of ClueWeb. Blue words are linked back to Freebase entities. Red italic words are what we would map to the Freebase */film/film/starring* relation. The whole corpus annotated 11 billion Freebase entities in 800 million documents.

- Q1: Who was President Cleveland's wife?
- S2 from Freebase:
  - REL: */people/person/spouse_s*(Grover Cleveland, marriage_super_node)

A QA engine should know the mapping between wife and */people/person/spouse_s*. While this is relatively easy since wife can be considered as a hyponym of *spouse*, according to a dictionary or semantic ontology, normally it is difficult to recognize these mappings. Two more examples:

- Q: What to see near Austin?
- Freebase relation: */travel/travel_destination/tourist_attractions*

and,

- Q: Who acted in Gravity?
- Freebase relation: */film/film/starring*

Mapping between, for instance, see and */travel/travel_destination/tourist_attractions*, is a very difficult task. We resort to web-scale data mining for a solution.

The Freebase Annotation of the ClueWeb Corpora (Gabrilovich et al., 2013) annotated about 11 billion Freebase entities in 800 million web documents. If two Freebase entities appear in a close local context (e.g., a sentence) and they are also connected by a direct binary relation according to Freebase, then we hypothesize that the words in the local context are natural language realizations of this relation. Figure 1.4 gives an example: actress Sandra Bullock and movie Gravity are connected by the */film/film/starring* relation. We observe that words like cast, play, and in appear much more than other words such as fears and astronaut and would like to either remove these noise words or rank down them. The task is then:

1. build a co-occurrence matrix $c(w, R)$ between any Freebase relation R and words w in a vocabulary V by trying to maximize the total data probability using EM;
2. compute the conditional probability table $P(w \mid R)$ and $P(R \mid w)$;
3. given a relation R, predict the top n most probable words according to $P(w \mid R)$;
4. given a word w, predict the top n most probable relations according to $P(R \mid w)$.

If we *map* the local contexts for all arguments of the same relation (e.g., */film/-film/starring*(Gravity, Bullock) and */film/film/starring*(The Matrix, Reeves)) to this relation, we can *reduce* the task to simple counting the co-occurrence between the context words and the relation. This provides a MapReduce solution for mining co-occurrence matrix in web scale (800 million documents). Note that the final mapping is a *soft* alignment between words and relations: we do not tell for sure that given a question what correct relation it maps to; but instead we predict several relations that are most probable fit to this question. Chapter 5 on page 198 systematically examines the effectiveness of this soft alignment technique and shows significant improvement to end QA performance.

1.3. Contributions

The substance of this dissertation involves four main aspects:

- *feature-driven question answering*: generating and learning answer patterns directly and automatically from linguistic signals in the data, in contrast to traditional manual template matching (c.f. § 2.3.2.2 on page 49). Further, answer patterns learned directly from data can improve the information retrieval front end for question answering.
- *monolingual alignment*: developing the algorithm, tool and application for the task of monolingual alignment; open souring the first discriminative monolingual aligners with state-of-the-art performance
- *alignment interface to text*: connecting the question with retrieved answer snippets with monolingual alignment and demonstrating that answer signals from monolingual alignment improve question answering performance significantly, over both without and with bilingual alignment.
- *alignment interface to knowledge base*: answering questions directly from knowledge base entries with state-of-the-art performance, via web-scale mining the mapping between knowledge base relations and natural language realizations.

The contributions of this dissertation, both in theory and in practice, to science and research are:

A fast, automatic and discriminative method for quickly kick-starting a statistical question answering system using both unstructured and structured sources. A mature QA system is often a very complex engineering assembly, involving but not limited to the following fields:

- Information Retrieval (IR), from text or Knowledge Base;

- Natural Language Processing (NLP), with various techniques from tagging, chunking, parsing, named entity recognition, coreference resolution, dialogue state tracking, summarization and generation, etc;
- Artificial Intelligence (AI) and Machine Learning (ML), involving reasoning over world knowledge and learning from data.

As a result, building modern QA systems usually needs dozens of engineers and various complex components, let alone painful manual template writing. The resulting system is very hard to describe clearly in research papers and difficult to replicate as well. The feature-driven idea in this dissertation, on the other hand, is easy to grasp, uses modern discriminative machine learning theories, applies well to both text and knowledge base, and helps the performance of both answer extraction in the back end and information retrieval in the front end. I acknowledge the fact that any commercial QA systems would inevitably use heuristics to guarantee very high precision for some important question types. But I think (and this dissertation later argues) that the feature-driven method can serve as the statistical backbone of a large-scale QA system, while still maintaining the flexibility of quickly adapting to other smaller or focused domains.

Specifically, the novel contributions down to the details are:

- the first to cast the problem of *answer extraction as sequence tagging*;
- the first to *automatically* couple the answer extraction back end with the IR front end using shallow structured information retrieval;
- a *novel* method to use information extraction over structured data for question answering.

Systematically developing, demonstrating, and justifying the task of monolingual alignment in the application of question answering. Prior to this dissertation, researchers have mostly used natural language alignment in the following fields:

- Machine translation with bilingual alignment, using mostly the open-source tool GIZA++ (Och and Ney, 2003);
- Monolingual tasks with the "bilingual" aligner GIZA++, because there was no suitable monolingual aligners;
- Recognizing Textual Entailment (RTE) with the MANLI monolingual aligner (MacCartney et al., 2008), which is not open-source and was only used in the NATLOG NATural language LOGic system (MacCartney, 2009).

There is a strong need for developing discriminative monolingual aligners in research, especially when machine learning theories on discriminative training have matured, more and more lexical resources have been created and similarity measures have been proposed. GIZA++ is not able to take advantage of all of them and there is no available monolingual solution off the shelf.

This dissertation describes jacana-align, an open-source discriminatively trained monolingual aligner. We use a different model (CRF and semi-CRF) than MANLI (perceptron) and more lexical resources. We have achieved state-of-the-art performance in alignment accuracy on two common evaluation corpora. Finally, we apply the aligner in the task of question answering and show that alignment-based features give significant boost to QA performance. Thus the task of monolingual alignment in question answering is justified, following the justification in the task of RTE (MacCartney, 2009). To be fair, previous QA research has explored in the direction of fuzzy-matching words or structures in QA pairs, i.e., a soft kind of alignment. But this topic has never been examined to the detail and extent of this dissertation.

Alignment performance greatly relies on external lexical resources. However, putting together multiple resources is an engineering burden. jacana-align ships with various collection of such resources. It is platform-independent and runs on all modern computers. I believe the release of this software would have a profound impact on future researchers' choice of aligners and even their approaches in NLP tasks.

Specifically, the novel contributions down to the details are:

- the first open-source and state-of-the-art monolingual word aligner;
- the first to use semi-Markov CRF models for phrase-based alignment;
- the first to systematically justify monolingual alignment in question answering;
- the first to do web-scale data mining for aligning natural language words with knowledge base relations.

State-of-the-art performance in several NLP tasks. The technologies described in this dissertation achieved state-of-the-art performance in the following tasks by the time of related publication:

- Answer sentence ranking on the TREC QA dataset contributed by Wang et al. (2007), Heilman and Smith (2010);
- Answer extraction on the TREC QA dataset contributed by Yao et al. (2013d);
- Monolingual alignment on the MSR06 (Brockett, 2007) and Edinburgh++ (Cohn et al., 2008, Thadani et al., 2012) datasets;
- Question answering from Freebase on the WebQuestions dataset (Berant et al., 2013).

Easy and fair comparison for future research. All the algorithms described in this dissertation and implementation codes are open-source, even though (and because) I am not producing a QA giant like Watson or search giant like Google. The resulting software is called jacana (/dʒəˈkɑːnə/)[6] consisting of three parts:

- jacana-qa, a question answering engine for TREC-and-Jeopardy-style questions, written in Java;
- jacana-align, a monolingual word aligner for English, written in Java and Scala;
- jacana-freebase, a question answering engine for web-style questions on Freebase, written in Java and Scala.

Along with the code, I have also made almost all datasets and evaluation scripts available to facilitate fair and easy comparison for researchers in the general field of question answering and monolingual alignment. Every single data point and evaluation score reported in this dissertation can be exactly reproduced by a specific snapshot of the above software given the version control number upon submission to corresponding conferences.

1.4. How to Read this Dissertation

The outline for each chapter is:

- Chapter 2 surveys related literature (about 150 publications) covering 50 years of progress in the general field of question answering. Both historical background in QA and IR and the current state of the art are reviewed. The interaction between linguistic analysis and machine learning on the problem of question answering is also explained as a dynamic process.

[6]`https://code.google.com/p/jacana` released with the Apache license with written permission from the Johns Hopkins Technology Transfer Office.

- Chapter 3 kick-starts a statistical QA system and introduces the general idea of generating and learning rich features from data.
- Chapter 4 describes two discriminative models for monolingual alignment and compares them with other monolingual and bilingual aligners on the task of aligning English words, achieving state-of-the-art results. Specifically, section 4.6 justifies the task of monolingual alignment for QA and shows that it improves QA significantly than simply using a bilingual aligner (GIZA++).
- Chapter 5 extends the idea of feature-driven QA from unstructured data source (text) to structured source (Freebase) and shows that this technique is both speedy and state-of-the-art on one question corpus collected from popular web searches.

If you only have 20 minutes: read Chapter 1, which uses one single example to illustrate all important ideas in this dissertation.

If you have 40 minutes: read Chapter 1, every Conclusion section in each Chapter, and the final Chapter 6 on page 236.

If you are a first-timer in question answering: Chapter 2 on page 27 should be a good reference, with most important approaches explained and evaluation methods described.

If you are interested in monolingual alignment: Chapter 4 on page 134 on discriminative models on monolingual alignment has been completely rewritten with models trained with new features and evaluated on additional dataset. It also includes a detailed comparison with bilingual alignment on literature review, evaluation method, and some ambiguous definition.

If you have read all my papers and are looking for something new: thank you! Here is a list of unpublished material in this dissertation:

- Chapter 1 on page 1 (Introduction), Chapter 2 on page 27 (Literature Review), new results and evaluation analysis in Chapter 4 on page 134 (Discriminative Models on Monolingual Alignment) and Chapter 6 on page 236 (Conclusion and Future Directions);
- Section 4.6 on page 180 describes experiments on a new dataset from the Jeopardy! quiz show;
- Chapters 3, 4 and 5 also include additional materials on background, datasets and illustrative figures that were not presented in conference publications due to space limit.
- Appendices include examples of error analysis and comparison on three tasks: monolingual alignment, QA from Jeopardy!, and QA from Freebase.

1.5. Related Publications

This dissertation is based on the following publications:

1. Xuchen Yao, Benjamin Van Durme, Peter Clark and Chris Callison-Burch. *Answer Extraction as Sequence Tagging with Tree Edit Distance.* NAACL. Atlanta, GA, USA. 2013.
2. Xuchen Yao, Benjamin Van Durme and Peter Clark. *Automatic Coupling of Answer Extraction and Information Retrieval.* ACL Short. Sofia, Bulgaria. 2013.
3. Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch and Peter Clark. *A Lightweight and High Performance Monolingual Word Aligner.* ACL Short. Sofia, Bulgaria. 2013.

4. Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch and Peter Clark. *Semi-Markov Phrase-based Monolingual Alignment.* EMNLP. Seattle, WA, USA. 2013.

5. Xuchen Yao and Benjamin Van Durme. *Information Extraction over Structured Data: Question Answering with Freebase.* ACL. Baltimore, MD, USA. 2014.

6. Xuchen Yao, Benjamin Van Durme and Jonathan Berant. *Freebase QA: Information Extraction or Semantic Parsing?.* ACL Workshop on Semantic Parsing. Baltimore, MD, USA. 2014.

Acronyms for conference names are:

- ACL: Annual Meeting of the Association for Computational Linguistics
- NAACL: the annual meeting of the North American Chapter of the Association for Computational Linguistics
- EMNLP: Conference on Empirical Methods in Natural Language Processing

2. 50 Years of Question Answering

2.1. Overview

Due to its bountiful amount of publications and rapid progress, survey papers for question answering are frequently authored. The following is a few of them on different areas of question answering:

- Simmons (1965): Answering English Questions by Computer: A Survey
- Simmons (1970): Natural language question-answering systems: 1969
- Androutsopoulos et al. (1995): Natural Language Interfaces to Databases - An Introduction
- Rahm and Bernstein (2001): A survey of approaches to automatic schema matching
- Voorhees (2001): The TREC question answering track
- Hirschman and Gaizauskas (2001): Natural language question answering: The view from here
- Andrenucci and Sneiders (2005): Automated question answering: Review of the main approaches

- Wang (2006): A Survey of Answer Extraction Techniques in Factoid Question Answering
- Prager (2006): Open-Domain Question-Answering
- Mollá and Vicedo (2007): Question Answering in Restricted Domains: An Overview
- Athenikos and Han (2010): Biomedical Question Answering: A Survey
- Kolomiyets and Moens (2011): A survey on question answering technology from an information retrieval perspective
- Allam and Haggag (2012): The Question Answering Systems: A Survey
- Gupta and Gupta (2012): A Survey of Text Question Answering Techniques

The techniques surveyed above are mainly of three types:

1. **IR QA**: retrieving answer passages with a search engine, then filtering or reranking the retrieved snippets. This was the form of evaluation in early years of TREC QA.
2. **NLP QA**: further extracting answer fragments from retrieved snippets, with different linguistic intuitions and machine learning methods.
3. **KB QA**: finding answers from structured data source (a knowledge base) instead of unstructured text. Normally standard database queries are used in replacement of word-based searches.

IR QA and NLP QA have different focuses. IR QA focuses on *reranking retrieved snippets* while NLP QA on *extracting exact answer fragments*. Both use mostly keyword based search engines as its front end, especially via commercial search engines with the web as the answer source. Note that almost all IR engines are designed serving the need of keyword search, rather than a whole natural language question. Thus research was done calling for *information retrieval specifically for the purpose of question answering.* Back-end NLP methods are applied backward to the IR front end to increase retrieval qualities. We summarize these approaches as IR+NLP QA, or **IR4QA** (Gaizauskas et al., 2004).

NLP QA and KB QA are answer extraction technologies from different sources: text and knowledge base. There are also hybrid approaches combining evidence from both sources. We also outline them as IR+NLP+KB QA, or **hybrid QA**. Two outstanding examples are the MIT START system (Katz et al., 2006) and IBM Watson (Ferrucci et al., 2010).

This dissertation covers all of the above except hybrid QA:

- § 3.2 on page 90 for answer sentence ranking (IR QA);
- § 3.3 on page 98 for answer extraction for TREC questions and § 4.6 on page 180 for answer extraction for Jeopardy! questions (NLP QA);
- § 3.4 on page 110 for structured information retrieval for question answering (IR4QA);
- Chapter 5 on page 198 for question answering with Freebase (KB QA).

Thus I intend to write yet another survey trying to cover an almost complete set of core techniques both important to this field and relevant to this dissertation. But it is not without limitation. I only focus on English factoid QA: providing fact-based answers (in either short phrases or snippets) to English questions.

Besides the above categorization of QA methods, the timeline is also noticeable: QA is a technology that has been constantly evolving over the last 50 years. During this time, theories, technologies and resources all have developed to a great extent. Among them, there are the following prominent evolvement:

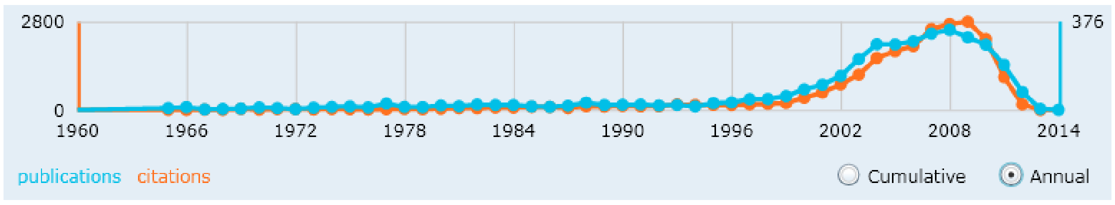
- computing infrastructure, including both static and dynamic storage, central processing unit, and world wide web. Abundant information and knowledge are stored, organized and shared online. Modern search engines are able to retrieve public information in real time.

- machine learning, including various learning and classification methods, generative and discriminative models, data mining and pattern recognition techniques. Along with either manually or automatically created resources, machine learning enables QA systems with the ability to capture the norms of answers in a much larger scale and more systematic and automatic way.
- linguistic theories and tools, including systematic development of syntactic and semantic structures, and mature and robust software to label POS tags, named entities, chunked phrases, dependency trees, semantic roles, logic forms, etc.

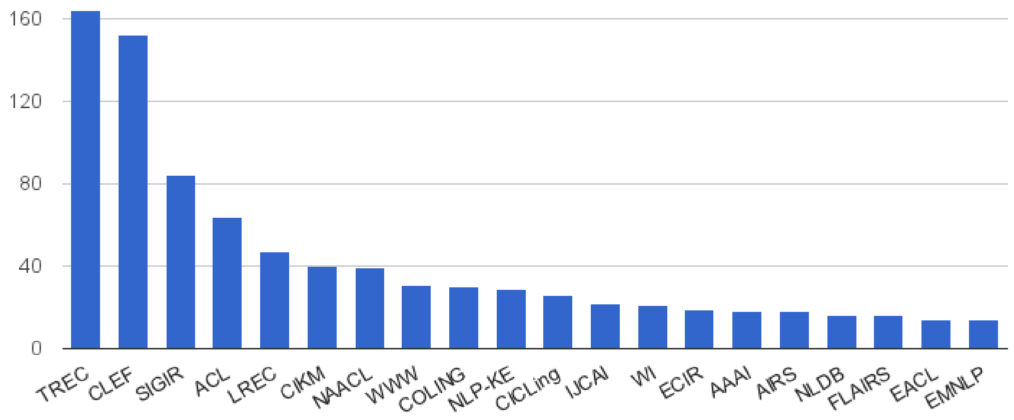
Thus various QA technologies can also be viewed with respect to the types of applied linguistic signal and level of machine learning scale. There is a clear trajectory here: as time goes by, deeper linguistic structures are applied and machine learning algorithms are used in larger scale. Ad-hoc rules are gradually replaced by statistical predictions. In § 2.4 we contrast linguistic features with machine learning scales in existing work, hoping to enlighten future research directions.

Finally, this literature review is based on the following material and my own thoughts:

- significant mentions in the above survey papers;
- first 100 papers by relevance/rank according to Google Scholar and Microsoft Academic Search;
- authors with more than 10 publications in this field according to Microsoft Academic Search;
- selected most recent papers that take distinct approaches other than those that have been explored.



(a) citation and publication per year



(b) top 20 conferences by publication numbers

Figure 2.1.: Publication and most-published conferences in question answering, according to statistics by Microsoft Academic Search

Microsoft Academic Search also gives some overview of the publication distribution over the years and over different conferences. Figure 2.1 gives a snapshot. We see a surge of publications along the timeline of TREC QA (1999-2007) and QA@CLEF (2000-now), both of which will be introduced immediately in the next section.

2.2. Conferences and Evaluation

2.2.1. TREC (Text REtrieval Conference) QA Track

The Text REtrieval Conference started in 1992 as a pure information retrieval challenge: how to scale up the retrieval system then from searching 2MB of texts to 2GB of text. The conference expanded with multiple tracks in various areas (interactive, legal, medical, genomics, video, terabyte, cross-language, etc) and is still very active. Harman and Voorhees (2006) overviewed the first 12 years until

2004 with a lot of background explained.

The TREC Question Answering track started in 1999 and had run for 9 years until the last one in 2007. It started more like an information retrieval task: for every question 5 snippets of 50 bytes and 250 bytes were retrieved and human raters ranked these snippets with whether they answer the question. Evaluation was performed in the form of Mean Reciprocal Rank (MRR), a standard IR evaluation metric. Later TREC QA ran much like an information extraction task: the exact answers were matched to compute accuracy or F_1. As for the information retrieval front end, participants can either use their own tool or use a retrieved document list provided by the organizers. Table 2.1 summarizes each year and Table 2.2 lists the corpora used.

The first 5 TREC runs used independent questions: each question is self-contained and can be asked out of the blue. The last 4 runs grouped questions into question series. A question series focused on a target and consisted of several factoid questions, one to two list questions, and exactly one "other" question. For instance:

target: Britain's Prince Edward marries

- factoid: When did Prince Edward engage to marry?
- factoid: Who did the Prince marry?
- factoid: Where did they honeymoon?
- factoid: Where was Edward in line for the throne at the time of the wedding?
- factoid: What was the Prince's occupation?
- factoid: How many people viewed the wedding on television?
- list: What individuals were at the wedding?

- other: (much like a definition question) "Tell me other interesting things about this target I don't know enough to ask directly".

The introduction of question series brought context into QA: a step towards dialogue-based question answering systems. This has great implications to future mobile personal assistants. In a human-to-computer interaction, users may assume that the subject can be changed as the conversation goes on, just like a human-to-human conversation. However, 10 years after the TREC QA track introduced context-based question answering, modern commercial QA systems still have very limited ability in this task.

I also want to make the comment, which took me a long time to realize, that it is very difficult to compare a QA system at the present time straightly and fairly to the TREC results back then, for two reasons:

1. Early TREC QA used humans to judge whether a 50/250-byte long snippet answers the question. The task was much easier to than strict answer extraction.

2. Later TREC QA allowed participants to search the web thus some systems explored answer redundancy. A static corpus was still provided and as long as the answer candidate (from the web) existed in the static corpus it can be used as the system output. If modern QA systems use the now much better search engine and much richer web content, it would not be a fair comparison with the previous approaches.

Based on the above arguments, if future QA publication were to compare with the previous result, they can only do it in an approximate way. The major concern is the information retrieval front end: both commercial search engines and open-source indexing systems have advanced; thus it is hard to conclude whether any improvement, if any, is from a better IR component, or from a better answer

TREC	questions	tasks	source	evaluation	groups
8 (1999)	2393	factoid	TREC disks 4-5, 1.9GB	50/250 bytes snippets, human ranks top 5 in MRR	20
9 (2000)		factoid, def.	TREC disks 1-5, 3GB		28
2001		factoid, def., list			36
2002		factoid, list	AQUAINT (LDC2002T31), 3GB	exact answer match, in accuracy and F1	34
2003		factoid, def., list			25
2004	351	factoid, list, other, in question series			28
2005	530				30
2006	567				27
2007	515		Blog06 (25G), AQUAINT-2 (2.5G)		21

Table 2.1.: Summary of 9 years of TREC QA on the main task. Overall it contributed about 4000 question answer pairs with about 250 groups participated (and a large collection of publications over the years).

extraction method. In other words, any standard QA dataset for fair comparison should contain three parts: the question, the standard answer, and a list of retrieved snippets for each question.

The MIT109 test collection by Lin and Katz (2006) satisfies the above condition. It contains 109 questions from TREC 2002 and provides a near-exhaustive judgment of relevant documents for each question. In practice researchers (e.g., Ogilvie, 2010) have removed an additional 10 questions without an answer supported by the documents, leaving the final set with 99 questions. Future researchers should be warned that it might be hard to show statistical significance given the small size of this dataset.

Finally, 9 years of TREC QA has left us with more than 4 thousand question answer pairs as training data and publications from about 250 participations. I describe some significant approaches in the next section by impact and relevance to this dissertation.

Corpus	Publisher	URL
TREC disks 1-3 (aka TIPSTER)	LDC93T3A	`http://catalog.ldc.upenn.edu/LDC93T3A`
TREC disks 4-5	NIST	`http://www.nist.gov/srd/nistsd{22,23}.cfm`
AQUAINT	LDC2002T31	`http://catalog.ldc.upenn.edu/LDC2002T31`
AQUAINT-2	LDC2008T25	`http://catalog.ldc.upenn.edu/LDC2008T25`
Blog06	University of Glasgow	`http://ir.dcs.gla.ac.uk/test_collections/blogs06info.html`

Table 2.2.: Corpora used in the TREC QA track. More source and statistics details of TREC disks 1-5 can be found in Table 2 of Voorhees and Harman (2000).

2.2.2. QA@CLEF (Cross Language Evaluation Forum)

The Conference and Labs of the Evaluation Forum (CLEF, previously known as Cross Language Evaluation Forum) started in 2000 with a focus on multilingual and multimode information systems. The Multiple Language Question Answering Track (QA@CLEF) ran from 2003 (Magnini et al., 2003) to 2010.

The first pilot track in 2003 consisted of both monolingual and bilingual tasks: monolingual QA was conducted on Dutch, Italian and Spanish, while bilingual QA contained 200 queries in each language of Italian, Spanish, Dutch, French and German searching over an English corpus from Los Angeles Times 1994. Evaluation was on MRR of 50-byte snippet. QA@CLEF 2004 (Magnini et al., 2005) expanded to 9 query languages and 7 corpus languages. Almost all pair-wise combinations were exploited, so in theory there were 63 combinations and in practice 19 combinations were visited by 18 teams. The questions in different languages were translations of each other. The corpora were local newspaper articles of the same time span. Evaluation was on accuracy of exact answers. The following QA@CELF expanded with more language combinations and teams until 2008.

The first 6 years (2003-2008) of QA@CLEF also produced some sub-tasks including:

- The Answer Validation Exercise (AVE) from 2006 to 2008;

- A time constrained exercise in 2006;
- WiQA in 2006: Evaluating Multi-Lingual Focused Access To Wikipedia;
- WSD QA: An evaluation exercise on Word Sense Disambiguation for Cross-Lingual Question Answering;
- QAST: Question Answering on Speech Transcript;
- GikiCLEF: Cross-language Geographic Information Retrieval from Wikipedia.

Into 2009 and 2010, QA@CLEF investigated whether the previous years's technology based on QA from newswire and Wikipedia can be adapted to the law domain (specifically, European legislation), thus ResPubliQA (Peñas et al., 2010). The finding was that scores were generally higher than in previous QA campaigns but no specific reasons were given.

After 2010, QA@CLEF corresponded to one big concern rising from previous years: the phenomenon of answer redundancy. It is hard to say whether a correct answer is actually from a correct understanding of the text, or just that it highly co-occurs with the question. QA systems also very often rely on the IR front end. Since a search engine almost always returns something, it is hard to produce no answer for a question even if the question does not have an answer from a corpus. The QA4MRE challenge, Question Answering for Machine Reading Evaluation, was designed to test the understanding ability of machine reading systems (Peñas et al., 2011) given a short text. It deliberately ruled out the IR front end. Attention was put on negation and modality in questions as well, two very difficult problems in QA to tackle. QA4MRE is presented in the form of multiple choice questions: given a document, there are ten questions with 5 choices each. Seven languages were used (Arabic, Bulgarian, English, German, Italian, Romanian, Spanish) and the test sets had four topics: AIDS, climate change, music, society and Alzheimer's disease.

Along with QA4MRE, there were the following sub-tasks:

- two pilot tasks on machine reading on biomedical texts (66, 222 Medline abstracts) about Alzheimer's disease using scientific language;
- two pilot tasks on processing modality and negation;
- two challenges on Japanese university entrance exams (in English);
- two BioASQ workshops on large-scale biomedical semantic indexing and question answering;
- four (the first two were run at other conferences) workshops of QALD: multilingual question answering over linked data (DBpedia, Drugbank, etc).

QA@CLEF overall has experimented with various ideas through its pilot tasks and exercises. It has always had a focus on truly improving the understanding of question answering systems. It started with traditional IR-base QA: ranking a 50-byte passage (2003), then measured answer accuracy in the following years. By 2005 the organizers realized that systems reached an upper bound of 60% due to error propagation in the NLP pipeline (Peñas et al., 2013) while more than 80% of questions were answered by at least one participant. Thus the Answer Validation Exercise was introduced in 2006, forcing systems to re-rank their proposed answers. After a brief technology transfer to the European legislation domain, the main task returned to the machine reading challenge. The aim is to remove the influence of the IR front end, and focus on deep understanding of a single document to produce correct answers. Results showed that this is still a challenging task: most systems still used "shallow" technologies up to dependency parsing, without further resorting to logic or inference. The average accuracy was around 25%, while a random baseline is 20%.

A year-by-year overview of the major and pilot QA@CLEF tasks is shown in Table 2.3.

'03	'04	'05	'06	'07	'08	'09	'10	'11	'12	'13	'14
Multiple Language QA Main Task						ResPubliQA		QA4MRE			
			Ans. Validation Exer.			Giki			Biomedical		
			Real Time		WSD QA			Modality& Negation		Entrance Exams	
			WiQA	QA on Speech Trans.				QA over Linked Data			
										BioASQ	

Table 2.3.: Tasks and exercises conducted with QA@CLEF. The structure of table is inspired by Anselmo Peñas's public slides on QA4MRE summary.

2.2.3. Evaluation Methods

One of the most tragic things that can and have happened to research is that the evaluation method is wrong, with or without the author knowing. Thus I decide to write a detailed description of evaluation methods used in this dissertation. The most commonly used evaluation metrics in QA are F_1 and accuracy; in IR they are MAP, MRR, and F_1. They are defined in a straight-forward way but often in practice cause a lot of confusion. In this section I spell out the definition and explain how they are used in research papers.

2.2.3.1. Precision, Recall, Accuracy, F_β for IR/QA

These categories are usually measures for binary classification tasks: the prediction is either correct or wrong. When used in information retrieval, the prediction is whether the document is *relevant* or not, then precision and recall are defined as:

$$Precision = \frac{\mid \{relevant\} \cap \{retrieved\} \mid\}}{\mid \{retrieved\} \mid}$$

$$Recall = \frac{\mid \{relevant\} \cap \{retrieved\} \mid\}}{\mid \{relevant\} \mid}$$

The numerator part is the same: how many documents or snippets are retrieved as relevant. The denominator part is different: precision cares about the percentage of relevance within retrieved while recall cares about the percentage within all that should have retrieved (more like coverage).

When used in QA, the corresponding terms would be roughly that *relevant* means *correctly answered* and *retrieved* means *attempted to answer*.

F_β is defined as:

$$F_\beta = \frac{(1+\beta^2) \cdot precision \cdot recall}{\beta^2 \cdot precision + recall}$$

Usually we take $\beta = 1$ to value precision and beta equally, then:

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

β decides how much we weigh recall with respect to precision. If $\beta = 0.5$, then recall is weighted half as important as precision; if $\beta = 2$, then recall is weighted twice as important.

In most QA tasks, it is general the case that every question has an answer, especially when using the web as a corpus, or in the case of Jeopardy! quiz show. Thus sometimes the term *accuracy* is used. In this situation all questions are relevant:

$$\mid \{relevant\} \mid = \mid \{all\} \mid$$

and accuracy is actually recall:

$$Accuracy = Recall = \frac{\mid \{relevant\} \cap \{retrieved\} \mid\}}{\mid \{relevant\} \mid} = \frac{\mid \{relevant\} \cap \{retrieved\} \mid\}}{\mid \{all\} \mid}$$

Furthermore, some QA systems answer *every* question, then in this case the "retrieved" questions are actually all questions:

$$| \{retrieved\} |=| \{relevant\} |=| \{all\} |$$

and the four terms are equal to each other:

$$Accuracy = F_1 = Precision = Recall = \frac{| \{relevant\} \cap \{retrieved\} |\}}{| \{all\} |}$$

Based on the above two special cases where each question has an answer, also assume that there are two QA systems:

1. S1 answers all questions, so $Accracy = F_1 = Precision = Recall$ for S1.
2. S2 answers some questions but not all of them, so $Accracy = Recall$ for S2.

It is often tricky to compare these two systems, depending on how bad it is to answer a question wrong: S1 tries to answer all of them since there is an answer for every question; S2 tries not to answer a question if it does not have enough confidence, or could not find an answer. S1 often chooses to report accuracy; S2 often chooses to report F_1. Some people value accuracy more since it pushes a QA system to its limit and forces it to answer every question; if it chooses not to, then the final result in terms of accuracy will suffer. Other people value F_1 more since it is less bad to say "I don't know" than giving a wrong answer to the user.

2.2.3.2. MAP, MRR for IR

Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) are standard measures for IR. Given a query, suppose there are n retrieved documents with rank, then the average precision (AveP) is defined as:

$$AveP = \frac{\sum_{k=1}^{n} P(k) \times rel(k)}{| \{relevant\} |}$$

where k is the rank of each document and $rel(k)$ is a binary indicator $\in \{0, 1\}$ about whether document k is relevant or not. $P(k)$ is the precision up to document k. The existence of $rel(k)$ decides whether we will count $P(k)$ into $AveP$ if k is relevant, otherwise not.

Given two rankings, both of which find 5 relevant documents from their 10 retrieved results. Then the ranking that puts the five relevant ones front in its list gets better AveP.

AveP is defined over a single query, if there are Q queries, then MAP is defined as:

$$MAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q}$$

MAP is defined on the entire retrieved list of Q queries. It is an average over ranks. Mean Reciprocal Rank (MRR) on the other hand only cares about the rank of the first relevant document:

$$MRR = \frac{1}{Q} \sum_{q=1}^{Q} \frac{1}{rank_q}$$

where $rank_q$ is the rank of the first relevant document for query q.

In terms of TREC evaluation, the organizers provide a tool to compute MAP and MRR, commonly referred as `trec_eval`.[1]

2.2.3.3. Precision-Recall Curve for IR/QA: Drawn Very Differently

One difference between IR and QA is that, for each query, the returned list of IR can compute a real precision and recall value $(precision, recall) \in ([0, 1], [0, 1])$ at each rank, while a QA system can only answer it right or wrong (I will talk about partial credit later), thus a binary indicator $\in \{0, 1\}$.

Also, for each query in IR, there is usually not an exhaustive list of *all* relevant

[1] `http://trec.nist.gov/trec_eval/`

documents. Assume for a query q an IR model returns with n ordered results with rank $k = 1..n$, m of which are relevant, then we assume the number of relevant documents for q is m. At each rank k, we can compute the precision and recall value $p(k)$ and $r(k)$. Then for result at rank $k + 1$:

- if the result is relevant, then both $p(k + 1)$ and $r(k + 1)$ increase.
- if the result is irrelevant, then $p(k+1)$ decreases and $r(k+1)$ stays the same.

After computing the $p(\cdot)$ and $r(\cdot)$ values for all rank point k, we can plot the precision-recall curve with recall the x axis and precision the y axis. But this curve is only for *one* query, the curve might have a saw-tooth shape. Early TREC evaluations interpolate these points with *11-point interpolated precision* with recall values $\in \{0.0, 0.1, ..., 1.0\}$ (assume precision is 1.0 at recall 0.0). For a total of Q queries, we average all precisions at each recall point and draw *11-point interpolated average precision.* This is the precision-recall curve for IR.

QA systems usually do not have a precision-recall curve for *each* query since the evaluation is binary. But we can draw a precision-recall curve based on system confidence.

A QA system normally outputs a confidence about its answer. A confidence is meaningful if its value is proportional to the chance that the answer is correct. Suppose the confidence score is in the range of [min, max]. Then we run a threshold from max down to min, and compute the precision and recall values for all answers answered with a confidence above this threshold. This is the precision-recall curve for QA. We can also draw the corresponding F_1-threshold curve to find out the threshold value that yields the best F_1.

2.2.3.4. Micro F_1 vs. Macro F_1 vs. Averaged F_1 for QA

For questions with a list of multiple answers, a system might not answer every item in the list perfectly. In this case we can compute a score with a partial credit. For each question, we can compute precision and recall by treating the gold standard answers as the relevant set and the predicted answers at the retrieved set. Then we take an average of precision and recall over all questions. Finally we compute *macro* F_1 by taking the harmonic mean of the average precision and recall. Macro F_1 treats each question equally by taking the harmonic mean of precision and recall values on each questions.

Instead of harmonic mean, we can also compute average F_1: first compute F_1 for each question, then average them.

Another form of averaging over all instances is *micro* F_1, which takes each prediction equally instead of each instance (each instance contains multiple predictions). For example, in alignment each instance is a sentence and each prediction is an alignment; in QA each instance is a question and each prediction is an answer. The way to compute micro F_1 is to keep global counters on each prediction and compute precision and recall only once after going through each prediction in each instance. The following is a mini example of prediction on two instances:

instance	gold standard	prediction	macro	micro
1	a	a	$p = 1, r = 1, F_1 = 1$	$ret = 1, rel = 1, correct = 1$
2	a,b,c	b,d	$p = 1/2, r = 1/3, F_1 = 2/5$	$ret = 2, rel = 3, correct = 1$
avg./sum			$p = 3/4, r = 2/3, F_1 = 7/10$	$ret = 3, rel = 4, correct = 2$

Then the final macro F_1 is:

$$macro\ F_1 = \frac{2pr}{p+r} = \frac{2 \times 3/4 \times 2/3}{3/4 + 2/3} = \frac{12}{17} \approx 70.1\%$$

The final average F_1 is:

$$average\ F_1 = \frac{7}{10} = 70\%$$

The final micro F_1 is:

$$p = \frac{correct}{retrieved} = \frac{2}{3}$$

$$r = \frac{correct}{relevant} = \frac{1}{2}$$

$$micro\ F_1 = \frac{2pr}{p+r} = \frac{2 \times {}^2\!/_3 \times {}^1\!/_2}{{}^2\!/_3 + {}^1\!/_2} = \frac{4}{7} \approx 57.1\%$$

Here the system had prediction on instance 1 entirely correct. Macro F_1 weighs each instance evenly, thus the macro F_1 is larger than micro F_1.

In QA, it makes sense to compute macro F_1 since each question should be weighed equally. In alignment, macro F_1 gives per-sentence performance while micro F_1 gives per-token performance. But it is traditionally computed as macro F_1 (MacCartney et al., 2008).

2.2.3.5. Permutation Test

Parametric significance test assumes prior conditions on the data and should only be used with caution. Paired randomization (or permutation) test, as a version of nonparametric test, has been empirically recommended by Smucker et al. (2007) for IR evaluation. I also found it very easy to understand and implement. It is nonparametric and gives exact answers without parametric assumptions.

Paired permutation test can be applied in deciding which system is "better" in a binary classification problem. The null hypothesis goes as follows. Given two systems A and B and their separate predictions on k instances, if there is no statistical significance between A and B, then it should make no difference if we randomly swapped the prediction of A and B on any instance. The pseudo code goes like this:[2]

[2]From Dan Ventura's page (Thanks to Jason Eisner for pointing me to this): `http://axon.`

1. Obtain k pairs of output from A and B: $\{(a_1, b_1), (a_2, b_2), \ldots (a_k, b_k)\}$; note that a_k and b_k have to be the output on the same instance
   - for instance, a and b here can be precision or partial F_1 on a question (in QA), a query (in IR), or a sentence (in alignment).
2. Compute the absolute difference: $\mu_{diff} = \mid \sum_{i=1}^{k}(a_i - b_i) \mid$
3. Let $n = 0$
4. For each possible permutation of swapping a_i and b_i (perhaps most easily computed by permuting the signs of the k differences):
   a) Compute the new absolute difference: $\mu_{new} = \mid \sum_{i=1}^{k}(a_i - b_i) \mid$
   b) If $\mu_{new} \geq \mu_{diff}$: $n+ = 1$
5. Report $p = {}^{n}\!/_{2^k}$ as the "*p-value*"

In practice, k can be too large and we do not have time to run all 2^k permutations. Often we just run, say, 100 thousand or 1 million times.

Note that the p value from the paired permutation test can only suggest whether to reject the null hypothesis: there is no statistically significant difference between A and B. But it does not tell which one is "better" if significant. The obvious thing to do here is to look at the final averaged measure on all instances and determine which one is better.

The first author of Smucker et al. (2007) also provides a Perl script[3] to compute statistical significance for any measure (MAP, MRR, etc) from the output of `trec_eval` utility. Figure 2.2 gives the 10-line Python script I used for the test.

`cs.byu.edu/Dan/478/assignments/permutation_test.php`

[3]`http://www.mansci.uwaterloo.ca/~msmucker/software/paired-randomization-test-v2.pl`

```
def perm_test(a_f1, b_f1, k = 100000):
    import random
    diff = [a-b for a,b in zip(a_f1, b_f1)]
    mu_diff = abs(sum(diff))
    n = 0
    for i in range(k):
        new_diff = [d if random.random()>0.5 else -d for d in diff]
        mu_new_diff = abs(sum(new_diff))
        if mu_new_diff >= mu_diff: n += 1
    print "p = %f (n = %d)" % (n*1.0/k, n)
```

Figure 2.2.: Python script for permutation test. Input is assumed to be two lists of equal size.

2.3. Significant Approaches

In this section we introduce significant approaches in question answering, most of which were spawned from TREC QA. These approaches are divided into:

- IR QA (§ 2.3.1): a retrieve-and-rerank strategy;
- NLP QA (§ 2.3.2): answer extraction;
- IR4QA (§ 2.3.3): IR for the purpose of QA;
- KB QA (§ 2.3.4): database queries and semantic parsing;
- Hybrid QA (§ 2.3.5): a mixture of above technologies, represented by IBM Watson (§ 2.3.5.1).

2.3.1. IR QA: Document and Passage Retrieval

Document retrieval is the task of retrieving a ranked list of relevant documents in response to a query. It is usually the fist step towards finding an answer. Besides web search engines, four systems have been commonly used in TREC QA: Smart, Prise, Indri and Lucene.

The **Smart** (System for the Mechanical Analysis and Retrieval of Text) information retrieval system (Salton, 1971, Buckley, 1985, Salton and Buckley, 1988) was developed by Gerard Salton at Cornell University from the 1960s to 1990s. It utilizes the vector space model with TF-IDF term weighting to provide relevance ranking of documents to the query. Light preprocessing of corpus is applied, including stopword filtering, suffix stemming, proper noun detection, statistical phrases grouping using high frequency adjacent word pairs, etc. **Prise** was NIST's in-house system (Harman and Candela, 1990) that utilized TF-IDF weighting, with a focus of high speed indexing and searching of gigabytes of data.

Indri and Lucene are two recent open-source and widely used rivals. **Indri** (Strohman et al., 2005) was developed at University of Massachusetts Amherst, after its predecessor **Inquery** (Callan et al., 1992). Indri combines the techniques of probabilistic language models (Ponte and Croft, 1998) and inference network (Turtle and Croft, 1989). It supports a complex query language and structured retrieval with the ability to search through terabytes of data. **Lucene** is developed by Doug Cutting and supported by the Apache Software Foundation for web-scale and single-site search. It employs the boolean model and vector space model for scoring documents. Turtle et al. (2012) compared the performance of Indri and Lucene with TREC 6-8 queries searching TREC disks 4 and 5 and concluded that "using precision at rank 20, Lucene rankings are roughly 30% worse for short queries and roughly 10% worse for long queries". This result is not surprising given that the two techniques Indri is based on, language models (Ponte and Croft, 1998) and inference network (Turtle and Croft, 1989), all claimed to perform significantly better than conventional TF-IDF weighting or boolean models, which are how Lucene scores documents. For a review of more recent information retrieval methods, check out Baeza-Yates et al. (1999), Singhal (2001), Manning et al. (2008), Croft et al. (2010).

Passage retrieval further extracts the location of relevant snippets from the result of document retrieval. However, the line between documents and passages can be blurred by dividing a whole document into multiple pseudo documents: in this case each paragraph, or a fixed number of sentences, or a fixed-length window defines a pseudo document. Tellex et al. (2003) compared various algorithms used for passage retrieval. The finding was that density-based scoring in general performed well (note that this was prior to Indri). Some of the algorithms used in comparison can be applied to both passage and document retrieval. Furthermore, both Tellex et al. (2003) and Clarke and Terra (2003) had consistent findings that passage retrieval algorithms interacted heavily with the more front-end document retrieval.

The difference between document and passage retrieval becomes much more distinctive when viewing in terms of scale: a document retriever is usually designed to search through millions or billions of documents, while a passage retriever can only choose to process a few hundred documents; thus a document retriever usually needs to build the inverted index, while a passage retriever can choose not to. Document retrieval provides coarse pruning of relevant documents. Then passage retrieval can afford more fine-grained search with more sophisticated methods.

In later publications, researchers focused on refined passage retrieval algorithms. The firs step, document retrieval, was usually taken care of by standard search tools, such as Indri, Lucene, or commercial web search engines. When experimenting on the TREC corpus, researchers also had the choice of using a standard list of answer-bearing documents provided by NIST to save the burden of employing a search engine. Lin and Katz (2006) provided a list of exhaustive judgment of relevant documents for 109 TREC questions, which can serve as an oracle for document retrieval.

We next review more heavy-weighted but precise answer extraction methods for question answering.

2.3.2. NLP QA: Answer Extraction

2.3.2.1. Terminology for Question Analysis

We start with some question terminologies used in the rest of this dissertation, defined by Prager (2006). Most of the terms are very intuitive. Thus we omit the exact definition but use an example for illustration. Given the question:

What is the height of Mt. Everest?

The corresponding "part of question" properties are:

- question phrase/word: what
- question focus: height
- question topic: Mt. Everest
- question type*: *factoid* (among *list*, *definition*, *yes-no*, *opinion*, *cause&effect*, etc)
- question category/type*: *numeric*, *numerical value*, etc (depending on the question category hierarchy/topology).
- (expected) answer type*: *number* (depending on the NER output)

*: the last three terms do not have standard usage conformity. In a lot of literature (including this dissertation) question type actually means question category. In some literature question category refers to expected answer types. Readers should differentiate them by the context.

The differentiation between the question focus word and the question topic word can be tricky. For instance, when asking for the height of Mt. Everest, how should a computer determine the focus (height) and the topic (Mt. Everest)? Usually it involves with finding the head word. A handful of rules based on the constituent parse trees should give good enough precision and coverage, given that most questions are not too complicated. I found Silva et al. (2011) a good source for this. Note that hand-written rules are highly dependent on the parser output, thus be careful while following the rules: your parser output might be different! In practice I found that different versions of a parser could also give different output. Thus one should practice extreme caution when practicing handwritten rules. Furthermore, if using a statistically trained parser for question analysis, it is better to use a grammar trained with the QuestionBank (Judge et al., 2006), which contains about 4 thousand parse-annotated questions.

2.3.2.2. Template Matching

Template based approaches use *question templates* to recognize the question types and *answer templates* to extract answers. Rankings of answers are usually summed weighting among various extraction methods, combining confidence scores from the templates, search engine, named entity types etc. Table 2.4 shows some examples from early TREC QA research papers.

Templates can be manually authored (Kupiec, 1993, Hovy et al., 2000, Soubbotin, 2001, Soubbotin and Soubbotin, 2002, Xu et al., 2003, Jijkoun et al., 2004, Peng et al., 2005, Schlaefer et al., 2006) or automatically learned (Ravichandran and Hovy, 2002, Fleischman et al., 2003, Ravichandran et al., 2003, Wu et al., 2005). They can be constructed from surface text, POS tags, named entity labels and syntactic paths. They were often written in the form of regular expressions to increase generality. Some of the patterns were of fairly high precision. Most of these templates were unfortunately not shared publicly.

As an effective early TREC QA approach, template matching was widely used. Lin (2002) made the observation that the types of TREC questions followed the Zipf's Law (Zipf, 1935). Thus a handful of templates can capture a good portion of question and answer patterns. Even though, these templates were highly optimized over both the TREC-style questions, and the specific NLP pipeline used in each participant's system. This was a technique difficult to be transferred to another

question	**question template**
Who was Johnny Mathis' high school track coach?	who be <entity>'s <role>
Who was Lincoln's Secretary of State?	
Who was President of Turkmenistan in 1994?	who be <role> of <entity>
Who is the composer of Eugene Onegin?	
Who is the CEO of General Electric?	

answer snippet	**answer template**
Lou Vasquez, track coach of...and Johnny Mathis	<person>, <role> of <entity>
...Turkmenistan's President Saparmurad Niyazov.	<entity>'s <role> <person>
...in Tchaikovsky's Eugene Onegin...	<person>'s <entity>
Mr. Jack Welch, GE chairman...	<role-title> <person> ... <entity> <role>
...Chairman John Welch said ...GE's	<subject>|<psv object> of related role-verb

(a) Selected question and answer templates used in Hovy et al. (2000)

	BIRTHYEAR		DISCOVERER
1.0	<NAME> (<ANSWER> -)	1.0	when <ANSWER> discovered <NAME>
0.85	<NAME> was born on <ANSWER> ,	1.0	<ANSWER> ' s discovery of <NAME>
0.6	<NAME> was born in <ANSWER>	1.0	<ANSWER> , the discoverer of <NAME>
0.59	<NAME> was born <ANSWER>	1.0	<ANSWER> discovers <NAME> .
0.53	<ANSWER> <NAME> was born	0.95	<NAME> was discovered by <ANSWER>
0.5	- <NAME> (<ANSWER>	0.91	of <ANSWER> ' s <NAME>

(b) Selected answer templates and their precision used in Ravichandran and Hovy (2002)

Table 2.4.: Question and answer templates examples in early TREC QA.

system, another domain or another language.

2.3.2.3. Answer Typing and Question Classification

Answer typing and question classification are twins from two opposite aspects of the QA problem. Their names communicate their focus: answer typing seeks for answer phrases whose types conform with the question's intention; question classification purely concentrates on classifying questions into meaningful categories, without sometimes even looking at the answers.

Answer typing tightens the correspondence between the question and the answer types, an obvious idea commonly used starting from the first TREC QA track (Voorhees, 1999). The answer types are commonly called either *Expected Answer Types* (EATs) or *Lexical Answer Types* (LATs). The most common answer types are named entity labels, and sometimes POS tags (Abney et al., 2000, Ittycheriah et al., 2001a,b, Kwok et al., 2001, Mann, 2001, Xu et al., 2002, Prager et al., 2006, Schlobach et al., 2007). Works later derived answer types from an ontology, such as WordNet (Pasca and Harabagiu, 2001, Prager et al., 2001, Schlobach et al., 2007).

Question classification developed into an individual subtask as the release of a dataset from Li and Roth (2002), which categorized TREC questions into 6 coarse and 50 fine classes. Various machine learning and feature engineering methods were proposed aiming at high prediction accuracies, including the Winnow algorithm (Li and Roth, 2002, 2006), Support Vector Machine (Metzler and Croft, 2005, Huang et al., 2008, Pan et al., 2008, Silva et al., 2011), log-linear models (Blunsom et al., 2006), and Conditional Random Field (Li et al., 2008). Loni (2011) provided an excellent survey on this topic. A "byproduct" of question classification is a detailed analysis of question structures, especially in finding question topic and focus in terms of syntactic headwords. Silva et al. (2011) analyzed this in detail with linguistic rules and headword extraction algorithms. Due to the share of this public dataset, the problem of question classification was well studied. However, the mapping between question categories and expected answer types was little presented.

There are two major problems with pre-defined question categories and answer types. First, the mapping between expected answer types and real answer types is hard to determine, or even not straightforward. For instance, to answer where questions, the expected answer types should be a location. But oftentimes an NER tagger could mistaken location names with person names, as a lot of places are named after person names (e.g., JFK can either mean a person or an airport).

Pinchak and Lin (2006) gave another example: for the question what are tourist attractions in Reims?, the expected answer types can be a lot of things, such as a park and a statue. The second problem is granularity: most NER taggers gave from 3 different labels (LOCATION, PERSON, ORGANIZATION) up to dozens. For instance, BBN's proprietary NER tagger IdentiFinder (Bikel et al., 1999) recognizes 31 different named entities. Table 2.5 summaries various proposals of question and answer type taxonomies. All of these papers showed success. But none of them dealt with the granularity problem.

To alleviate the first problem (mapping), Mann (2001) mapped between expected answer types with named entity types using mutual information; Yao et al. (2013c) learned the mapping automatically from data. To attack the second problem (granularity), Pinchak and Lin (2006) dropped pre-determined answer types, proposed a probabilistic models for word-based answer type clusters and question context, and later refined the model with discriminative reranking (Pinchak et al., 2009a). Both approaches require large corpora to estimate statistics, which is not a problem given today's computing resource.

Do we use pre-defined answer classes or not? My personal opinion on this question can be summarized as:

- When solving *very* open-domain QA problems (such as the questions queried on Google everyday), do not pre-define answer types.
- When focusing on a relative small domain (such as TREC QA), a carefully handcrafted and fine-grained answer type hierarchy could give very good performance. That is one of the reasons why LCC excelled in several years of TREC QA evaluation.
- Always use statistics to estimate the mapping between expected answer types and real answer types. Do not ever write manual rules, unless you know your NER tagger well enough and do not care about portability.

work	question/answer types	domain
Harabagiu et al. (2000)	27 named entity labels with many-to-many mapping to answer types of 15 top nodes	TREC
Hovy et al. (2000, 2002)	topology of 94 nodes (47 leaf nodes) analyzing 17,384 questions	answers.com
Hermjakob (2001)	122 question targets from above questions	answers.com
Pasca and Harabagiu (2001)	taxonomy connecting 153 WordNet sub-hierarchies	TREC
Li and Roth (2002)	6 coarse classes and 50 fine classes	TREC
Prager et al. (2006)	flat hierarchy with 80 semantic classes	TREC

Table 2.5.: Question categories or expected answer type taxonomies.

2.3.2.4. Web Redundancy

Around 2001, TREC QA started to permit usage of external resources for answer extraction, as long as the proposed answers can be found from the official TREC corpus. Participants started to explore web redundancy for answers. One commonly used example was who shot Abraham Lincoln?. The claim was that this kind of information was so abundant that it provided a good clue about the final answer. Clarke et al. (2001a,b) and Magnini et al. (2002) explored web redundancy for answer validation. The AskMSR system (Brill et al., 2001) on the other hand directly experimented with this idea in their TREC participation. Answer candidates extracted from the web were projected back to the TREC corpus for finding supporting documents. Other early systems using the web as the source include Mulder (Kwok et al., 2001), Nsir (Radev et al., 2005) and Aranea (Lin, 2007).

The large volume of the web has incomparable advantage over other closed corpus (Dumais et al., 2002). Using the web as the answer source has become the norm for open-domain question answering systems. The constantly changing web provides up-to-date information about time-sensitive queries. The information it contains is also expanding every second. However, careful attention has to be paid to using the web as the answer source, which contains a lot of noise and unauthentic information. For instance, a search of "king in the united states" gives various information about Martin Luther King and Burger King. Lin (2002, 2007) discussed various issues in this matter.

2.3.2.5. Tree/Graph Matching

Tree and graph matching methods take a deeper step towards linguistic analysis of QA pairs. The structures of the pair are usually presented as dependency trees or semantic role graphs. Then answer ranking or extraction becomes the task of matching two structures. Several significant methods are introduced below.

Lin and Pantel (2001) computed inference (or paraphrase) rules in the forms of dependency parse paths, such as X is author of Y $\leftrightarrow$ X writes Y. The similarity of paths is computed through computing the mutual information of their co-shared arguments X and Y. There was no keyword based document or passage retrieval. A whole 1GB of news text was parsed and stored in a triple database. For a given question, the most similar dependency paths were retrieved and their corresponding arguments were treated as candidate answers. Lin and Pantel (2001) converted question answering into a dependency path matching problem, with paraphrase paths measured by monolingual distributional similarity. The scale of text files was comparable to the official TREC corpora. Evaluation results were mixed: inference rules headed by verbs were in general better than those headed by nouns. But question answering cannot be satisfyingly accomplished with *only* inference rules.

The method of Lin and Pantel (2001) was essentially strict matching relation paths: recall cannot be great in their case. Various methods have been proposed to do fuzzy matching. Cui et al. (2005) applied alignment between dependency parses in passage ranking. For the dependency parse of a given question, all possible alignments to dependency parses in each retrieved passage were summed over to compute a score for the best possible passage that answers the question. The IBM translation model 1 (Brown et al., 1993) implementation in GIZA++ (Och and Ney, 2003) was used to estimate the parameters. Evaluation showed that the fuzzy matching method outperformed density-based passage retrieval methods, some of which were examined in Tellex et al. (2003).

Shen and Klakow (2006) computed the correlation between dependency paths with Dynamic Time Warping (Rabiner et al., 1978). Wang et al. (2007) hypothesized that both the question and the answer sentence were generated by a Quasi-Synchronous Grammar (Smith and Eisner, 2006). Other works include aligning two parse trees with Tree Edit Distance models and kernel methods (Heilman and Smith, 2010, Wang and Manning, 2010, Yao et al., 2013d, Severyn and Moschitti, 2013). Finally, Kaisser (2012) connected two dependency parses of the question and the sentence with their common phrases and drew answer patterns based on the path from the question word to the answer.

All the previous work was based on syntactic parse trees. A syntax tree does not bear too much information regarding named entity types, paraphrases and semantic roles. These extra annotation was mostly attached to the tree by a separate process, or used as features in a machine learning framework. Researchers noticed that semantic role labeling might provide a more unified solution. For instance, the previous paraphrase X is author of Y $\leftrightarrow$ X writes Y would both trigger the Text_creation frame in FrameNet. So there is no need to either use WordNet to recognize the paraphrase in the relations, or monolingual distributional similarity for statistical correlation evidence. Works applying either FrameNet or PropBank style semantic role labeling include that of (Narayanan and Harabagiu, 2004, Sun et al., 2005, Shen and Lapata, 2007). Experimental results varied on the findings. But a general concern with using semantic roles is that the coverage provided by existing tools is very limited, thus recall was not great (more explained in § 2.4.1

on page 80).

2.3.3. IR4QA: Structured Retrieval

Information retrieval and question answering have very different objectives: information retrieval seeks the best correlated snippets to what is *known* in a given query while question answering seeks the best correlated snippets to what is *unknown* in a query. For instance, for the question what country is Berlin in, a density based search would retrieve snippets mostly surrounding the words country and Berlin. However, what an information retrieval front end should really look for is a place that is a country instead of the word country itself. Moreover, the preposition in here takes an important role in indicating the relationship between Berlin and country. However, a lot of density based search would simply rate down stopwords. Information retrieval, serving the purpose of finding the most relevant snippets to country and Berlin, simply cannot satisfy the information need of the answer extraction back end. Thus there has been a call for information retrieval specifically designed for question answering, i.e., information retrieval *for* question answering, or IR4QA.

The idea of retrieving the answer by its expected answer type was firstly proposed as **predicted annotation** by Prager et al. (2000) and later described in more detail in Prager et al. (2006). Expected answer types of questions are manually mapped to about 80 named entity types. Documents are annotated and indexed in a predicted manner with words recognized with these entity types. Then they can be retrieved with the preference of containing words that match the expected answer type when a new question comes in. Harabagiu et al. (2001) had a similar idea without describing details. Predicted annotation was shown most effective for questions that seek definition, person or place names. It also has a few limitations. First, it is labor intensive. New answer types need to be manually defined for each new question type observed. Also, it cannot easily scale up to multiple different annotations as adding more types of annotations quickly become unmanageable. Yao et al. (2013c) attacked these limitations by automatically learning useful annotations from the answer extraction back end, which directly guided what to search from the retrieval front end. Indexing on multiple layers of annotation (POS and NER used in their work) on the same token was made possible with Indri.

We call the line of work from predicted annotation **flat structured retrieval**: texts are analyzed with relatively flat annotations (POS, NER, chunking). This technique serves the purpose of retrieving words of expected answer types. Relations in the question (such as in(Berlin, country)) are approximated by favoring a dense neighborhood of words. Another thread of work, mostly represented by Bilotti et al. (2007), addressed the structured matching problem by going deeper to predicate-argument relations. This technique was published in paper as *structured retrieval*. But we call it **deep structured retrieval** in contrast. Specifically, Bilotti et al. (2007) proposed indexing text with their semantic roles and named entities. Queries then include constraints of semantic roles and named entities for the predicate and its arguments in the question. Improvements in recall of answer-bearing sentences were shown over the flat structured retrieval baseline (called bag-of-words baseline in the paper, which also queried named entities beyond words). Zhao and Callan (2008) later extended this work with approximate matching and smoothing.

Next we discuss these retrieval techniques in several different aspects.

unstructured vs. flat structured vs. deep structured. We reuse the example from Bilotti et al. (2007) to illustrate how these retrieval methods differ. The query is written in the Indri query language:

Question: What year did Wilt Chamberlain score 100 points?

unstructured (possibly with the question word removed):

#combine(what year wilt chamberlain score 100 point)

flat structured:

#combine(#any:DATE wilt chamberlain score 100 point)

deep structured (simplified version):

#combine[TARGET](score #combine[./ARGM-TMP](#any:DATE) #combine[./ARG0] (#combine[PERSON](chamberlain)) #combine[./ARG1](100 point))

An annotated and indexed example snippet:

<ARG0><PERSON>Wilt Chamberlain</PERSON></ARG0> <TARGET>set</TARGET> <ARG1>the single-game scoring record</ARG1> ... by <TARGET>scoring</TARGET> <ARG1>100 points</ARG1> ... on <ARGM-TMP><DATE>March 2, 1962</DATE></ARGM-TMP>.

The #combine operator is the default unweighted operator for combining multiple queries used in Indri. Both the flat and deep structured queries require the DATE annotation in the retrieved snippets. The deep structured query is more specific on the parent-child relation between the predicate score and its arguments.

Both flat and deep structured queries require building extra inverted indices for corresponding annotations, with deep structured retrieval building many more indices due to the massive annotation of predicate-argument structures. In the example given above, both flat and deep structured queries would successfully retrieve the annotated example snippet. How do they compare with each other? Table 4 in Bilotti et al. (2010b) showed that there is no significant difference between retrieving with *only* named entities or *only* semantic roles. But combining both of them would yield a significant improvement over a single one. Table 4.10 in Ogilvie (2010) even showed that the flat structured retrieval with named entities outperformed deep structured retrieval with semantic roles. He explained that this different observation can "be attributed to the differences in queries, the more thorough tuning provided by the grid search, or the inclusion of the length prior".

structured retrieval vs. structured reranking. Structured retrieval retrieve passages with structured constraints directly from an annotated corpus; structured reranking instead relies first on traditional bag-of-words approaches for document retrieval, then refines the ranking of passages by approximately matching their syntactic or semantic structures. From the aspect of computing, structured retrieval has a much bigger *static indexing overhead* during corpus preprocessing while structured reranking has a bigger *dynamic parsing overhead* after documents are retrieved. Structured matching is done *at* query time for structured retrieval and *after* query time for structure reranking.

The document retrieval part is the bottleneck for structured reranking methods: reranking would not work if the retrieved documents do not contain answers. Yao et al. (2013c) showed an example that for the question when was Alaska purchased, none of the top 1000 retrieved snippets based on a plain bag-of-words approach contained the correct answer, simply because the bag-of-words approach would not retrieve any date information. In this case the structured reranking method will never find the answer-bearing sentence.

How do structured retrieval and structured reranking compare in performance? Bilotti et al. (2010b) made a comparison. The experiments showed that the structured reranking methods by Cui et al. (2004) and Moschitti et al. (2007) were statistically indistinguishable from the flat structured retrieval baseline using Indri. Bilotti et al. (2010b) actually also reranked the structurally retrieved passages with multiple features based on (long distance) syntactic and semantic constraints, a typical **learning to rank** task for reordering of search engine results. In their work it is vaguely that:

learning to rank = structured retrieval + structured reranking.

So far we have reviewed some of the QA techniques using information retrieval and natural language processing, both of which seek answers from unstructured text. Next we review approaches seeking answers from encoded databases.

2.3.4. KB QA: Database Queries

In this section we first review rule-based procedures in early years. Then we move on to modern statistical semantic parsing techniques.

2.3.4.1. Early Years: Baseball, Lunar and 15+ More

Early (prior to TREC or Internet) QA systems were constrained by its source of knowledge and computing power: there was no large amount of electronic data to extract answers from; or even if there was, there was not enough computing power and storage. For instance, the fist personal computer, IBM 5100, had up to 64KB RAM and 64KB ROM. An early 3.5 inch "high density" floppy disk holds 1.44MB, while the complete works of Shakespeare take 5MB.[4] Thus early QA systems all worked in a small domain with internally hand-crafted knowledge base.

Baseball (Green et al., 1961) was a pioneer of "many future computer-centered systems [that] will require men to communicate with computers in natural language". It read simple questions about baseball games from punched cards and printed lookup answers from its internal dictionary representation of knowledge. It was invented in the 1960s and thus limited linguistic processing was expected. Baseball avoided complex questions by prohibiting questions with multiple dependent clauses, or logical connectives (e.g., and, or, not). Still it was able to answer quite some questions with constraints. To give an example of how it

[4]To have an idea of how NLP was done in the 1970s, check out Ken Church's Master thesis at MIT: *On memory limitations in natural language processing* (Church, 1980).

worked. For the question:

Where did the Red Sox play on July 7?

The following shallow syntactic structure was obtained (very much like the modern chunking task):

[Where] did [the Red Sox] play (on [July 7])?

Noun phrases are bracketed in [] and preposition phrases in (). A meaning lookup table converts the actual words into "attribute = value" pairs. The above question can be converted into:

Place	=	?
Team	=	Red Sox
Month	=	July
Day	=	7

Then the missing answer is searched from its internal KB and printed.

Baseball discussed the problem of domain adaptation: "considerable pains were taken to keep the program general". Most of the issues came from the internal domain-specific knowledge representations. Very similarly, 40 years later, the semantic parsing approaches (next section) to question answering also started from very small domains, such as geo location and football. Later on researchers realized that these toy domains were a good start to craft a semantic parser but hardly useful in everyday life. Thus the community very quickly switched to open-domain question answering, utilizing general purpose knowledge bases such as Freebase and DBpedia.

A later system called **Lunar** (Woods, 1977) aimed to enable a lunar geologist to query the chemical analysis data on lunar rock and soil composition as a result of Apollo moon missions. Questions such as What is the average concentration of aluminum in high-alkali rocks? were first parsed with a general purpose grammar (Augmented Transition Network, not previously published), then the parse was mapped with rules to a Meaning Representation Language (Woods, 1978), finally database queries were fired to retrieve the answer. LUNAR had a database with 13,000 entries. Evaluation showed that 78% of test questions were answered correctly. The meaning representation language of LUNAR dealt with quantification (e.g., it had operators for the words every and average in counting) and semantic scope, which looked a lot like the first-order logic commonly used nowadays.

Overall, the LUNAR technology from 1977 bears a resemblance to today's semantic parsing approaches to question answering, except for that: 1. the former used mostly in-house solutions such as parsing with ATN and semantic interpretation with MRL, while the latter used more "popular" and "standard" technologies, such as parsing with CCG/dependence and meaning representation in lambda calculus; 2. the former used majorly rules to learn the mapping between parse and meaning, while the latter used machine learning (i.e., rule-based vs. statistical). The journal article (Woods, 1978) about LUNAR's internal semantic representation was very well-thought and well-written. It discussed some long-standing issues in computational linguistics, including: quantification scope; negation, modification and relative clauses in question interpretation; anaphora resolution; syntactic parsing vs. semantic interpretation; top-down vs. bottom-up parsing; role of pragmatics. It also briefly described another application about flight schedules, answering questions such as What flights go from Boston to Washington?, exactly the task later Zettlemoyer and Collins (2009) tried to solve.

Other database-powered QA systems include LADDER (Language Access to Distributed Data with Error Recovery) (Hendrix et al., 1978), PLANES (Programmed LANguage-based Enquiry System) (Waltz, 1978), the Berkeley Unix Consultant (Wilensky et al., 1988), Janus (Weischedel, 1989, Bobrow et al., 1990). For more information about QA systems in the 1960s, Simmons (1965) served as an excellent source by surveying **15 early QA systems**. These systems were classified as "list-structured data-based, graphic data-based, text-based and inferential". The challenges remained throughout 50 years of question answering: "measuring meaning, dealing with ambiguities, translating into formal languages and searching large tree structures".

2.3.4.2. Statistical Semantic Parsing

Semantic parsing can be naively understood as "parsing into a semantic form". Usually a grammar is adopted in parsing a natural language query and the resulting parse trees are mapped into logic forms. Finally the logic form is converted into database queries to retrieve the answer. Modern statistical semantic parsing approaches are mostly represented by works of Raymond Mooney (various methods), Luke Zettlemoyer (CCG parsing), and Percy Liang (dependency parsing).

One of the earliest approaches is that of Zelle and Mooney (1996). The task was to map natural language queries about US geography into a logic form, for instance:

```
What is the capital of the state with the largest population?
answer(C, (capital(S,C), largest(P, (state(S), population(S,P)))))
```

Both the question and the logic forms were input to a semantic parser for supervised training. Evaluation was on whether the correct answer was retrieved from the geography relational database. Zelle and Mooney (1996) applied a general purpose shift-reduce parser and used the input logic form to guide the parser in outputting the correct logic form. Results showed that when given enough training examples (more than 150), the statistically trained parser was able to achieve better accuracy than a hand-crafted rule-based system. The accompanying dataset, **Geoquery** (or **Geo880**), has also become standard in later works. Thompson and Mooney (2003) applied active learning to semantic lexicons and also tested on 3 other languages (Spanish, Japanese, Turkish) translated from 250 sentences of Geo880. Wong and Mooney (2007) adopted a machine translation approach by learning a synchronous grammar that generates logical forms in lambda calculus. Their Wasp system had improved results on the four languages.

Tang and Mooney (2001) applied multi-strategy learning in their Cocktail system with multiple clause constructors and tested on both the Geoquery domain, and another database called **Jobs640**, which contains 640 sentences from a job postings domain. Ge and Mooney (2005)'s system, Scissor (Semantic Composition that Integrates Syntax and Semantics to get Optimal Representations), parsed sentences into semantically augmented trees that compositionally generated formal meaning representations in the domains of Geoquery and Clang (or **Robocup**), which contains 300 pieces of coaching advice from the log files of the 2003 RoboCup Coach Competition. Kate and Mooney (2006) used string kernels with SVMs to classify the correct meaning representation from queries in the same domains of Geoquery and Robocup. They found the kernel-based approach, Krisp (Kernel-based Robust Interpretation for Semantic Parsing), was particularly robust to noise. Lu et al. (2008) proposed a generative model that did not use any explicit grammars, but constructed domain-specific grammars (on Geoquery and Robocup) based on training examples (questions and logic forms). The model was defined over hybrid trees with both syntactic structures and meaning representations on lexicons. Learning was done via the EM algorithm and the produced parse trees were reranked discriminatively with perceptrons to upweight long range dependencies.

The domains of Geoquery, Jobs and Robocup are small but also highly-compositional. The Combinatory Categorial Grammar (Steedman, 2000) is a natural fit for this nature of compositionality. For instance, with the functional application rule, the sentence Utah borders Idaho can be parsed as:

Utah	:=	NP : utah
Idaho	:=	NP : idaho
borders	:=	$(S\backslash NP)/NP : \lambda x.\lambda y.\text{borders}(y,x)$
Utah borders Idaho	:=	borders(utah, idaho)

Zettlemoyer and Collins (2005) was the first to learn a joint model of probabilistic CCG with logic forms using conditional *log-linear models*, yielding an accurate model with precisions on both Geoquery and Jobs exceeding 96% (recall was 79.29%). Later, Zettlemoyer and Collins (2007) *relaxed the constraints of compositionality* on the **ATIS** (Air Travel Information System) travel-planning domain (Dahl et al., 1994), as it contains spontaneous, unedited spoken language input, such as show me the flights to Boston. They relaxed the functional application rules in CCG and added additional rules for type raising to allow flexible word order, or insertion and deletion of lexical items. The online learning algorithm based on perceptron allowed example-by-example training (as compared to the batch learning in the previous paper). This made large-scale training more efficient: Atis contains 5418 sentences as compared to Geo880 or Jobs640. Results showed the recall was also improved, possibly due to the relaxed rules in CCG. Kwiatkowski et al. (2010), with their Ubl (Unification Base Learning) system, also applied CCG but with higher-order unification to learn a more general grammar that was able to map different natural languages to a wide variety of logical representations. This was a step *towards language and logic form independence*. Note that all of these approaches were fairly accurate (with precisions above 90% on Geo880 and above 85% on atis). However, it came with the price of a fairly verbose lexicon. For instance, the unambiguous word flight had three different meaning representations depending on the context:

flight	:=	N : λx.flight(x)	flight
flight	:=	N/(S\NP) : λfλx.flight(x)$\wedge$f(x)	flight departing Boston
flight	:=	N\N : λfλx.flight(x)$\wedge$f(x)	from Boston flight to New York

Kwiatkowski et al. (2011) attacked this problem with lexical generalization. They used templates for words to fit in systematic variation in word usage and learned a factored compact lexicon. Experiments results showed lower precision but higher recall, due to the *more general lexicon.*

All the previous work so far had focused on small domains, until Cai and

Yates (2013a) addressed this problem by creating a database based on Freebase with 917 questions (thus the **Free917** dataset) from 81 domains. This was a big step towards *domain independence*. The major challenge is textual schema matching: identify natural language words that correspond to Freebase relations. For instance, the words directed, by and produced are good cues for the relation *film_director*. Their approach was to find common words for a certain relation by searching the relation's arguments from hundreds of search engine queries, then select by computing the point-wise mutual information between the words and the relation. In this way, the semantic parser's performance on one domain did not degrade much even when it was trained on another domain. Kwiatkowski et al. (2013) approached cross-domain semantic parsing differently. They first defined a general purpose grammar with a wide word class coverage due to the POS information of the word in Wiktionary. Then questions were parsed into underspecified logic forms based on the general grammar. Finally in the logic forms the compositional structures were collapsed and literals were collapsed or split according to some pre-defined conditions and operations. That was the step for fitting a specific domain ontology based on Freebase.

All of the approaches reviewed so far have used the annotated logic forms as supervision, which are usually expensive, slow, and prone to errors. Clarke et al. (2010) instead just replied on the question and answer pairs for the semantic parsing task. The answer served as a form of *distant supervision* in guiding the production of semantic forms. They used Integer Linear Programming to constrain how text spans mapped to predicates and how they composed with each other. Unlike previous approaches, where logic forms followed the syntactic structures of the questions strictly, in this work syntax was only used to bias semantic composition. Experiment results on Geoquery showed that the approach was competitive to fully supervised methods. Liang et al. (2011) also took the same idea of omitting logic forms as supervision and learned directly from the end-to-end QA pairs. Instead of using CCG parses, they applied the more commonly used dependency parsing, yielding logic forms in DCS (dependency-based compositional semantics). Here the parse trees were latent variables. Despite the lack of training with logic forms, they had the best results then on GEOQUERY and JOBS.

The releases of GEO880, JOBS640, ROBOCUP300 and ATIS5418 greatly aroused research interest. But these are all focused closed domains. Cai and Yates (2013a) with their FREE917 dataset was *the first step towards large-scale open-domain* semantic parsing, but with two limitations: the questions were synthesized via looking at Freebase data; their approach still relied on logic forms. Berant et al. (2013) took a bigger step by releasing a *more realistic* corpus, **WEBQUESTIONS**, with 5810 questions crawled from Google Suggest. The answers to the questions were then extracted from Freebase using Amazon Mechanic Turk. The alignment between texts and relations, which was approached with the PMI method by Cai and Yates (2013a), was now approximated by textual tuples extracted from the ReVerb OpenIE system (Fader et al., 2011). Later, Berant and Liang (2014) improved their results by incorporating two paraphrase models: one based on learning word alignment from a large monolingual parallel corpus of 18 million wikianswers questions (Fader et al., 2013); the other based on monolingual distributional similarity measured by the continuous bag-of-word tool word2vec (Mikolov et al., 2013).

The WEBQUESTIONS dataset is more realistic since it was mined off the Google Suggest service, which prompts the most asked queries on the web. The dataset itself is difficult, since the best F_1 so far is less than 50% (Yao and Van Durme, 2014). However, the questions are not "difficult" in the sense that the lack of compositionality in the questions does not show the power of semantic parsing approaches. Yao and Van Durme (2014) showed that a simple information extraction approach, that learned the coupling of question features and Freebase topic properties, easily had equivalent performance with the semantic parsing methods. They presented a detailed back to back comparison on the results of the two methods in Yao et al. (2014). The analysis called for a dataset with more complex questions.

2.3.5. Hybrid QA (IR+NLP+KB)

High performance QA systems make use of as many types of resources as possible, especially with the prevailing popularity of modern search engines and enriching community-contributed knowledge on the web.

One early form of structured data on the web is fact tables, such as Wikipedia infoboxes. Lin (2002) collected 10 knowledge sources on the web, including biograph.com, CIA World Factbook, dictionary.com, 50states.com, etc, and manually verified that they were capable of answering 27% of TREC-9 (500 in total) and 47% of TREC-2001 questions (500 in total). Chu-Carroll et al. (2003) used the Cyc knowledge base as an answer sanity checker. Cucerzan and Agichtein (2005) employed a hybrid QA systems consisting of two components, one based on pattern matching unstructured text, the other one table extraction. They mined 200 million tables from 100 million web documents, Wikipedia and FactMonster. Unfortunately the table-based QA component performed poorly, with only 5.8% accuracy in TREC 05, in contrast with the 24.6% accuracy from the text-based component. Note that in their system the two components worked independently, which left an answer ranking problem about selecting which component to trust more.

Echihabi and Marcu (2003) had a different way, which was modeling all question and snippet pairs in a noisy channel. Structured data coming from World Fact Book and Biography.com etc was turned into natural language statement with a few manual generation templates. The MIT START system (Katz et al., 2006)

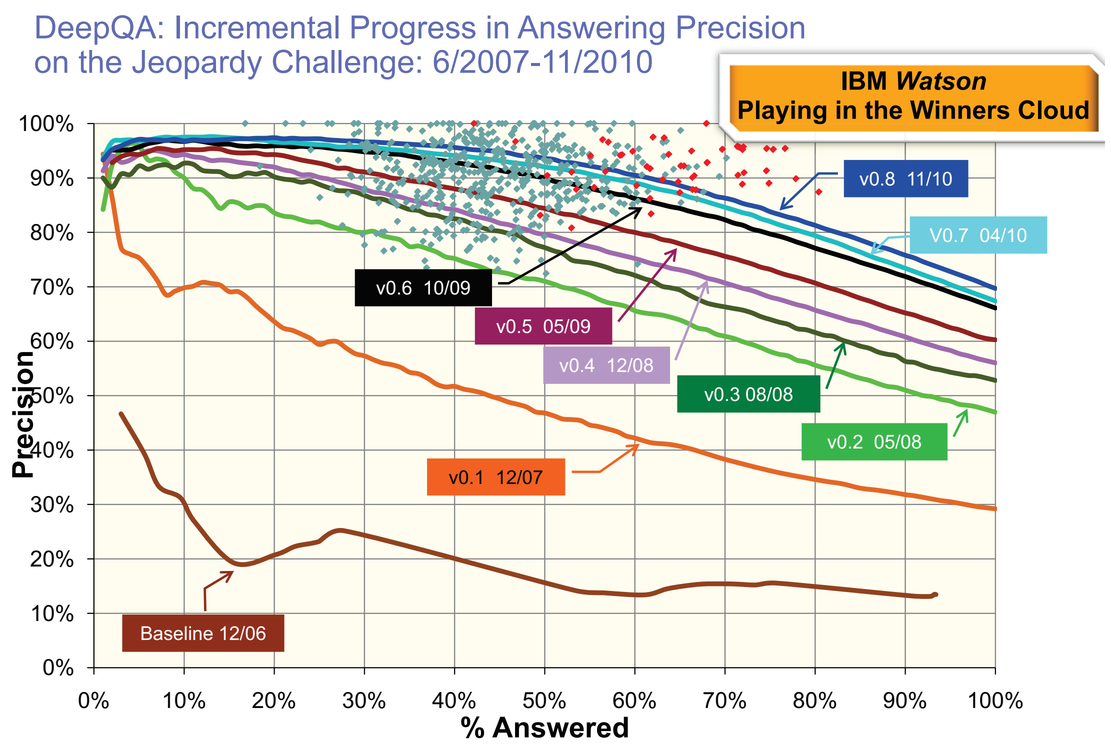


Figure 2.3.: Watson's incremental performance on 12,000 questions from 200 games' worth of blind data, with written permission for reuse from the IBM Journal of Research and Development article (Ferrucci, 2012).

went exactly the opposite direction: all semi-structured and structured information were stored in their Omnibase system (Katz et al., 2002), which provided a universal interface to the QA front end. For semi-structured information on the web, they made use of the language regularity of expressing certain things, for instance:

[*number*] people live in the metropolitan area of [*city*]

The hypothesis is that there are multiple different [number]-[city] pairs expressed in the same way on the web. In the way of natural language annotation, they harvested facts from the web and stored them in Omnibase. It was not clear how many annotation templates they used.

2.3.5.1. IBM Watson

IBM Watson marks the highest achievement so far in question answering. It is a masterpiece of teamwork, QA research, software engineering, and program management. The way how Watson approached the Jeopardy! challenge was thought-evoking. In my mind, the most significant achievement of Watson is that it has *proven possible to do better than humans* in such a high-intelligence task. A short introduction of Watson is an AI Magazine article on "Building Watson" (Ferrucci et al., 2010). Later, the IBM Journal of Research and Development published a dedicated issue (Issue 3.4, May-June 2012) with about 20 articles themed "This is Watson" (Ferrucci, 2012). This section overviews Watson, with a focus of what Watson has done differently.

The Watson project started in 2006 with adapting IBM's own in-house QA system, Piquant (Practical Intelligent Question Answering Technology). Piquant (Chu-Carroll et al., 2003) participated in TREC and was among the top 3 or 5 for several years. It had been under development by a four-person team for 6 years prior to the Jeopardy challenge. After 4 weeks of adaptation to Jeopardy, the result was discouraging: Piquant had a performance of 16% precision at 70% (16% Precision@70 in short) answered questions (see the baseline curve in Figure 2.3), while on average winning human players attempted between 40% and 50% of the questions in a game and scored between 85% and 95% correctly. This result led to a complete overhaul of their technical approach and architecture. The outcome was the extensible **DeepQA architecture** and the **AdaptWatson methodology** for rapid advancement and integration of core algorithms. By the end of 2007, the DeepQA framework was implemented and reconstructed as the v0.1 version of Watson (second curve from the bottom in Figure 2.3). In the next 5 years, the AdaptWatson method was employed over thousands of iterations of development, gradually pushing the confidence curve to more than 85% Precision@70: good enough to compete against humans.

The DeepQA framework consists of more than 100 core algorithmic classifiers (experts), each of which was effective on some certain types of questions. Given the huge size of test data, it was hard to draw insight based on statistical significance when, say, a new classifier was added on top of 99 other classifiers. Thus there existed an internal baseline system, called Watson answer-scoring baseline (**WASB**). WASB includes most components in the DeepQA framework, such as question analysis, passage retrieval, and candidate generation, but only one evidence-scoring component based on answer typing. Answer typing was the most used and intuitive technology employed in most TREC QA systems. That is why it was included in WASB. Usually one expert was able to improve the accuracy of 2% to 5% over WASB. Then with a couple hundred of different exports, the full-fledged system was human-level competitive.

In the following I briefly describe how Watson approached common QA subproblems.

Question Classification and Answer Types (Lally et al., 2012). Watson used both manual rules and logistic regression for question analysis. The whole tool suite included a parser using the English Slot Grammar (ESG) with an associated predicate-argument structure builder, a named entity recognizer, a co-reference resolution component, and a relation extraction component. The main problems were to detect the question focus (often pronouns in Jeopardy clues) and the lexical answer type (LAT). The baseline algorithm contained rules in more than $6,000$ Prolog clauses. These rules fail in complicated cases such as that multiple pronouns appear in one clue and wrong head word detection due to inaccurate parse. Thus additionally a logistic regression classifier was trained to detect LATs. Experiments in LAT detection showed an almost 10% (70.0% vs. 79.6%) improvement on F_1 over the baseline and this also improved the final end accuracy on $3,500$ questions from 67.5% to 71.0%.

One interesting question is whether Watson used a pre-defined LAT ontology.

The answer is mixed. On one hand, Watson used lots of pre-existing components. But a single common type system did not exist: they just mapped the LAT to each component's internal type. On the other hand, Jeopardy questions are very broad: 2500 distinct and explicit LATs were found in a 20,000 question sample. The most frequent 200 explicit LATs cover less than 50 percent of the data. Thus Watson directly used the lexical word as the type, an idea very similar to that of Pinchak and Lin (2006). All LATs detected from different components were in the end fed into a suite of type coercion algorithms (introduced later).

Deep Parsing (McCord et al., 2012). Watson used an augmented version of the English Slot Grammar (McCord, 1980). The Watson-specific parser produces a parse tree that is essentially a dependency structure with roughly the following augmented information: morphological analysis, constituent parses, POS tags, NER labels, chunking, predict-argument structure with argument frames (similar to SRL, but nouns and verbs can share frames (such as "celebration" and "celebrate")), word senses, etc. Parsing was used in every component of Watson, notably among question analysis, structured ranking in passage retrieval, answer typing and knowledge extraction. Watson used only the highest-ranked parse mostly for the sake of speed: ESG (implemented in C) parses about $5,000$ words per second on standard laptops. As a comparison, the Charniak parser is about 100 time slower when running in a single thread. To project this comparison onto other commonly used parsers (Stanford, Berkeley, MSTParser, etc), see Kong and Smith (2014). In my own experience, the Stanford CoreNLP pipeline processes about 25 tokens per second with POS and NER tagging and parsing.

Their evaluation of ESG vs. the Charniak parser on the Jeopardy and Wikipedia data reinforced my concern of modern statistical parser: overfitting on the Penn Treebank annotation. Parser correctness on 100 "segments" of two different data source is:

parser	Jeopardy	Wikipedia
ESG	92.0%	88.7%
Charniak	83.6%	81.1%

In terms of memory footprint, ESG occupies about 52MB memory during parsing. Its binary code and data take 5.7MB. They report that it "easily compiles and runs on a smartphone".

Knowledge Extraction (Fan et al., 2012). The output of ESG contains frames and slot fillers, which can be aggregated as strong evidence for certain tasks, such as answer typing (e.g., <Obama, isa, man>). One notable decision for knowledge extraction was that the frames were not restricted on only verbs, binary semantic relations, or a specific type hierarchy. Even the modifiers from ERG parses can be relations in frames. Watson's knowledge extraction component, PRISMATIC, was able to extract 995 million frames from 30GB of text, averaging 1.4 frames per sentence. When serving as a check for LATs, PRISMATIC consistently ranked in the top 3 among 17 different type coercion components in Watson and improved the overall accuracy by 2.4%. PRISMATIC was also used for candidate generation. For instance, it listed the answer Arabic in its top 20 Semitic languages to a question on this Semitic language. On a test set of $3,508$ questions, it generated answers for 42.6% of them.

Search and Candidate Generation (Chu-Carroll et al., 2012). Watson searches both unstructured source, i.e., text, and structured source, e.g., PRISMATIC and Freebase. In text search, it employed both Indri and Lucene for document and passage retrieval. One distinction between the Jeopardy questions and TREC QA questions is that some Jeopardy questions are very complicated with many constraints. Then it turned out the evidence had to be gathered from all over the documents, and thus document title serves as the answer. For instance, consider the question this country singer was imprisoned for robbery and in 1972 was pardoned by Ronald Reagan, the Wikipedia article for Merle Haggard mentions him as a country singer, his imprisonment for robbery, and his pardon by Reagan. This favors document retrieval over passage retrieval.

On a randomly selected set of 3,500 questions, all but 4.53% of the answers were Wikipedia titles. Thus the way Watson generated answer candidates is **very different** from traditional TREC approaches. TREC QA systems usually used named entities detected from retrieved passages. However, Watson used document metadata such as titles and anchor texts. This method turned out to be extremely effective: it generated answer candidates for 99.07% of all questions from a test set of $3,344$ questions and the final system had an accuracy of 62.65%. In contrast, when employing the TREC QA strategy, the system only achieved an accuracy of 54.9%.

Structured search mostly used SPARQL queries. A sample of $20,000$ Jeopardy questions showed that 11% of them were about 20 relations from Freebase, for instance, track, album, artist, containedBy, author. Thus special attention was paid to focus on the most frequent relations. Watson has a relation detector that has 80% precision and 60% recall, based on English expression that maps to the relation. This relation lookup component contributed 3.53% recall and 2.18% accuracy on the same test set of $3,344$ questions. PRISMATIC search (over curated KB) on the same test set provided answers for 43.75% of all questions, contributed 8.31% recall and 5.65% accuracy.

Overall, search and candidate generation in Watson stabilized at about 85 percent binary recall for the top 250 candidates.

Type Coercion (Murdock et al., 2012b). Jeopardy questions have a very broad and flexible range of answer types. In $20,000$ questions sampled: roughly $5,000$ different type words were used; more than half occurred less than three times; 12% occurred only once; 15% of questions did not explicitly have an LAT. Watson uses a technique called *type coercion*, a process that determines whether the answer candidate satisfies the answer type from the question, to tackle this challenge. Type coercion contains multiple components based on the ontology source or algorithms. Each component provides some evidence from its own area of expertise and all the evidences are gathered and ranked statistically. For instance, the WordNet hypernym and *isInstanceOf* relations are of high precision and low recall; Wikipedia and YAGO ontologies have a broader coverage on nominal entities; IBM's internal named entity recognizer (Prager et al., 2006) has more than 100 LATs; PRISMATIC contributes lexical evidence about answer typing through its *is_ a* relation. In some sense, type coercion in Watson is a miniature model of Watson's many experts as classifiers strategy.

On a test set of $3,508$ questions with the full Watson system, type coercion improved precision@70 from 81.5% to 87.5% and the total accuracy (precision@100) by 4.9%. Among the 4.9% improvement, each of 12 most interesting answer typing components contributed from 0.5% to almost 3%. The three most contributing sources were: YAGO (close to 3%), NER with about 100 types and PRISMATIC (both close to 2.5%). With the Watson Answer-Scoring Baseline (WASB) that only had the NER component enabled for answer typing, type coercion of more than 10 experts increased the accuracy by about 8%.

Evidence Scoring (Murdock et al., 2012a). The primary search in Watson generates a list of answer candidates. But they are not ranked at this point. Each of the answer candidates spawns a separate parallel process that includes this candidate in the search to retrieve further evidence. When evidence-bearing passages are returned, a passage-scoring component with four algorithms ranks them. The four algorithms are:

- Passage Term Match measures how often a candidate answer appears in the same passages as the question terms using roughly TF-IDF weighting.
- Skip-Bigram computes how many terms are shared in the syntactic-semantic

graphs from the question and the retrieved passages. It captures structural closeness as compared to bag-of-word co-occurring closeness.

- Textual Alignment aligns the question with passage according to the Waterman-Smith algorithm (Smith and Waterman, 1981) for DNA sequence matching.
- Logical Form scores two logic graphs with term weight and degree of match. The latter is a product of a term match score and a structural match score.

After all four algorithms are applied to each passage, the scores from each passage are merged with respect to each individual algorithm. Common ways of merging includes using *maximum*, *sum*, or *decaying sum* of scores from the passages. The best fit found by the Watson team was: summing for Skip-Bigram, decaying summing for Textual Alignment and Passage Term Match, and maximization for Logical Form.

Evaluation on $3,508$ questions showed the the four evidence scoring methods improved the full Watson system from 67.1% accuracy to 70.4% and WASB from 54.9% to 61.7%. Also these methods are more effective when used in evidence scoring than used in solely the primary search (2% improvement over full and 3% over WASB).

Knowledge Base and Inference (Kalyanpur et al., 2012a). Watson uses DBpedia, YAGO, a small portion of Freebase targeting spatial information, and some special handcrafted collection for U.S. presidents, works of Shakespeare, U.S. states, and countries. The major usage of KB in Watson is answer typing, which has been introduced above. Some manual efforts on differentiating mutually exclusive relations were also made for more precise answer typing. Another usage is on temporal and geospatial reasoning. They each separately had an 1% to 2% improvement over the full and base version of Watson system.

Question Decomposition (Kalyanpur et al., 2012b). Some Jeopardy questions are complicated. Thus Watson decomposes questions into parallel or nested sub-questions. Parallel sub-questions can be solved independently while nested sub-questions have to be solved in sequence. For parallel questions, rules are defined for independent subtrees, composable units and segments with qualifiers. These rules applied on 598 questions out of a test set of $1,269$. The end-to-end accuracy of Watson was also improved from 50.05% to 50.66% and accuracy on the decomposable subset increased from 56.68% to 58.02%. Heuristics are also defined for nested questions and they also improved the end performance from 50.05% to 50.82%. The overall impact of both parallel and nested decomposition was 1.4% gain.

Answer Ranking (Gondek et al., 2012). Watson was trained with $25,000$ questions comprising 5.7 million question-answer pairs and 550 features. Two biggest problems are the diverse types of questions and broad range of features. The following is a short summary of key ideas that could shed lights on current research:

- Answer Merging contains multiple components based on morphological and pattern analysis. It inspects every pair of answer candidates after Watson ranks them and make a binary judgement. If merge, then the candidate with higher initial ranking is kept.
- Ranking: logistic regression was found with consistently better performance among other classifiers, including SVM, boosting, neural nets, decision trees and locally weighted learning.
- Feature Normalization: existing features vary in their relative values, thus they are augmented with standardized features. Standardization is *per query*: each feature is normalized by subtracting the mean of all features of that type and scaling to unit variance.
- Missing Feature Indicator: a feature value of 0 could either mean a zero feature value, or missing feature. Thus an extra indicator was added to each

feature. It was shown that a sizable gain can be obtained with indicator features.

- One Model Per Question Class: each question class could favor different features thus an individual model is trained for each class. Another classifier automatically determines the question class with information from LAT etc.

- Feature Selection: for models with a lot of training data (and accordingly a lot of sparse features), Watson removes features per a pre-defined firing threshold besides using feature regularization. For models with a little training data and too many features, Watson uses the consistency subset attribute evaluator in Weka (Hall et al., 2009) for feature selection.

- Successive Refinement: Watson first refines the answer candidate list to its top 100 members, then near the end of learning considers only the top 5 candidates.

- Instance Weighting: the ratio of positive vs. negative instances in answer candidates produced by Watson is 1 to 94. Various methods including cost-sensitive learning and resampling were experimented with to level this imbalance. The final solution is to use an instance weighting of 0.5 for the negative instances in logistic regression.

- Evidence Diffusion: evidence can be shared between candidate answers. One example is that when asking the Sunan airport is in which *country*, there was overwhelming evidence about Pyongyang (a city), rather than North Korea (the country). Some criteria (such as *located-in* relation between answer candidates) was made to trigger evidence diffusion, where the feature values of North Korea were linearly combined with those of Pyongyang.

The above components fire at different stages of the pipeline. Figure 1 of Gondek et al. (2012) illustrates this. Overall, three (normalization, missing feature indication, successive refinement) of the above techniques improved the baseline system with 67% accuracy by 4.5%.

In Summary, the dominant principles in DeepQA are *massive parallelism*, *many experts*, *pervasive confidence estimation*, and *integration of shallow and deep knowledge* (Ferrucci, 2012). With the AdaptWatson methodology for rapid development and testing individual components, Watson matured into a super competitive question answering system within 5 years of time. I hope that their experience shared after the final game will become valuable lessons for future researchers.

2.4. A Different View: Linguistic Features vs. Machine Learning

In this section I look at the QA approaches from a different perspective: how deeper linguistic analysis was utilized and how larger the scale of learning became over the years. Research is a dynamic process. Especially in the field of computer science, technology rapidly evolves itself. Question answering started on machines with less than 1MB of memory and hand-crafted database. Nowadays the IBM Watson system has $2,880$ 3.5GHz POWER7 processor cores and 16 terabytes of RAM. As writing of this dissertation, researchers are already using terabytes of web data in large-scale machine learning the question answering problem.

I divide the analysis in a few dimensions, as shown in Table 2.6. In each category I select one or two most representative paper, illustrate the idea, and try to summarize the pros and cons of this category. I do realize that the categorization in Table 2.6 can be very crude. Some papers might fit into multiple categories. I also might have misread some papers and put them in a wrong place. But I believe the table captures the general trend in QA research in the past decades.

Linguistics			**Learning**		
			ad-hoc / pattern	**small scale (generative / discriminative)**	**large scale (discriminative)**
surface	IR QA	word	16 (1995-2010) A1	6 (2003-2006) A2	**A3**
surface	NLP QA	word/POS	4 (2001-2005) B1	4 (2001-2003) B2	**B3**
structure	NLP QA	chunking /NER	14 (2000-2008) C1	10 (2001-2010) C2	2 (2006-2013) **C3**
structure	NLP QA	syntax	16 (1993-2010) D1	8 (2001-2010) D2	15 (2003-2013) **D3**
structure	NLP QA	discourse	3 (2003-2009) E1		
meaning	NLP QA / KB QA	syntax, semantic roles /frames	5 (1976-2008) F1	4 (2004-2010) F2	3 (2007-2014) **F3**
meaning	KB QA	logic forms	13 (1961-2005) G1	9 (1996-2011) G2	8 (2007-2014) G3
meaning	KB QA	reasoning /inference	8 (1965-2003) H1	1 (2006) H2	

Table 2.6.: Categorization with respect to linguistic features and machine learning scales used in about 150 publications surveyed so far, shown with counts, year ranges and grid numbers. Detailed list is in § 2.4.3 on page 84. Techniques originated from this dissertation are marked with bold grid numbers (A3, B3, C3, D3, F3).

2.4.1. Linguistics: Word, POS, NER, Syntax, Semantics and Logic

From the linguistics point of view, the analysis goes from flat surface (lexicon and POS tags) to syntactic structures (shallow: chunking, NER; deeper: constituent or dependency parsing) to meaning representations (semantic frames, logic forms and reasoning). In my opinion, linguistic analysis is all about *language specification and generalization*: it provides tools to investigate various *specific aspects* of the language but also tries to *generalize* them well enough so that patterns are observed. QA research focuses on almost all kinds of linguistic analysis to draw meaningful patterns, either hand-crafted or machine learned.

Word-based patterns are often called *templates*. Lexicalized answer templates are often used to match against retrieved snippets (A1: Ravichandran and Hovy, 2002) for answer extraction. Automatic learning of these templates are possible (A2: Ravichandran et al., 2003). But languages can be expressed in so many ways that some regular expression based templates will always have trouble with matching against new data, especially those from a new domain. A lot of QA research falling into the A category is also due to that they are IR-based approaches (A1: Tellex et al., 2003): the first few years of TREC QA evaluated on the retrieved snippets instead of exact answer phrase.

Going from word to POS tags was partially motivated by the use of WordNet: it is more precise to fetch the synsets, and especially hypernym/synonym/etc relations if the word POS is known. Both Hammond et al. (B1: 1995) and Prager et al. (B1: 2001) explored in this direction. POS tags also help with morphological analysis. Ittycheriah et al. (B2: 2001a) used POS tags to help identify word lemmas during relevance scoring on retrieved passages.

Named entity labels are arguably the most widely used linguistic annotation in modern question answering systems. This is mostly motivated by answer typing (C1: Abney et al., 2000, Harabagiu et al., 2000, Prager et al., 2000) and they are used jointly with question classification (C2: Li and Roth, 2002). The linguistic patterns are so intuitive that any one can quickly write a few manual patterns. The number of entity labels also ranges from dozens to hundreds (c.f. Table 2.5 on page 53).

NER labels are good cues for answer types but they are too weak a signal for precisely pinpointing answer fragments with confidence. Syntax came to help. It was first applied on the question side to help question analysis (D1: Hovy et al., 2000), then gradually to the retrieved snippets, or even the whole corpus. Syntactic analysis gained its popularity in QA systems perhaps with the open availability of the MINIPAR dependency parser (Lin, 1993). QA based on syntax was first treated as a slot-filling like task (D1: Lin and Pantel, 2001): given a predicate-argument relation missing one argument, what is a correct answer to fill in the empty argument slot? Realizing the data sparsity problem for syntactic parses, various fuzzy matching methods were introduced based on either generative models such as EM used by Cui et al. (D2: 2004) or discriminative models such as SVM used by Moschitti et al. (D3: 2007).

Syntactic parses introduced two problems: sparsity of structures and lack of meaning representation. The first was mostly tackled by fuzzy matching. The second is essentially a problem of structured paraphrasing: different syntactic paths can mean the same thing. It was approached in two different ways: clustering with distributional similarity (D1: Lin and Pantel, 2001) or clustering with frames in semantic role labeling (SRL). SRL was majorly used in two types of work: IR-based learning to rank (F2: Bilotti et al., 2010b) and SRL-based graph matching (F2: Shen and Lapata, 2007). Although with reported success, coverage has been an issue for SRLs. For instance, Shen and Lapata (F2: 2007) observed that the Shalmaneser semantic parser (Erk and Pado, 2006) based on FrameNet (Fillmore et al., 2003) only applied to about 35% of the TREC data . Similarly, Bilotti et al. (F2: 2010b) used ASSERT (Pradhan et al., 2004), a PropBank (Palmer et al., 2005) parser, on 109 questions but only 48 were annotated with predicate-argument structures.

It is also interesting to see how research papers define "semantics" over the time. In early years NER was treated as semantic information. Also, it was common to use the term "lexical semantics" once WordNet was involved. In my categorization I treat semantic role labeling as a form of shallow semantics while logic form based semantic parsing as "deeper" semantics.

Going deeper from semantic roles to semantic parsing, logic forms start to emerge. Early years' logic forms were parser-dependent (G1: Woods, 1978), or database-dependent (G1: Zelle and Mooney, 1996). Later on research converged to a uniform representation: lambda calculus, either from general parses based on CCG (G3: Zettlemoyer and Collins, 2007) or dependency trees (G2: Liang et al., 2011). Logic forms are good at capturing constrained operators, such as "max" and "sum". Focus is put mostly on *query understanding*: usually the question side is parsed into a logic form and that transforms into database queries. On text-based semantic parsing, Harabagiu et al. (G1: 2000) parsed both the question and answer snippets into logic forms.

Based on logic forms, some degree of reasoning or inference can be made. Slagle (H1: 1965) used deductive logic with 68 given facts in their QA system. The system was able to compute how many fingers a man has given the number of hands on a man and the number of fingers on a hand. Moldovan and Rus (G1: 2001) transformed WordNet glosses into logic forms then to axioms to provide ground truth for question answering. Harabagiu and Hickl (G2: 2006) incorporated a textual entailment component in answer ranking and selection to improve the overall accuracy by 20%. The textual entailment component utilized features based on syntactic analysis and lexical semantics from paraphrasing. Another form of pure reasoning and inference exists in Expert Systems (Jackson, 1990), which is not in the scope of this literature review.

2.4.2. Learning: Ad-hoc, Small Scale and Large Scale

The way I categorize the use of machine learning is by:

- If there is no learning in the problem solving, then put it into the ad-hoc category;
- If there is, and if the model used is generative, or it is discriminative but the number of training instances is within hundreds, put it under small scale; otherwise large scale.

Almost all work exploring various linguistic features started with ad-hoc or pattern-based methods. In IR-based QA, systems rely on the IR front end for answer retrieval. Ranking of snippets largely came from search engine scores. In NLP-based QA, there were also manual mappings between question categories and answer types. Ranking on answer candidates correlated with answer typing conformability. Patterns were defined over surface forms, POS tags, NER labels, or syntactic paths. Also, Lin (2002) observed that the TREC questions followed the Zipf's law. Thus hand-written templates easily captured large proportion of questions. In KB-based QA, early systems had rule-based reasoning components for their internal logic forms. Systems could not generalize well beyond the scope of existing rules.

Machine learning was introduced to QA after the year 2000. I suspect it was mainly due to that TREC QA had run for 2 years by 2000, thus manual pattern based methods were exploited and shortcomings exposed accordingly. Also by that time, a few years of evaluation only left with a few hundred questions for training. Thus learning was kept small-scale. Still, various machine learning techniques were explored, including for instance:

- In IR QA: learning to rank passages with perceptron or SVM (Bilotti et al., 2010b).
- In NLP QA: generative models (with EM optimization) for matching syntactic patterns (Cui et al., 2004), or discriminative models (mostly Maxim Entropy based) for optimizing feature weights (Ittycheriah et al., 2001a, Ravichandran et al., 2003).
- In KB QA: various methods represented by Raymond Mooney (Zelle and Mooney, 1996, Ge and Mooney, 2005, Tang and Mooney, 2001, Kate and Mooney, 2006, Wong and Mooney, 2007) for semantic parsing on closed-domain small-scale datasets, such as GEOQUERY, JOBS, and ROBOCUP.

All of these examples were applied on a few hundred training instances. It was a first attempt to show success and results were proven promising. Still, the features used were mostly manually authored. All features defined were intuitive, but how many types of feature can a pair of hands write? I experimented with the idea of automatically generating all kinds of features based on lexicon, POS tags, NER labels, and dependency parses. The feature space was exploded. As a consequence, lots of training data was needed. In § 4.6 on page 180 I will show the experiments conducted on almost 100 thousand Jeopardy questions with about 35 million features: this idea is able to provide a competitive statistical backbone for modern QA systems.

Another direction in large-scale machine learning question answering patterns is syntactic structure matching with latent alignment variables (Wang et al., 2007, Wang and Manning, 2010) or tree similarity models (Shen and Klakow, 2006, Heilman and Smith, 2010, Severyn and Moschitti, 2013). Discriminative models help with incorporating multiple external resources in feature engineering, especially given that there are a handful of lexical resources for English QA. They also help learning from features in a high dimensional space based on convolutional tree kernels (Moschitti et al., 2007).

Finally, in the work of semantic parsing, there have been releases of more open-domain and much larger datasets (Cai and Yates, 2013a, Berant et al., 2013, Fader et al., 2013). Semantic parsing researchers focus on more challenging and realistic problems, in contrast to rule-based systems in the 1960s.

To summarize, as Liang and Potts (2014) stated about the progress in bridging machine learning and compositional semantics together, with more and more accessible data for both supervised learning and unsupervised mining, I believe QA systems in the future would bear more statistical backbone, be more generalized in open domain questions, and behave as a more natural human computer interface than keyword based search engines.

2.4.3. Appendix: Publications Per Grid

- A1: Burke et al. (1997), Kwok et al. (2001), Brill et al. (2001), Clarke et al. (2001a,b), Hermjakob et al. (2002), Magnini et al. (2002), Soubbotin and Soubbotin (2002), Ravichandran and Hovy (2002), Sneiders (2002b,a), Zheng (2002), Tellex et al. (2003), Xu et al. (2003), Katz et al. (2006), Bunescu and Huang (2010)

- A2: Ravichandran et al. (2003), Zhang and Lee (2003), Soricut and Brill (2004), Wu et al. (2005), Buscaldi and Rosso (2006), Wei et al. (2006)

- B1: Hammond et al. (1995), Prager et al. (2001), Peng et al. (2005), Prager et al. (2006)

- B2: Ittycheriah et al. (2001a), Chu-Carroll et al. (2003), Girju (2003), Ramakrishnan et al. (2003)

- C1: Abney et al. (2000), Harabagiu et al. (2000), Hovy et al. (2000), Srihari and Li (2000), Prager et al. (2000), Hermjakob (2001), Magnini et al. (2002), Xu et al. (2002), Leidner et al. (2003), Yang et al. (2003), Cucerzan and Agichtein (2005), Schlaefer et al. (2006), Prager et al. (2006), Kosseim and Yousefi (2008)

- C2: Ittycheriah et al. (2001b), Mann (2001), Pasca and Harabagiu (2001), Li and Roth (2002), Chu-Carroll et al. (2003), Fleischman et al. (2003), Nyberg et al. (2003), Radev et al. (2005), Schlobach et al. (2007), Tellez-Valero et al. (2010)

- C3: Pinchak and Lin (2006), Yao et al. (2013d)

- D1: Kupiec (1993), Harabagiu et al. (2000), Hovy et al. (2000), Lin and Pantel (2001), Hermjakob (2001), Hermjakob et al. (2002), Li (2003), Leidner et al. (2003), Katz and Lin (2003), Yang et al. (2003), Jijkoun et al. (2004), Punyakanok et al. (2004), Peng et al. (2005), Bouma et al. (2005), Bunescu and Huang (2010), Kaisser (2012)

- D2: Hermjakob (2001), Echihabi and Marcu (2003), Mollá et al. (2003), Cui et al. (2005), Wu et al. (2005), Sun et al. (2005), Harabagiu and Hickl (2006), Bunescu and Huang (2010)

- D3: Zhang and Lee (2003), Blunsom et al. (2006), Pinchak and Lin (2006), Shen and Klakow (2006), Moschitti et al. (2007), Wang et al. (2007), Huang et al. (2008), Li et al. (2008), Pan et al. (2008), Pinchak et al. (2009b), Heilman and Smith (2010), Wang and Manning (2010), Fader et al. (2013), Severyn and Moschitti (2013), Yao et al. (2013c)

- E1: Yang et al. (2003), Sun and Chai (2007), Quarteroni and Manandhar (2009)

- F1: Plath (1976), Lopez et al. (2005), Bilotti et al. (2007), Frank et al. (2007), Hartrumpf (2008)

- F2: Narayanan and Harabagiu (2004), Sun et al. (2005), Shen and Lapata (2007), Bilotti et al. (2010a)

- F3: Moschitti et al. (2007), Fader et al. (2013), Yao and Van Durme (2014)

- G1: Green et al. (1961), Green (1969), Winograd (1972), Lehnert (1977), Woods (1977, 1978), Winiwarter (1999), Harabagiu et al. (2000), Zajac (2001), Moldovan and Rus (2001), Moldovan et al. (2002), Mollá et al. (2003), Mollá and Van Zaanen (2005)

- G2: Zelle and Mooney (1996), Tang and Mooney (2001), Thompson and Mooney (2003), Ge and Mooney (2005), Kate and Mooney (2006), Wong and Mooney (2007), Lu et al. (2008), Clarke et al. (2010), Liang et al. (2011)

- G3: Zettlemoyer and Collins (2007), Kwiatkowski et al. (2010, 2011, 2013), Cai and Yates (2013a,b), Berant et al. (2013), Berant and Liang (2014)
- H1: Slagle (1965), Green et al. (1961), Bruce (1972), Pollack (1986), Lin and Pantel (2001), Moldovan and Rus (2001), Moldovan et al. (2002, 2003b)
- H2: Harabagiu and Hickl (2006)

3. Feature-driven QA from Unstructured Data: Text

This chapter illustrates the idea of automatically generating and learning features from basic linguistic annotations on unstructured text (§ 3.1). We first show an alignment model based on Tree Edit Distance that achieved state-of-the-art result on an answer sentence ranking task (§ 3.2). Then we cast the task of answer extraction as a sequence tagging task and report QA performance with automatically generated features, including ablation test results (§ 3.3). Finally we backtrace from the answer extraction back end to the information retrieval front end and demonstrate how the automatically learned features can help IR performance (§ 3.4). The accompanying implementation is jacana-qa.

Each of the sections § 3.2 to § 3.4 defines their own task for evaluation. I reuse the example in § 3.4 to illustrate the tasks here:

Q: When was **Alaska purchased**?

Retrieved Sentences:

S_1: Eventually **Alaska** Airlines will allow all travelers who have **purchased** electronic tickets through any means.

S_2: **Alaska**'s online check-in program works like this: A passenger who already has **purchased** an electronic ticket online, through a ticket agent or over the phone logs on to Alaska 's Web site from a home or office computer .

S_3: Russia and the United States signed a treaty selling **Alaska** to the U.S. for $7.2 million on March 30, 1867.

...

Answer: March 30, 1867 (or simply 1867).

The three tasks in this chapter are:

1. § 3.2: answer-bearing sentence ranking. Given the question Q and a list of *already retrieved* sentences (S_1, S_2, S_3, ...), *judge* and *rank* whether each sentence contains an answer or not. It is essentially a re-ranking task: given a list of retrieved sentences, re-order them so that sentences with higher chances of bearing an answer are ranked higher. Ideal output for the above example would be: ((S_3, true), (S_1, false), (S_2, false)) or ((S_3, true), (S_2, false), (S_1, false)). Evaluation measures are Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR), defined in § 2.2.3.2 on page 40. They take a ranked list as input, and output a score between 0 and 1.

2. § 3.3: answer extraction. Given the question Q and a list of *already retrieved* sentences ($S1$, $S2$, $S3$, ...), extract the answer fragment that answers the question. Ideal output for the above example would be: March 30, 1867. Evaluation measure is precision, recall and F_1, defined in § 2.2.3.1 on page 38.

3. § 3.4: (shallow structured) information retrieval. Given the question Q and a corpus, retrieve all relevant sentences and rank them by relevance. Ideal

output for the above example would be (S_3, S_1, S_2, ...) or (S_3, S_2, S_1, ...). Evaluation measures are MAP and MRR defined in § 2.2.3.2 on page 40. This often requires an *exhaustive* relevance judgment of *all* sentences in a corpus given a question. Lin and Katz (2006) contributed such a judgement for 109 TREC QA questions.

A minimal QA system normally contains either 1 and 2, or 2 and 3, depending on whether the QA system employs its own information retrieval engine. Normally, when using the web as a corpus, QA systems often resort to (commercial) web search engines. They do not have much power in deciding how the sentences are retrieved beyond framing the queries properly. Thus such systems would usually consist of component 1 to perform answer-bearing sentence reranking if the *order* of original retrieved sentences is not good enough. This has the limitation that if the relevant sentences are not retrieved, then no matter how good component 1 is, there is no way for the answer extraction component to pinpoint the exact answer fragment.

On the other hand, if the corpus is relatively small, such as the one used in TREC QA, then open-source text indexing tools can be directly employed. QA systems would have much better control over how the answer-bearing sentences are retrieved. Thus component 3 is desirable in such setting. Further more, the answer extraction component is the most essential one in either setting. This chapter visits all three components and covers both settings.

3.1. Introduction

The success of IBM's Watson system (Ferrucci et al., 2010) has illustrated a continued public interest in Question Answering. Watson is a sophisticated piece of software engineering consisting of many components tied together in a large parallel architecture. It took many researchers working full time for years to construct.

Such resources are not available to individual academic researchers. If they are interested in evaluating new ideas on some aspect of QA, they must either construct a full system, or create a focused subtask paired with a representative dataset. We follow the latter approach and focus on the task of answer extraction, i.e., producing the exact answer strings for a question.

We propose the use of a linear-chain Conditional Random Field (CRF) (Lafferty et al., 2001) in order to cast the problem as one of *sequence tagging* by labeling each token in a candidate sentence as either Beginning, Inside or Outside (BIO) of an answer. This is to our knowledge the first time a CRF has been used to extract answers.[1] We utilize not only traditional contextual features based on POS tagging, dependency parsing and Named Entity Recognition (NER), but most importantly, features extracted from a Tree Edit Distance (TED) model for aligning an answer sentence tree with the question tree. The linear-chain CRF, when trained to learn the associations between question and answer types, is a robust approach against error propagation introduced in the NLP pipeline. For instance, given an NER tool that always (i.e., in both training and test data) recognizes the pesticide DDT as an ORG, our model realizes, when a question is asked about the type of chemicals, the correct answer might be *incorrectly but consistently* recognized as ORG by NER. This helps reduce errors introduced by wrong answer types, which were estimated as the most significant contributor (36.4%) of errors in the then state-of-the-art QA system of Moldovan et al. (2003a).

The features based on TED allow us to draw the connection between the question and answer sentences *before* answer extraction, whereas traditionally the exercise of *answer validation* (Magnini et al., 2002, Penas et al., 2008, Rodrigo et al., 2009) has been performed *after* as a remedy to ensure the answer is really "about" the question.

QA systems often are required for a fast run time. For instance, Watson was

[1] CRFs have been used in judging answer-bearing sentences (Shima et al., 2008, Ding et al., 2008, Wang and Manning, 2010), but not extracting exact answers from these sentences.

designed under the constraint of a 3 second response time, arising from its intended live use in the television game show, `Jeopardy!`. Motivated by this desire, we base our TED implementation on the dynamic-programming approach of Zhang and Shasha (1989), which helps our final system process (including alignment, feature extraction, and classification) 200 QA pairs per second on standard desktop hardware, when both the question and sentence are syntactically pre-parsed.

In the following we first provide background on the TED model, going on to evaluate our implementation against prior work in the context of question answer sentence ranking (QASR), achieving state of the art in that task. We then describe how we couple TED features to a linear-chain CRF for answer extraction, providing the set of features used, and finally experimental results on an extraction dataset we make public (together with the software) to the community. Related prior work is interspersed throughout each section.

3.2. Tree Edit Distance Model

Tree Edit Distance (§3.2.1) models have been shown effective in a variety of applications, including textual entailment, paraphrase identification, answer ranking and information retrieval (Reis et al., 2004, Kouylekov and Magnini, 2005, Heilman and Smith, 2010, Augsten et al., 2010). We chose the variant proposed by Heilman and Smith (2010), inspired by its simplicity, generality, and effectiveness. Our approach differs from those authors in their reliance on a greedy search routine to make use of a complex tree kernel. With speed a consideration, we opted for the dynamic-programming solution of Zhang and Shasha (1989) (§3.2.1). We added new lexical-semantic features §(3.2.2) to the model and then evaluated our implementation on the QASR task, showing strong results §(3.2.3). The QASR task aims to judge whether a sentence contains the answer to the question or not.

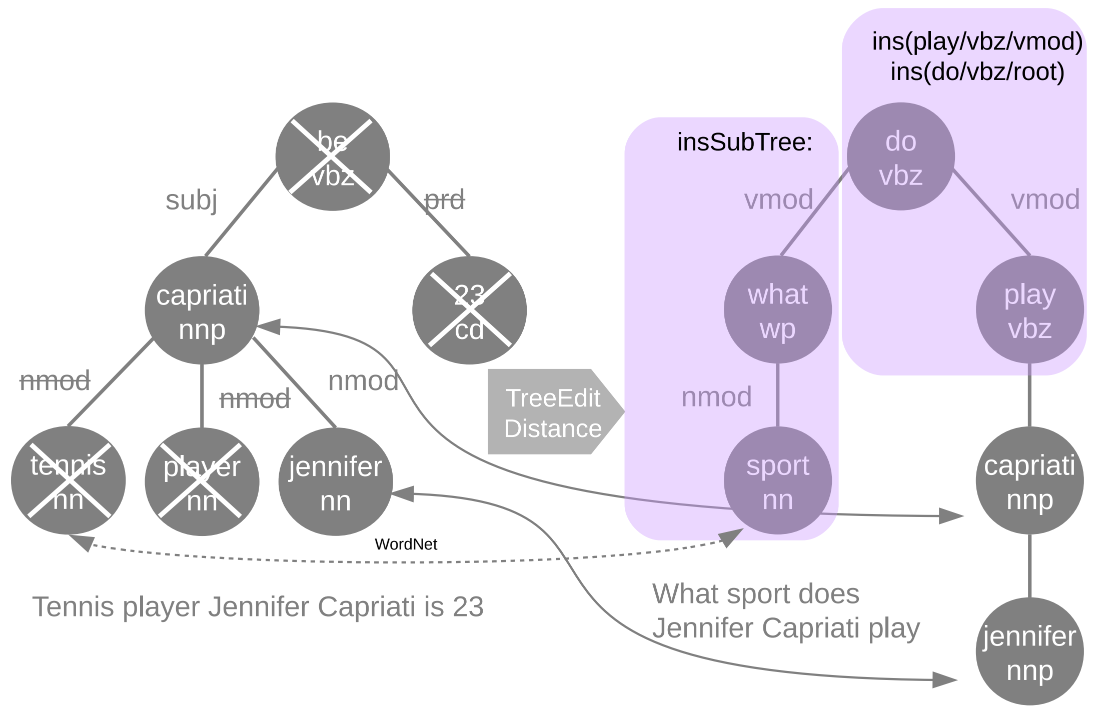


Figure 3.1.: Edits transforming a source sentence (left) to a question (right). Each node consists of: lemma, POS tag and dependency relation, with root nodes and punctuation not shown. Shown includes deletion (× and ~~strikethrough~~ on the left), alignment (arrows) and insertion (shaded area). Order of operations is not displayed. The standard TED model does not capture the alignment between *tennis* and *sport* (see Section 3.2.2). But it can be enabled by WordNet relations shown in dashed arrows. Step-by-step edit sequence can be found in Figure 3.2.

It has implications for providing better ranking from a retrieved passage list in information retrieval. Here we use it to test whether we have implemented the TED model correctly. Later in the next section (§ 3.3) we apply the same TED model in answer extraction and show much improved result.

3.2.1. Cost Design and Edit Search

Following Bille (2005), we define an *edit script* between trees T_1, T_2 as the edit sequence transforming T_1 to T_2 according to a *cost function*, with the total summed cost known as the *tree edit distance*. *Basic edit* operations include: *insert*, *delete*

Feature	Description
distance	tree edit distance from answer sentence to question
ren{Noun,Verb,Other}	# edits changing POS from or to noun, verb, or other types
ins{N,V,Punc,Det} insOtherPos	# edits inserting a noun, verb, punctuation mark, determiner or other POS types
delN, delV, ...	deletion mirror of above
ins{N,V,P}Mod ins{Sub, Obj} insOtherRel	# edits inserting a modifier for {noun, verb, preposition}, subject, object or other relations
delNMod, ...	deletion mirror of above
renNMod, ...	rename mirror of above
XEdits	# basic edits plus sum of ins/del/ren edits
alignNodes align{Num, N, V, Proper}	# aligned nodes, and those that are numbers, nouns, verbs, or proper nouns

Table 3.1.: Features for ranking QA pairs.

and *rename*.

With T a dependency tree, we represent each node by three fields: lemma, POS and the type of dependency relation to the node's parent (DEP). For instance, Mary/nnp/sub is the proper noun *Mary* in subject position.

Basic edits are refined into 9 types, where the first six (INS_LEAF, INS_SUBTREE, INS, DEL_LEAF, DEL_SUBTREE, DEL) insert or delete a leaf node, a whole subtree, or a node that is neither a leaf nor part of a whole inserted subtree. The last three (REN_POS, REN_DEP, REN_POS_DEP) serve to rename a POS tag, dependency relation, or both.

We begin by uniformly assigning basic edits a cost of 1.0,[2] which brings the cost of a full node insertion or deletion to 3 (all the three fields inserted or deleted). We allow renaming of POS and/or relation type iff the lemmas of source and target nodes are identical. This is aimed at minimizing node variations introduced by morphology differences, tagging or parsing errors. When two nodes are identical and thus do not appear in the edit script, or when two nodes are renamed due to

[2]This applies separately to each element of the tripartite structure; e.g., deleting a POS entry, inserting a lemma, etc.

the same lemma, we say they are *aligned* by the tree edit model (see Figure 3.1). These cost decisions conform with the triangle inequality.

We used Zhang and Shasha (1989)'s dynamic programming algorithm (implemented by Augsten et al. (2010)) to produce an optimal edit script with the lowest tree edit distance. The approach explores both trees in a bottom-up, post-order manner, running in time:

$O(|T_1|\,|T_2|\min(D_1, L_1)\min(D_2, L_2))$

where $|T_i|$ is the number of nodes, D_i is the depth, and L_i is the number of leaves, with respect to tree T_i.

Figure 3.2 shows an example of transforming the dependency tree of Tennis player Jennifer Capriati is 23 (source) to the tree of What sport does Jennifer Capriati play (target) in 9 steps. In *post order*, it performs the following edits:

1. On the left branch:
   a) insert the left branch (what sport) of the target tree into the source tree (steps 1 and 2);
   b) remove the non-overlapping node of the left branch of the source tree (tennis player, steps 3 and 4);
   c) rename the POS tag of the jennifer node in the left branch of the source tree since it appears in both trees (step 5);
2. On the right branch:
   a) insert the play node of the target tree into the source tree (step 6);
   b) delete the right branch of the source tree (23 and be, steps 7 and 8);
3. On the root node: insert the root node of the target tree to the source tree (do, step 9).

The total 9 steps form the final edit script for transforming from the source tree to the target tree.

Additionally, we fix the cost of stopword renaming to 2.5, even in the case of identity, regardless of whether two stopwords have the same POS tags or relations. Stopwords tend to have fixed POS tags and dependency relations, which often leads to less expensive alignments as compared to renaming content terms. In practice this gave stopwords "too much say" in guiding the overall edit sequence. The reason behind this was: when the algorithm walks up both trees and decides to rename a pair of stopwords, the next expected search direction is fixed by the current edit. In some situations, it might fail to rename a pair of content words, and end up with first deleting it from T_1, then inserting the content word with the same lemma from T_2 to T_1 (instead of just renaming). This is an undesirable effect because less meaningful stopwords are aligned but content words are not. A cost of 2.5 for stopword is less than 3 so that stopwords do not get deleted/inserted, but larger than 2 so content words (with a renaming cost of at most 2) get renamed first.

The resultant system is fast in practice, processing $10,000$ pre-parsed tree pairs per second on a contemporary machine.[3] In later tasks, feature extraction and decoding will slow down the system, but the final system was still able to process 200 pairs per second.

3.2.2. TED for Sentence Ranking

The task of Question Answer Sentence Ranking (QASR) takes a question and a set of source sentences, returning a list sorted by the probability likelihood that each sentence contains an appropriate answer. Prior work in this includes that of: Punyakanok et al. (2004), based on mapping syntactic dependency trees; Wang et al. (2007) utilizing Quasi-Synchronous Grammar (Smith and Eisner, 2006);

[3]As a relevant note, later we tried to optimize these costs over a development set with grid searching cost parameters. The search was done in conjunction with using WordNet relations (described in the next section). Experiments showed that no significant improvement over the original uniform cost setting. Soon later we decided to move on with discriminative aligners that optimize these costs in a more principled way. The work is described in Chapter 4.

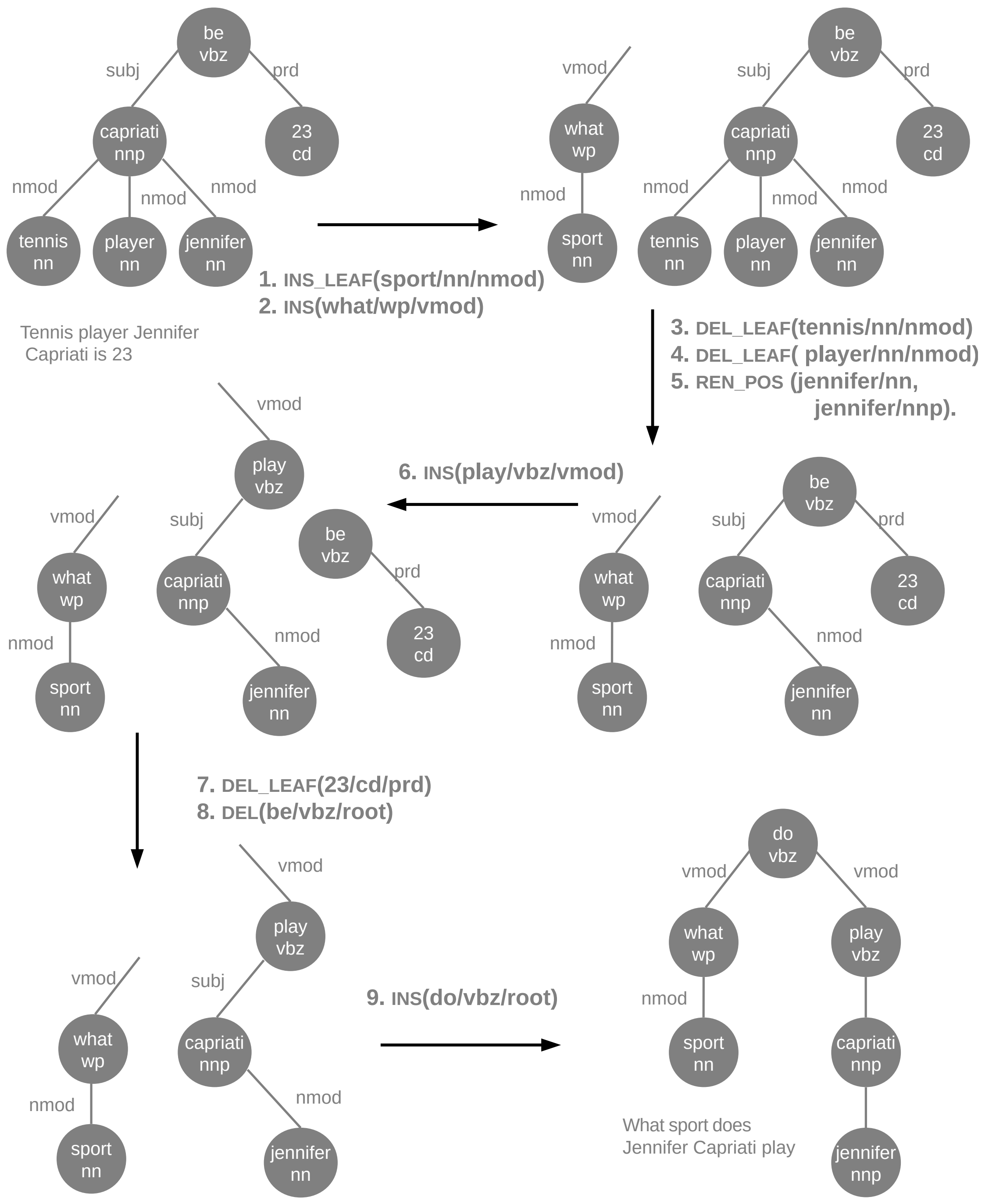


Figure 3.2.: Tree edit sequence according to the Zhang and Shasha (1989) algorithm. This illustration transforms the tree of "Tennis player Jennifer Capriati is 23" to the tree of "What sport does Jennifer Capriati play" in 9 steps. The parse trees and POS tags are real thus contain errors. Note that steps 1 and 2 can be merged to INS_SUBTREE(what/wp/vmod).

Heilman and Smith (2010) using TED; and Shima et al. (2008), Ding et al. (2008) and Wang and Manning (2010), who each employed a CRF in various ways. Wang et al. (2007) made their dataset public, which we use here for system validation. To date, models based on TED have shown the best performance for this task.

Our implementation follows Heilman and Smith (2010), with the addition of 15 new features beyond their original 33 (see Table 3.1). Based on results in DEV, we extract edits in the direction from the source sentence to the question.

In addition to syntactic features, we incorporated the following lexical-semantic relations from WordNet: *hypernym* and *synonym* (nouns and verbs); *entailment* and *causing* (verbs); and *membersOf*, *substancesOf*, *partsOf*, *haveMember*, *haveSubstance*, *havePart* (nouns). Such relations have been used in prior approaches to this task (Wang et al., 2007, Wang and Manning, 2010), but not in conjunction with the model of Heilman and Smith (2010).

These were made into features in two ways: **WNsearch** loosens renaming and alignment within the TED model from requiring strict lemma equality to allowing lemmas that shared any of the above relations, leading to renaming operations such as REN_...(`country`, `china`) and REN_...(`sport`, `tennis`); **WNfeature** counts how many words between the sentence and answer sentence have each of the above relations, separately as 10 independent features, plus an aggregate count for a total of 11 new features beyond the earlier 48.

These features were then used to train a logistic regression model using Weka (Hall et al., 2009). We kept the default parameter setting but tried various features on the dev set. The input is (*question, sentence, does_ sentence_ answer_ question*) tuples and the prediction is on whether a new sentence contains the answer to the question.

set	**source**	**#question**	**#pairs**	**%positive**	**length**
TRAIN-ALL	TREC8-12	1229	53417	12.0	any
TRAIN	TREC8-12	94	4718	7.4	≤ 40
DEV	TREC13	82	1148	19.3	≤ 40
TEST	TREC13	89	1517	18.7	≤ 40

Table 3.2.: Distribution of data, with imbalance towards negative examples (sentences without an answer).

3.2.3. QA Sentence Ranking Experiment

We trained and tested on the dataset from Wang et al. (2007), which spans QA pairs from TREC QA 8-13 (see Table 3.2). Per question, sentences with non-stopword overlap were first retrieved from the task collection, which were then compared against the TREC answer pattern (in the form of Perl regular expressions). If a sentence matched, then it was deemed a (noisy) positive example. Finally, TRAIN, DEV and TEST were manually corrected for errors. Those authors decided to limit candidate source sentences to be no longer than 40 words.[4] Keeping with prior work, those questions with only positive or negative examples were removed, leaving 94 of the original 100 questions for evaluation.

The data was processed by Wang et al. (2007) with the following tool chain: POS tags via MXPOST (Ratnaparkhi, 1996); parse trees via MSTParser (McDonald et al., 2005) with 12 coarse-grained dependency relation labels; and named entities via Identifinder (Bikel et al., 1999). Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) (defined in detail in § 2.2.3.2 on page 40) are reported in Table 3.3. Our implementation gives state of the art performance, and is furthered improved by our inclusion of semantic features drawn from WordNet.[5]

[4]TRAIN-ALL is not used in QASR, but later for answer extraction; TRAIN comes from the first 100 questions of TRAIN-ALL.

[5]As the test set is of limited size (94 questions), then while our MAP/MRR scores are 2.8% $\sim$ 5.6% higher than prior work, this is not statistically significant according to the Paired Randomization Test (Smucker et al., 2007), and thus should be considered *on par* with the current state of the art.

System	MAP	MRR
Wang et al. (2007)	0.6029	0.6852
Heilman and Smith (2010)	0.6091	0.6917
Wang and Manning (2010)	0.5951	0.6951
this dissertation (48 features)	0.6319	0.7270
+WNsearch	**0.6371**	0.7301
+WNfeature (11 more feat.)	0.6307	**0.7477**

Table 3.3.: Results on the QA Sentence Ranking task.

3.3. Answer Extraction as Sequence Tagging

In this section we move from *ranking* source sentences, to the next QA stage: *answer extraction*. Given our competitive TED-based alignment model, the most obvious solution to extraction would be to report those spans aligned from a source sentence to a question's *wh*-terms. However, we show that this approach is better formulated as a (strongly indicative) feature of a larger set of answer extraction signals. The result is a compact system combining a state of the art model for measuring syntactic correspondences, with a well-understood method for sequence tagging. Together these form a question answering solution that is competitive with more complicated QA pipelines that have been used in the past.

3.3.1. Sequence Model

Edit scripts resulting from the TED model is rich in useful information for answer extraction. Intuitively, when transforming from answer tree to question tree, answers should appear mostly in the deleted edits, sometimes also in aligned words (unedited or renamed).[6] Since the edit script is also a sequence of different edits, which appears in an order of deterministic post-order tree traversal, it also has a Markov property that the next possible edit only depends on the current edit node. A direct way is to build a Markov model over the edit script and tag all possible

[6]For instance, What is the largest city in the US? – New York City. Here City appears in the aligned edit.

answers. However, this is not desirable in practice: the tree edit sequence does not guarantee that answers span a continuous region in the original word order (a string edit distance algorithm guarantees this but loses the nice properties of tree edits). For instance, suppose the fragment in 90 days with a dependency parse $90 \xrightarrow{\text{nmod}} \text{days} \xrightarrow{\text{pmod}} \text{in}$ is the answer to a when question, then the edit script would contain a sequence roughly like del(90), del(days), del(in). Here the beginning of answer (in) comes *after* the inside of answer (90 days), which is counter-intuitive to model (imagine learning the transition probability from I-ANSWER to B-ANSWER).

Thus instead of building a Markov model over the edit script, we build one over the original sentence and heavily utilize features extracted from the edit script. This way nicely preserves transition directions without losing the power from tree edit movement. Further, to utilize a good quantity of features, and to also connect the answer sentence with the question sentence through tree edits, we need to inspect an entire input at each token. Thus a linear-chain CRF model (Lafferty et al., 2001) is a better fit to HMMs. We only used a first-order CRF, as there is not too much clean training data for higher-order CRFs. The task is defined as tagging each token in a sentence with one of the following labels: B-ANSWER (beginning of answer), I-ANSWER (inside of answer), O (outside of answer). Figure 3.3 shows an example.

Some TREC questions (starting from 2004) were also designed in a way that for each document, several related questions were asked in a row. Thus subsequent answers should still be on the same topic and it is more likely to find the answer from the same document, i.e., there are inter-sentence dependencies in consecutive questions and answers. Ideally, a skip-chain CRF (Sutton and Mccallum, 2004, Galley, 2006) would model this property better. However, the way that this QA dataset was processed drops this property, i.e., each question is assumed independent. Thus we think a linear-chain CRF would be sufficient for this task.

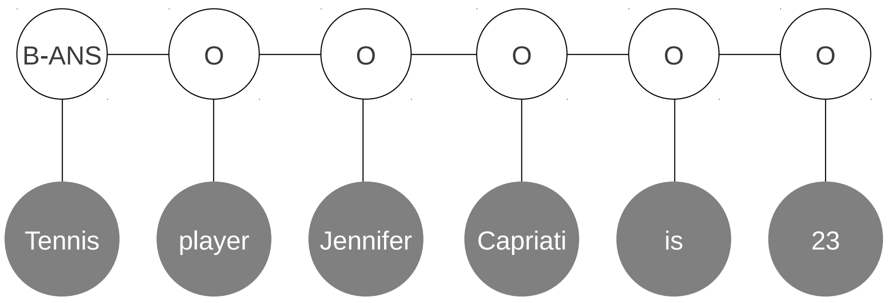


Figure 3.3.: An example of linear-chain CRF for answer sequence tagging.

3.3.2. Feature Design

In this subsection we describe the local and global features used by the CRF.

Chunking We use the POS/NER/DEP tags directly just as one would in a chunking task. Specifically, suppose t represents the current token position and $pos[t]$ its POS tag, we extract unigram, bigram and trigram features over the local context, e.g., $pos[t-2], pos[t-2] : pos[t-1]$, and $pos[t-2] : pos[t-1] : pos[t]$. Similar features are extracted for named entity types ($ner[t]$), and dependency relation labels ($dep[t]$).

Unlike chunking, where memorizing lexicons for phrase types is helpful to the task (e.g., the word John often is of type B-NP), we did not include any lexicalized features in answer extraction as it is harmful to memorize answers for a general question.

Our intuition is these chunking features should allow for learning which types of words tend to be answers. For instance, we expect adverbs to be assigned lower feature weights as they are rarely a part of answer, while prepositions may have different feature weights depending on their context. For instance, *of* in *kind of silly* has an adjective on the right, and is unlikely to be the Beginning of an answer to a TREC-style question, as compared to *in* when paired with a question on time, such as seen in an answer *in 90 days*, where the preposition is followed by a number then a noun. Later on in Table 3.8 on page 111 we should that this

Feature	Description
edit=X	type of edit feature. X: DEL, DEL_SUBTREE, DEL_LEAF, REN_POS, REN_DEP, REN_POS_DEP or ALIGN.

X_pos=?
X_ner=?
X_dep=?
	Delete features. X is either DEL, DEL_SUBTREE or DEL_LEAF. ? represents the corresponding POS/NER/DEP of the current token.

Xpos_from=$?_f$
Xpos_to=$?_t$
Xpos_f_t=$?_f$_$?_t$
Xner_from=$?_f$
Xner_to=$?_t$
Xner_f_t=$?_f$_$?_t$
Xdep_from=$?_f$
Xdep_to=$?_t$
Xdep_f_t=$?_f$_$?_t$
	Rename features. X is either REN_POS, REN_DEP or REN_POS_DEP. Suppose word f in answer is renamed to word t in question, then $?_f$ and $?_t$ represent corresponding POS/NER/DEP of f and t.

align_pos=?
align_ner=?
align_dep=?
	Align features. ? represents the corresponding POS/NER/DEP of the current token.

Table 3.4.: Features based on edit script for answer sequence tagging.

intuition holds.

Question-type Chunking features do not capture the connection between question word and answer types. Thus they have to be combined with question types. For instance, how many questions are usually associated with numeric answer types. Following a limited version of prior work in question-type classification, we intended to avoid sophisticated engineering of question classification but chose to do a simple question analysis and encode the question word lexicon into features. We skimmed through TRAIN and DEV and found the following question types that clearly indicate the types of answers: who, whom, when, where, how many, how much, how long, and then for each token, we combine the question term with its chunking features described in (most tokens have different features because they have different POS/NER/DEP types). One feature example of the QA pair how_much/100_dollars for the word 100 would be: qword=how_much|pos[t]=CD|pos[t+1]=NNS. We expect high weight for this feature since it is a good pattern for matching question type and answer type. Similar features also apply to what, which, why and how questions, even though they do not indicate an answer type as clearly as how much does.

Some extra features are designed for what/which questions per required answer types. The question dependency tree is analyzed and the Lexical Answer Type (LAT) is extracted. The following are some examples of LAT for what questions:

- color: what is Crips' gang color?
- animal: what kind of animal is an agouti?

The extra what-LAT feature is also used with chunking features for what/which questions. For instance, Table 3.8 on page 111 shows a feature when the LAT of a what question is lake.

There is significant prior work in building specialized templates or classifiers for labeling question types (Hermjakob, 2001, Li and Roth, 2002, Zhang and Lee, 2003, Hacioglu and Ward, 2003, Metzler and Croft, 2005, Blunsom et al., 2006, Moschitti et al., 2007). We designed our shallow question type features based on the intuitions of these prior work, with the goal of having a relatively compact approach that still extracts useful predictive signal. One possible drawback, however, is that if an LAT is not observed during training but shows up in testing, the sequence tagger would not know which answer type to associate with the question. In this case it falls back to the more general qword=? feature and will most likely pick the type of answers that are mostly associated with what questions in training.

Edit script Our TED module produces an edit trace for each word in a candidate sentence: the word is either deleted, renamed (if there is a word of the same lemma in the question tree) or strictly aligned (if there is an identical node in the question tree). A word in the deleted edit sequence is a cue that it could be the answer. A word being aligned suggests it is less likely to be an answer. Thus for each word we extract features based on its edit type, shown in Table 3.4.

These features are also appended with the token's POS/NER/DEP information. For instance, a deleted noun usually carries higher edit feature weights than an aligned adjective. We do not intersect edit features with question type features any more as we believe the conjunction between question type features and chunking features is already discriminative against answer type selection.

Alignment distance We observed that a candidate answer often appears close to an aligned word (i.e., answer tokens tend to be located "nearby" portions of text that align across the pair), especially in compound noun constructions, restrictive clauses, preposition phrases, etc. For instance, in the following pair, the answer Limp Bizkit comes from the leading compound noun:

- What is the name of Durst 's group?
- Limp Bizkit lead singer Fred Durst did a lot ...

Past work has designed large numbers of specific templates aimed at these constructions (Soubbotin, 2001, Ravichandran et al., 2003, Clark et al., 2003, Sneiders, 2002b). Here we use a single general feature that we expect to pick up much of this signal, without the significant feature engineering.

Thus we incorporated a simple feature to *roughly* model this phenomenon. It is defined as the distance to the nearest aligned non stopword in the original word order. In the above example, the only aligned non stopword is Durst. Then this nearest alignment distance feature for the word Limp is:

nearest_dist_to_align(Limp):5

This is the only integer-valued feature. All other features are binary-valued. Note this feature does not specify answer types: an adverb close to an aligned word can also be wrongly taken as a strong candidate. Thus we also include a version of the POS/NER/DEP based feature for each token:

- nearest_dist_pos(Limp)=NNP
- nearest_dist_dep(Limp)=NMOD
- nearest_dist_ner(Limp)=B-PERSON

In practice we also tried to use the nearest distance to aligned word in dependency trees instead of in word order. However, this did not work better.

3.3.3. Overproduce-and-vote

In DEV, each question comes with on average 13 candidate answer sentences, only 2.5 of which contain an answer. We make an assumption that each sentence produces a candidate answer and then vote among all answer candidates to select the most-voted as the answer to the original question. An example is shown in Figure 3.4. Specifically, this overproduce-and-vote strategy applies voting in two places:

1. If there are overlaps between two answer candidates, a partial vote is performed. For instance, for a when question, if one answer candidate is April , 1994 and the other is 1994, then besides the base vote of 1, both candidates have an extra partial vote of $^{\#\text{overlap}}/_{\#\text{total words}} = 1/4$. We call this *adjusted vote.*

2. If the CRF fails to find an answer, we still try to "force" an answer out of the tagged sequence, (which contains all O's). thus *forced vote.* Due to its lower credibility (the sequence tagger does not think it is an answer), we manually down-weight the prediction score by a factor of 0.1 (divide by 10).

The modified score for an answer candidate is thus: total vote = adjusted vote + 0.1 × forced vote. To compute forced vote, we make the following observation.

when did amtrak begin operations ?
S1: Amtrak has not turned a profit since it was founded in 1971 [**0.95**].
S2: In 1971 [**0.98**], Amtrak – which combined and streamlined the operations of 18 intercity passenger railroads – went into service .
S3: Amtrak has not made a profit since Congress created it in 1971 [**0.58**] to take over passenger operations of private railroads .
S4: In 1997 [*0.98*], Congress said Amtrak must become self-sufficient by 2002 [*0.98*].
S5: Congress gave Amtrak in 1997 [*0.69*] an infusion of aid along with a 2003 [*0.74*] deadline to become self-sufficient .

Figure 3.4.: A real example of CRF answer tagging. Probabilities for proposed answers are marked in [], with **bold** being correct and *italics* being wrong.

Sometimes the sequence tagger does not tag an answer in a candidate sentence at all, if there is not enough probability mass accumulated for B-ANS. However, a possible answer can still be caught if it has an "outlier" marginal probability. Table 3.5 shows an example. The answer candidate World War II has a much lower marginal probability as an "O" but still not low enough to be part of B-ANS/I-ANS.

To catch such an outlier, we use (MAD), which is the median of the absolute deviation from the median of a data sequence. Given a data sequence $\mathbf{x}$, MAD is defined as:

$$\text{MAD}(\mathbf{x}) = \text{median}(\mid \mathbf{x} - \text{median}(\mathbf{x}) \mid)$$

Compared to mean value and standard deviation, MAD is more robust against the influence of outliers since it does not directly depend on them. We select those words whose marginal probability is 50 times of MAD away from the median of the whole sequence as answer candidates. They contribute to the forced vote. Down-weight ratio (0.1) and MAD ratio (50) were hand-tuned on DEV.[7]

[7]One might further improve this by leveraging the probability of a sentence containing an answer from the QA pair ranker described in Section 3.2 or via the conditional probability of the sequence labels, $p(\mathbf{y} \mid \mathbf{x})$, under the CRF.

Gold	Prediction	Reference Tokens
O	O:0.921060	Conant
O	O:0.991168	had
O	O:0.997307	been
O	O:0.998570	a
O	O:0.998608	photographer
O	O:0.999005	for
O	O:0.877619	Adm
O	O:0.988293	.
O	O:0.874101	Chester
O	O:0.924568	Nimitz
O	O:0.970045	during
B-ANS	O:0.464799	World
I-ANS	O:0.493715	War
I-ANS	O:0.449017	II
O	O:0.915448	.

Table 3.5.: A sample sequence tagging output that fails to predict an answer. The first column is the gold output and the second column is the model output with the marginal probability for predicated labels. Note that World War II has much lower probabilities as an O than others.

3.3.4. Experiments

3.3.4.1. QA Results

The dataset listed in Table 3.2 was not designed to include an answer for each positive answer sentence, but only a binary indicator on whether a sentence contains an answer. We used the answer pattern files (in Perl regular expressions) released along with TREC8-13 to pinpoint the exact answer fragments. Then we manually checked TRAIN, DEV, and TEST for errors. TRAIN-ALL already came as a noisy dataset so we did not manually clean it, also due to its large size.

We trained on only the positive examples of TRAIN and TRAIN-ALL separately with CRFsuite (Okazaki, 2007). The reason for training solely with positive examples is that they only constitute 10% of all training data and if trained on all, the CRF tagger was very biased on negative examples and reluctant to give an answer for most of the questions. The CRF tagger attempted an answer for about

2/3 of all questions when training on just positive examples.

DEV was used to help design features. A practical benefit of our compact approach is that an entire round of feature extraction, training on TRAIN and testing on DEV took less than one minute. Table 3.6 reports F1 scores on both the positive and negative examples of TEST.

Our baseline model, which aligns the question word with some content word in the answer sentence,[8] achieves 31.4% in F1. This model does not require any training. "CRF" only takes votes from those sentences with an identified answer. It has the best precision among all models. "CRF forced" also detects outliers from sentences not tagged with an answer. Large amount of training data, even noisy, is helpful. In general TRAIN-ALL is able to boost the F1 value by $7 \sim 8\%$. Also, the overgenerate-and-vote strategy, used by the "forced" approach, greatly increased recall and achieved the best F1 value.

We also experimented with the two methods utilizing WordNet in Section 3.2.2, i.e., WNsearch and WNfeature. In general, WNsearch helps F1 and yields the best score (63.3%) for this task. For WNfeature[9] we observed that the CRF model converged to a larger objective likelihood with these features. However, it did not make a difference in F1 after overgenerate-and-vote.

Finally, we found it difficult to do a head-to-head comparison with other QA systems on this task.[10] Thus we contribute this enhanced dataset with manual answer labels to the community, hoping to solicit direct comparisons in the future. Also, we believe our chunking and question-type features capture many intuitions most current QA systems rely on, while our novel features are based on TED. We further conduct an ablation test to compare traditional and new QA features.

[8] This only requires minimal modification to the original TED algorithm: the question word is aligned with a certain word in the answer tree instead of being inserted. Then the whole subtree headed by the aligned word counts as the answer.

[9] These are binary features indicating whether an answer candidate has a WordNet relation (c.f. §3.2.2) with the LAT. For instance, tennis is a hyponym of the LAT word sport in the what sport question in Figure 3.1.

[10] Reasons include: most available QA systems either retrieve sentences from the web, have different preprocessing steps, or even include templates learned from our test set.

System	**Train**	**Precision%**	**Recall%**	**F1%**
CRF	TRAIN	55.7	43.8	49.1
	TRAIN-ALL	**67.2**	50.6	57.7
CRF + WNsearch	TRAIN	58.6	46.1	51.6
	TRAIN-ALL	66.7	49.4	56.8
CRF forced	TRAIN	54.5	53.9	54.2
	TRAIN-ALL	60.9	59.6	60.2
CRF forced + WNsearch	TRAIN	55.2	53.9	54.5
	TRAIN-ALL	63.6	**62.9**	**63.3**

Table 3.6.: Performance on TEST. "CRF" only takes votes from candidates tagged by the sequence tagger. "CRF forced" (described in §3.3.3) further collects answer candidates from sentences that CRF does not tag an answer by detecting outliers.

	All	-POS	-NER	-DEP	-LEFT 3	-EDIT	-ALIGN	-LEFT 2
CRF	49.1	44.7	44.0	49.4	19.4	44.3	47.4	40.5
Forced	54.2	48.9	50.8	54.5	25.3	47.5	51.1	42.0

Table 3.7.: QA F_1 based on feature ablation tests.

3.3.4.2. Ablation Test

We did an ablation test for each of the four types of features. Note that the question type features are used in combination with chunking features (e.g., qword=how_much|pos[t]=CD|pos[t+1]=NN), while the chunking feature is defined over POS/NER/DEP separately. We tested the CRF model with deletion of one of the following features each time:

- POS, NER or DEP. These features are all combined with question types.
- The three of the above. Deletion of these features also deletes question type feature implicitly.

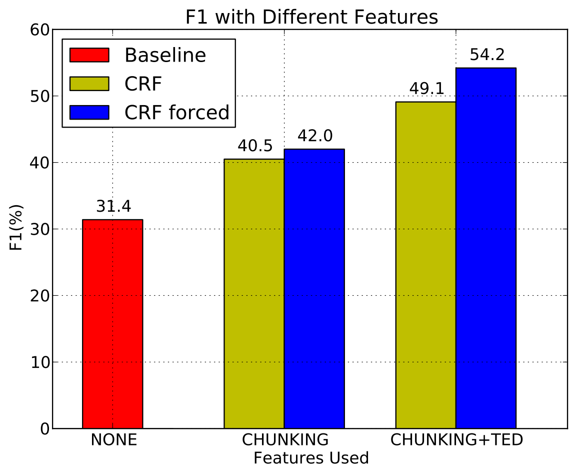


Figure 3.5.: Impact of adding features based on chunking and question-type (CHUNKING) and tree edits (TED), e.g., EDIT and ALIGN.

- EDIT. Features extracted from edit script.
- ALIGN. Alignment distance features.
- The two of the above, based on the TED model.

Table 3.7 shows the F1 scores of ablation test when trained on TRAIN. NER and EDIT are the two single most significant features. NER is important because it closely relates question types with answer entity types (e.g., qword=who |ner[t]=PERSON). EDIT is also important because it captures the syntactic association between question tree and answer tree. Taking out all three POS/NER/DEP features means the chunking and question type features do not fire anymore. This has the biggest impact on F1. Note the feature redundancy here: the question type features are combined with all three POS/NER/DEP features thus taking out a single one does not decrease performance much. However, since TED related features do not combine question type features, taking out all three POS/NER/DEP features decreases F1 by 30%. Without TED related features (both EDIT and ALIGN) F1 also drops more than 10%.

Figure 3.5 is a bar chart showing how much improvement each feature brings. While having a baseline model with 31.4% in F1, traditional features based on

POS/DEP/NER and question types brings a 10% increase with a simple sequence tagging model (second bar labeled “CHUNKING” in the figure). Furthermore, adding TED based features to the model boosted F1 by another 10%.

3.3.5. Summary

Answer extraction is an essential task for any text-based question-answering system to perform. We have cast answer extraction as a sequence tagging problem by deploying a fast and compact CRF model with simple features that capture many of the intuitions in prior “deep pipeline” approaches, such as Harabagiu et al. (2001), Pasca and Harabagiu (2001), Nyberg et al. (2003). We introduced novel features based on TED that boosted F_1 score by 10% compared with the use of more standard features. Besides answer extraction, our modified design of the TED model is the state of the art in the task of ranking QA pairs. Finally, to improve the community’s ability to evaluate QA components without requiring increasingly impractical end-to-end implementations, we have proposed answer extraction as a subtask worth evaluating in its own right, and contributed a dataset that could become a potential standard for this purpose. We believe all these developments will contribute to the continuing improvement of QA systems in the future.

Next we move on to introduce how the IR front end can be easily improved with features learned from the answer extraction back end.

3.4. Structured Information Retrieval for QA

3.4.1. Motivation

A general architecture for a Question Answering system starts with an Information Retrieval module that feeds potentially relevant passages to an answer extraction module that identifies potential answers. The overall performance of such a system is thus

	feature	**label**	**weight**
1	qword=when|POS_0=CD	B-ANS	0.86
2	qword=when|NER_0=DATE	B-ANS	0.79
3	qword=when|POS_0=CD	O	-0.74
4	what-lake|NER_0=LOC	B-ANS	0.12
5	what-lake|POS_0=NNP|POS_1=NNP	B-ANS	0.16
6	what-lake|POS_{-1}=IN|POS_0=NNP	B-ANS	0.15
7	what-lake|POS_0=VBN	O	0.001
8	qword=what|POS_0=NN	B-ANS	0.85
9	qword=what|NER_0=I-MONEY	B-ANS	0.56
10	qword=what|NER_0=B-PRODUCT	B-ANS	0.56
11	qword=what|NER_0=I-LOC	B-ANS	0.53
12	qword=what|POS_0=CC	I-ANS	0.31

Table 3.8.: Learned weights for sampled features with respect to the label of *current* token (indexed by [0]) in a CRF. The larger the weight, the more "important" is this feature to help tag the current token with the corresponding label.

bounded by the IR component, resulting in research specifically on Information Retrieval for Question Answering (IR4QA) in either a monolingual or crosslingual setting (Greenwood, 2008, Sakai et al., 2010). Common approaches to improving IR4QA include query expansion, structured retrieval, and translation models. However, despite their individual success, these approaches either show patterns of complicated engineering on the IR side, or isolate the upstream passage retrieval from downstream answer extraction. We argue that:

1. an IR front end should deliver exactly what a QA[11] back end needs;
2. many intuitions employed by QA should be and can be *re-used* in IR, rather than *re-invented.*

Through this section, we demonstrate the idea that, given a QA system, the IR component does *not* need to do *any extra* analysis on questions, but instead gains all the analytical power *for free* from its downstream QA system. We propose a structured retrieval method with prior knowledge of its downstream QA component, that feeds QA with exactly the information needed for answer extraction.

[11]After this point in § 3.4 we use the term QA in a narrow sense: QA without the IR component, i.e., answer extraction.

As a motivating example, using the question `When was Alaska purchased` from the TREC 2002 QA track as the query to the Indri (Strohman et al., 2005) search engine, the top sentence retrieved from the accompanying AQUAINT corpus is:

`Eventually Alaska Airlines will allow all travelers who have purchased electronic tickets through any means.`

While this relates `Alaska` and `purchased`, it is not a useful passage for the given question. Based on a non-optimized IR configuration, none of the top 1000 returned passages contained the correct answer: `1867`. It is apparent that the question asks for a date; intuitively this motivates an IR component that:

1. Recognizes that the answer type is a date;
2. Copes with variations of date seeking questions, such as `when`, `what year`, `how long ago`, etc.;
3. Ensures that the named entity (NE) tagger really tags dates in the corpus as date entities.

Prior work followed this intuition via predictive annotation (Prager et al., 2000, 2006): there text is first annotated in a predictive manner (of what types of questions it might answer) with approximately 20 answer types and then indexed. A question analysis component (consisting of roughly 400 question templates) maps the desired answer type to one of the 20 existing answer types. Retrieval is then performed with both the question and predicated answer types in the query. This was the first work combining both structured retrieval and answer typing for question answering.

However, predictive annotation has its limitations: it is labor intensive (new answer types need be manually defined for each new question type observed), and it makes certain strong assumptions on the abilities of the underlying NLP pipeline (e.g., it assumes that preprocessing text with named entity tags will result in *accurate* named entity tags).

Our proposed retrieval method instead directly asks the downstream QA system for the information about *which entities answer which questions*, via two steps:

1. reusing the question analysis components from downstream QA;

2. forming a query based on the most relevant answer features given a question from the learned QA model.

There is no query-time overhead and no manual template creation for answer typing. Moreover, this approach is more robust against, e.g., entity recognition errors, because answer typing knowledge is learned from how the data was *actually* labeled, not from how the data was *assumed* to be labeled (e.g., manual templates usually assume perfect labeling of named entities, but often it is not the case in practice).

We use a statistically-trained QA system described in § 3.3 that recognizes the association between question type and expected answer types through various features. The QA system employs a linear chain Conditional Random Field (CRF) (Lafferty et al., 2001) and casts the task of answer extraction as a sequence tagging problem: given a question, it tags each token as to whether it is the Beginning/Inside/Outside (BIO) of an answer. This will be our off-the-shelf QA system, which automatically learns a compact model that captures many template-based intuitions from traditional QA approaches, directly interpreted by its features extracted from, e.g., part-of-speech tagging (POS) and named entity recognition (NER), and learned weights, as shown in Table 3.8. For instance, line 1 says when answering a `when` question, and the POS of current token is CD (cardinal number), it is likely (large weight) that the token is tagged as B-ANS (beginning of the answer). Lines 4-7 list features learned for the question `what is the deepest lake in the US?` in the training set. Line 4: current token with NER=LOC is likely to be B-ANS; Line 5: current token with POS=NNP and followed by a token with POS=NNP is likely to be B-ANS; Line 6: current token with POS=NNP and previous token with POS=IN is likely to be B-ANS; Line 7: current token with POS=VBN is possibly not an answer (positive weight for the label type O).

With weights optimized by CRF training, we can learn how answer features are correlated with question features. These features, whose weights are optimized by the CRF training, directly reflect what the most important answer types associated with each question type are. For instance, line 2 in Table 3.8 says that if there is a `when` question, and the current token's NER label is DATE, then it is likely that this token is tagged as B-ANS. IR can easily make use of this knowledge: whenever there is a `when` question, IR retrieves sentences containing tokens labeled as DATE by NER, or POS tagged as CD. The only extra processing is to pre-tag and index the text with POS and NER labels. The analyzing power of discriminative answer features for IR comes *for free* from a trained QA system. Unlike predictive annotation, statistical evidence determines the best answer features given the question, with no manual pattern or templates needed.

To compare again predictive annotation with our approach: predictive annotation works in a *forward* mode, downstream QA is tailored for upstream IR, i.e., QA works on whatever IR retrieves. Our method works in reverse (*backward*): downstream QA dictates upstream IR, i.e., IR retrieves what QA wants. Moreover, our approach extends easily beyond fixed *answer types* such as named entities: we are already using POS tags as a demonstration. We can potentially use any helpful *answer features* in retrieval. For instance, if the QA system learns that `in order to` is highly correlated with `why` question through lexicalized features, or some certain dependency relations are helpful in answering questions with specific structures, then it is natural and easy for the IR component to incorporate them. Any properties helpful to answer a question can be abstracted as answer features and learned through the QA system.

There is also a distinction between our method and the technique of *learning to rank* applied in QA (Bilotti et al., 2010b, Agarwal et al., 2012). Our method is a *QA-driven* approach that provides supervision for IR from a learned QA model, while learning to rank is essentially an *IR-driven* approach: the supervision for IR comes from a labeled ranking list of retrieval results.

Overall, we make the following contributions:

- Our proposed method tightly integrates QA with IR and the reuse of analysis from QA does not put extra overhead on the IR queries. This QA-driven approach provides a holistic solution to the task of IR4QA.
- We learn statistical evidence about what the form of answers to different questions look like, rather than using manually authored templates. This provides great flexibility in using answer features in IR queries.

retrieval	document	passage	methods employed
unstructured	plain text	plain text	word overlap, TFIDF, Okapi BM25, etc
structured	plain text	structured	dependency path or SRL graph match, etc
	structured	structured	SRL structure match, this work

Table 3.9.: The format of document retrieval and passage retrieval for both unstructured and structured retrieval.

Our systematic experiments give a full spectrum evaluation of all three stages of IR+QA: document retrieval, passage retrieval and answer extraction, to examine thoroughly the effectiveness of the method.[12]

3.4.2. Background

We describe the most relevant research about IR4QA in this section, and divide them into two categories: unstructured retrieval and structured retrieval (see Table 3.9 for better navigation).

We start unstructured retrieval with Tellex et al. (2003), who summarized and quantitatively compared 8 passage retrieval algorithms, from the description of the IR components of the top-performing TREC10 systems, including simpler algorithms based on word overlap (Light et al., 2001), density estimation from number of terms with high *idf* values in a fixed-length window (Clarke et al., 2000), and more sophisticated estimations from summing various matches and mismatches between the query and the passage (Ittycheriah et al., 2001a). More recently, Veeravalli and Varma (2009) built a translation model for each answer type and ranked each retrieved passage with the probability of the given question generated by this passage, outperforming the then state-of-the-art algorithms such as Okapi BM25 (Robertson and Walker, 1999). A side finding from the experiments is that the mature open-source retrieval engine Indri (Strohman et al., 2005) is competitive against other common retrieval weighting algorithms such as TFIDF and KL-divergence. Other work on translation models includes that of Berger and Lafferty

[12]Rarely are all three aspects presented in concert (see §3.4.2).

(1999) and Murdock and Croft (2004).

Structured retrieval can be applied at two stages: document retrieval and/or passage retrieval. In structured document retrieval, text is first annotated (by tagging, parsing, etc), then indexed and retrieved. In structured passage retrieval, conventional document retrieval is first performed on plain text, then passages or sentences in the returned document are annotated and retrieved.

For structured passage retrieval, there are techniques of (fuzzy) dependency path mapping (Cui et al., 2005, Kaisser, 2012), graph matching with Semantic Role Labeling (SRL) (Shen and Lapata, 2007) and answer type checking with POS tagging and chunking (Pinchak et al., 2009b). Document retrieval is still the bottleneck for these approaches since structured analysis is only applied after relevant documents are returned, leading to a larger initial search space.

For both structured document and passage retrieval, it is often easier to simplify the task to structured sentence retrieval: retrieving single sentences or structures directly from the built indices. Bilotti et al. (2007) proposed indexing text with their semantic roles and named entities. Queries also include constraints of semantic roles and named entities for the predicate and its arguments in the question. Improvements in recall of answer-bearing sentences were shown over the bag-of-words baseline. Zhao and Callan (2008) extended this work with approximate matching and smoothing. Lin and Pantel (2001) matched dependency paths (with paraphrase or inference rules) to extract answers from missing arguments of predicates.

Most research uses parsing to assign deep structures. Compared to shallow (POS, NER) structured retrieval, deep structures need more processing power and more smoothing, but might also be more precise. [13]

Most of the above research (except for Shen and Klakow (2006), Kaisser (2012)) only reported IR or QA results, but not both, assuming that improvement in one naturally leads to improvement in the other. Bilotti and Nyberg (2008) challenged this assumption

[13] Ogilvie (2010) showed in chapter 4.3 that keyword and named entities based retrieval actually outperformed SRL-based structured retrieval in MAP for the answer-bearing sentence retrieval task in their setting. In this paper we do not intend to re-invent another parse-based structure matching algorithm, but only use shallow structures to show the idea of coupling QA with IR; in the future this might be extended to incorporate "deeper" structure.

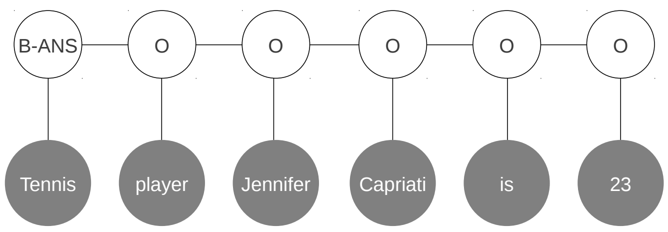


Figure 3.6.: The CRF-based sequence tagging task of answer extraction to the question: `what sport does Capriati play?`. The single-token answer is tagged as B-ANS while other tokens are tagged as O (outside of answer).

and called for tighter coupling between IR and QA. This paper is aimed at that challenge. We further take a step further to make the coupling process fully automatic, as compared to the manual effort from Prager et al. (2006).

3.4.3. Method

We start by recapping the downstream QA system described in § 3.3, which treats answer extraction as a sequence tagging task (re-drawn in Figure 3.6 for convenience): each token in a candidate sentence is tagged as either the beginning (B-ANS), inside (I-ANS), or outside (O) of the answer.[14] The CRF-based sequence tagger employs four kinds of features: chunking (common features used in an NLP chunking task, such as POS tags and NER labels), question-type, edit script (based on tree edit distance) and alignment distance. [15]

Among them, chunking and question-type features are combined to capture the association between answer types and question types. Table 3.8 already shows some examples based on unigram and bigram chunking features. We store the features and their learned weights from the trained model for IR usage.

[14]For QA I-ANS helps characterize words that only appear in multi-word answers, thus BIO labels. For IR the binary labels of ANS and O suffice. In principle we should train QA with two labels for IR usage, but in practice we treated B-ANS and I-ANS as just ANS to keep it simple as we found there was no difference in the learned top answer types when QA was trained with 3 vs. 2 labels.

[15]Edit script and alignment distance features are based on parsed dependency trees. In this paper we do not incorporate these "deeper" features but only focus on *shallow* structured retrieval. Although in theory they can be incorporated in our framework.

We let the trained QA system guide the query formulation when performing structured retrieval with Indri (Strohman et al., 2005), given a corpus already annotated with linguistic structures such as POS tags and NER labels. Then the retrieval process runs in four steps (Figure 3.7):

1. Question Analysis. The question analysis component from QA is reused here. In this implementation, the only information we have chosen to use from the question is the question word (e.g., `how`, `who`) and the lexical answer types (LAT) in case of what/which questions (e.g., `sport` is the LAT of the `what sport` question in Figure 3.6).

2. Answer Feature Selection. Given the question word analyzed from above, we select the 5 most weighted answer features (e.g., POS[0]=CD and NER[0]=DATE for a `when` question). If the LAT word for a `what` question is new and does not appear in the trained model, we fall back to a plain `what` question (features and weights listed in Table 3.8 on page 111).

3. Query Formulation. The original question is combined with the selected answer features as the query.

4. Structured Retrieval. Indri retrieves a ranked list of documents or passages for later evaluation and answer extraction.

As motivated in the introduction, this framework is aimed at providing the following benefits:

1. Reuse of QA components on the IR side. The IR component reuses code for question analysis from QA. The top weighted features are also obtained from the pre-trained QA model. Query formulation is made easier.

2. Various answer features with respect to the question types are statistically determined. For instance, the NER tagger we used divides location into two categories: GPE (countries, cities, states, etc.) and LOC (non-GPE locations, mountain ranges, bodies of water). Both of them are learned to be important to answer `where` questions.

3. Error tolerance along the NLP pipeline. IR and QA share the same processing pipeline. Systematic errors made by the processing tools are tolerated, in the sense that if the same error is made on both the question and sentence, an answer may still be found. Take the `where` question mentioned previously, besides NER[0]=GPE and NER[0]=LOC, we also found oddly NER[0]=PERSON an important feature for it due to that the NER tool sometimes mistakes person names for location names. For instance, the volcano name `Mauna Loa` is labeled as a PERSON instead of a LOC. But since the importance of this feature is recognized by downstream QA to answer `where` questions, the upstream IR is still motivated to retrieve it, per the need of QA.

Queries were lightly optimized using the following strategies:

Query Weighting The original query with the Indri #combine operator implicitly assumes a uniform weight for each term:

#combine(What year was Alaska purchased #max(#any:CD #any:DATE))[16]

In practice we instead use the weighted query:

#weight(1.0 When 1.0 was 1.0 Alaska 1.0 purchased α #max(#any:CD #any:DATE))

with a weight α for the answer types tuned via cross-validation. In general $\alpha < 1.0$, giving the expected answer types "enough say" but not "too much say": as NER and POS tags are not lexicalized they accumulate many more counts (i.e. term frequency) than individual words, thus we down-weight by $\alpha < 1.0$.

NER Types First We found NER labels better indicators of expected answer types than POS tags. The reasons are two-fold: 1. In general POS tags are too coarse-grained in answer types than NER labels. E.g., NNP can answer `who` and `where` questions, but is not as precise as PERSON and GPE. 2. POS tags accumulate even more counts than NER labels, thus they need separate downweighting. *Learning* the interplay of these weights in a joint IR/QA model, is an interesting path for future work.

[16]Question words and common be/do verbs are already in the stopword list of indexing and query so the retrieved sentences do not usually contain the original question words (such as `when` in this case). The #max(.) operator returns the maximum belief of its argument.

If the top-weighted features are based on NER labels, then we do not include POS tags for that question. Otherwise POS tags are useful, for instance, in answering `how` questions.

Unigram QA Model The QA system uses *up to* trigram features (Table 3.8 shows examples of unigram and bigram features). Thus it is able to learn, for instance, that a POS sequence of IN CD NNS is likely an answer to a `when` question (such as: `in 5 years`). This requires that the IR queries look for a consecutive IN CD NNS sequence. We drop this strict constraint (which may need further smoothing) and only use unigram features, not by simply extracting "good" unigram features from the trained model, but by re-training the model with only unigram features. In answer extraction, we still use up to trigram features. [17]

3.4.4. Experiments

We want to measure and compare the performance of the following retrieval techniques:

1. *off-the-shelf retrieval* with an existing IR engine by using the question as query. This is our baseline.
2. *QA-driven shallow structured retrieval* (proposed method).
3. *answer-bearing retrieval* by using both the question and known answer as query (only evaluated for answer extraction). This provides an approximate upper bound.

at the three stages of question answering:

1. Document retrieval (finding relevant documents from the corpus), measured by Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR).

[17]This is because the weights of unigram to trigram features in a loglinear CRF model is a balanced consequence for maximization. A unigram feature might end up with lower weight because another trigram containing this unigram gets a higher weight. Then we would have missed this feature if we only used top unigram features. Thus we re-train the model with only unigram features to make sure weights are "assigned properly" among only unigram features.

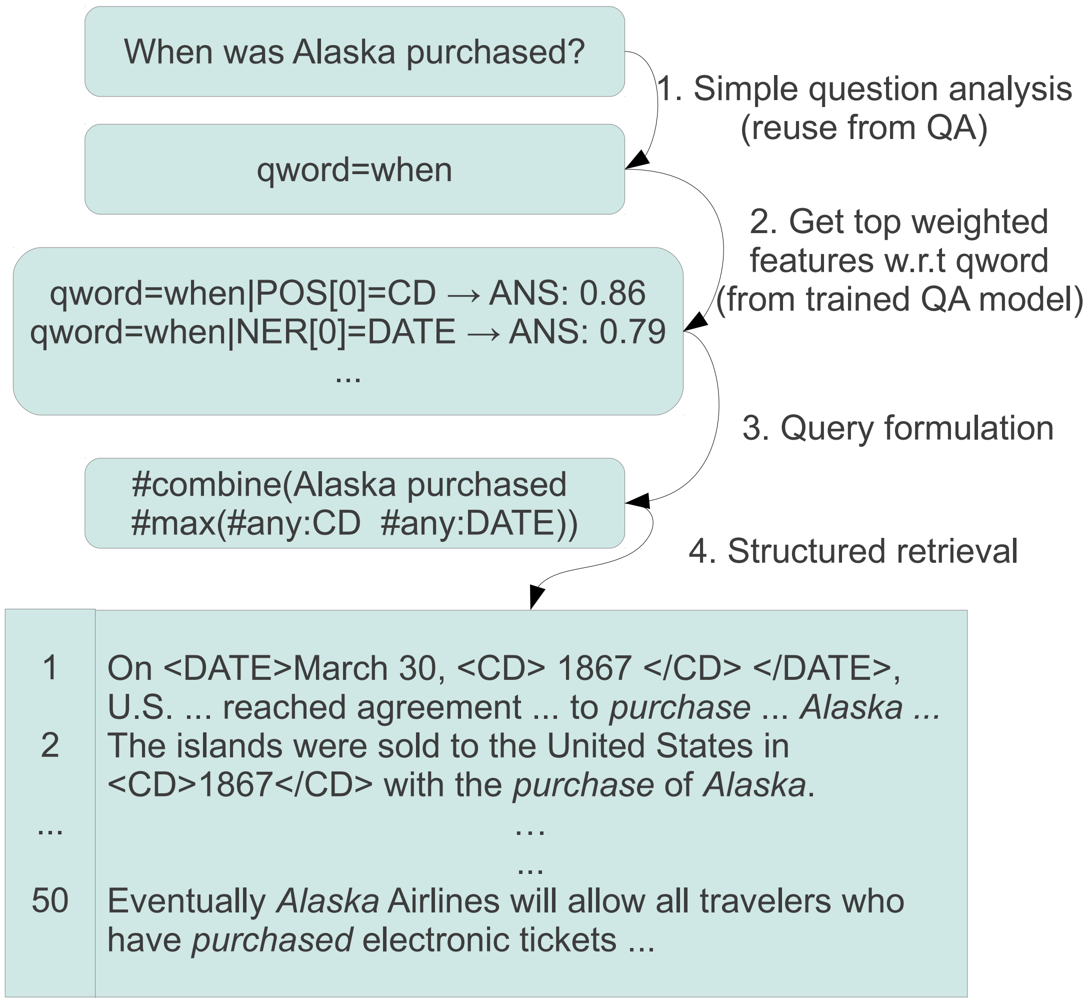

When was Alaska purchased?
1. Simple question analysis (reuse from QA)
qword=when
2. Get top weighted features w.r.t qword (from trained QA model)
qword=when|POS[0]=CD → ANS: 0.86
qword=when|NER[0]=DATE → ANS: 0.79
...
3. Query formulation
#combine(Alaska purchased #max(#any:CD #any:DATE))
4. Structured retrieval
1 On <DATE>March 30, <CD> 1867 </CD> </DATE>, U.S. ... reached agreement ... to purchase ... Alaska ...
2 The islands were sold to the United States in <CD>1867</CD> with the purchase of Alaska.
...
50 Eventually Alaska Airlines will allow all travelers who have purchased electronic tickets ...

Figure 3.7.: Structured retrieval with queries directly constructed from learned most weighted features of downstream QA. The retrieved and ranked list of sentences is POS and NER tagged, but only query-relevant tags are shown due to space limit. A bag-of-words retrieval approach would have the sentence shown above at rank 50 at its top position instead.

set	**questions**		**sentences**	
	#all	**#positive**	**#all**	**#positive**
TRAIN	2205	1756 (80%)	22043	7637 (35%)
TEST$_{\text{gold}}$	99	88 (89%)	990	368 (37%)

Table 3.10.: Statistics for AMT-collected data (total cost was around $800 for paying three Turkers per sentence). Positive questions are those with an answer found. Positive sentences are those bearing an answer.

2. Passage retrieval (finding relevant sentences from the document), also by MAP and MRR.

3. Answer extraction, measured by F_1.

All structured and unstructured queries are performed with Indri v5.3 (Strohman et al., 2005). Indexing and retrieval parameters were left at the default settings for all experiments. We chose to use Indri as our off-the-shelf retrieval baseline based on the following reasons:

1. Turtle et al. (2012) found Indri to be significantly better than the other popular IR engine, Apache Lucene, for short queries, and quantitatively better in long queries.

2. Indri performed better than the use of standard weighting schemes such as TFIDF, Okapi BM25 and KL-divergence (Veeravalli and Varma, 2009).

3. It is commonly used to provide a bag-of-words baseline in other research papers (Bilotti et al., 2007, Zhao and Callan, 2008, Ogilvie, 2010).

Section 2.3.1 on page 46 also provides a more broad comparison of information retrieval tools.

3.4.4.1. Data

Test Set for IR and QA The MIT109 test collection by Lin and Katz (2006) contains 109 questions from TREC 2002 and provides a near-exhaustive judgment of

```
<DOC>
<DOCNO> JMM19990101.0001 </DOCNO>
<TEXT>
<S> <NER−PERSON><POS−NNP>John</POS−NNP></NER−PERSON> <POS−VBZ>loves</POS−
    VBZ> <NER−PERSON><POS−NNP>Mary</POS−NNP></NER−PERSON> . </S>
<S> <POS−PRP>They</POS−PRP> <POS−VBD>married</POS−VBD> <POS−IN>in</POS−IN>
    <NER−DATE><POS−CD>1998</POS−CD></NER−DATE> . </S>
</TEXT>
</DOC>
```

Figure 3.8.: A short artificial example of annotated AQUAINT. Punctuation symbols are not annotated since Indri does not index them.

relevant documents for each question (17 docs on average with a median of 6). Thus it is used as a gold-standard test set for IR. Following Ogilvie (2010), we removed 10 questions that do not have an answer by matching the TREC answer patterns on the relevant documents indicated by MIT109. Then we call this test set **MIT99**.

Training Set for QA We used Amazon Mechanical Turk to collect training data for the statistical QA system. The objective was to gather as many positive (answer-bearing) sentences as possible. Thus we issued answer-bearing queries for TREC1999-2003 questions.[18] We selected the top 10 retrieved sentences for each question, and asked three Turkers to judge whether each sentence contained the answer (also aiding them by showing a potential answer along with the question) and used the majority vote for each sentence. The inter-coder agreement rate was 0.81 (Krippendorff, 2004, Artstein and Poesio, 2008). For each positive sentence, we matched it with the TREC answer patterns to pinpoint the span of the answer and manually resolved multiple matching cases.

The 99 questions of MIT99 were extracted from the Turk collection as our $\text{TEST}_{\text{gold}}$ with the remaining as TRAIN. QA performance on $\text{TEST}_{\text{gold}}$ would indicate an upper bound since the retrieval was answer-aware. The statistics are shown in Table 3.10. Note that only 88 questions out of MIT99 have an answer from the top 10 answer-bearing query results.

Finally both the training and test data were sentence-segmented and word-tokenized

[18]Starting from 2004, TREC questions were grouped by topics. Most questions asked on a topic use anaphora. Thus we did not use these questions as anaphora resolution is not a focus of this work.

Q-type	number	structured		unstructured	
		MAP	MRR	MAP	MRR
how	9	0.1755	0.4377	0.1668	0.3920
what	54	0.2716	0.4596	0.2107	0.3937
when	13	0.2584	0.4596	0.2144	0.3838
where	6	0.3429	0.4610	0.2977	0.5584
which	3	0.2162	0.3636	0.2178	0.3750
who	14	0.1908	0.5835	0.1987	0.5930
ALL	99	**0.2524**	**0.4835**	0.2110	0.4298

Table 3.11.: Structured vs. unstructured document retrieval in MAP and MRR on MIT99. Significance level (Smucker et al., 2007) in the last row: $p < 0.001$ for MAP and $p < 0.01$ for MRR.

Q-type	number	structured		unstructured	
		MAP	MRR	MAP	MRR
how	9	0.0580	0.1273	0.0577	0.1267
what	54	0.1279	0.3026	0.1051	0.2591
when	13	0.0895	0.2134	0.0687	0.1289
where	6	0.2174	0.3123	0.1817	0.1695
which	3	0.0706	0.0411	0.0706	0.0382
who	14	0.2411	0.5151	0.2410	0.5151
ALL	99	**0.1375**	**0.2987**	0.1200	0.2544

Table 3.12.: Structured vs. unstructured sentence retrieval in MAP and MRR on MIT99. Significance level (Smucker et al., 2007) in the last row: $p < 0.001$ for MAP and $p < 0.05$ for MRR.

by NLTK (Bird and Loper, 2004), dependency-parsed by the Stanford Parser (Klein and Manning, 2003), and NER-tagged by Illinois Named Entity Tagger (Ratinov and Roth, 2009) with an 18-label type set. We report F_1 on $\textsc{test}_{\text{gold}}$ in §3.4.4.4.

Corpus Preprocessing for IR We processed the AQUAINT (LDC2002T31) corpus, on which the MIT99 questions are based, in exactly the same manner as we processed the QA training set. But only sentence boundaries, POS tags and NER labels were kept as the annotation of the corpus. An artificial sample is shown in Figure 3.8.

top		**1**	**5**	**10**	**50**	**100**	**200**	**500**	**1000**
#question	**S**	26	44	50	76	82	87	92	94
	U	23	40	45	68	78	83	90	91
#sentence	**S**	26	67	101	308	460	642	984	1176
	U	23	59	92	288	425	620	941	1075

Table 3.13.: Recall numbers of structured (S) and unstructured (U) retrieval for up to top $K = 1000$ retrieved sentences for each question. #question is the number of questions with an answer found while #sentence is the total number of positive sentences for all questions (thus the number of all retrieved sentences should be 99×1000) in MIT99.

3.4.4.2. Document Retrieval

We issued off-the-shelf unstructured queries consisting of only words from the question, and QA-driven structured queries consisting of both the question and expected answer types, then retrieved the top 1000 documents, and finally computed MAP and MRR against the gold-standard MIT99 per-document judgment.

To find the best weighting α for structured retrieval, we tried different values using 5-fold cross-validation on MIT99 and finalized at $\alpha = 0.1$. Table 3.11 shows the overall and per-question-type evaluation. Overall, structured retrieval outperforms (20% by MAP with $p < 0.001$ and 12% by MRR with $p < 0.01$) unstructured retrieval significantly according to paired randomization test (Smucker et al., 2007). The largest gain comes from `what` questions, which consist more than half of the whole test set and are considered hard to answer due to its various lexical answer types (such as `what lake`, `what sport`, etc.).

3.4.4.3. Passage Retrieval

Passage retrieval is the process of extracting relevant sentences from documents. We only focus on a ranked list of single sentences since the downstream QA system does not take passages as input. Recall that MIT99 only contains document-level judgment. To generate a test set for sentence retrieval, we matched each sentence from relevant documents provided by MIT99 for each question against the TREC answer patterns.

Bilotti (2009) and Ogilvie (2010) in their PhD thesis also used MIT109 to evaluate sentence retrieval but manually checked the relevance of each sentence. Unfortunately the manual judgment was lost so we could not do a direct comparison with their work. We sampled 100 of the automatically matched sentences, manually checked them and found the error rate was about 20%.

There is only one difference in query formulation for document retrieval and sentence retrieval. In sentence retrieval the query evaluates for each extent associated with the sentence field: #weight[s](...) while in document retrieval the query does not specify the sentence field: #weight(...). Thus sentence retrieval can be performed as either the stage following document retrieval, or as a single retrieval stage.

We experimented with these two settings: 1. sentences were retrieved from the documents returned by document retrieval; 2. sentences were directly retrieved from the corpus (no document retrieval). No significant difference was found in the performance between these two settings. We only report the numbers of setting 2, shown in Table 3.12, since in practice it is much easier to perform.

Sentence retrieval is essentially a harder task as it requires precise sentence-level judgment rather than more coarse-grained document-level judgment, thus lower MAP and MRR values in general. Still, structured retrieval outperforms unstructured retrieval significantly by about 10% in MAP and 17% in MRR. We were curious to see how these measures reflect quantitatively in the number of questions with an answer found (#question) and the total number of answer-bearing sentences (#sentence) for the 99 test questions. Table 3.13 shows the difference. In general, structured retrieval returns more answer-bearing sentences in the top K returned results, thus better MAP and MRR.

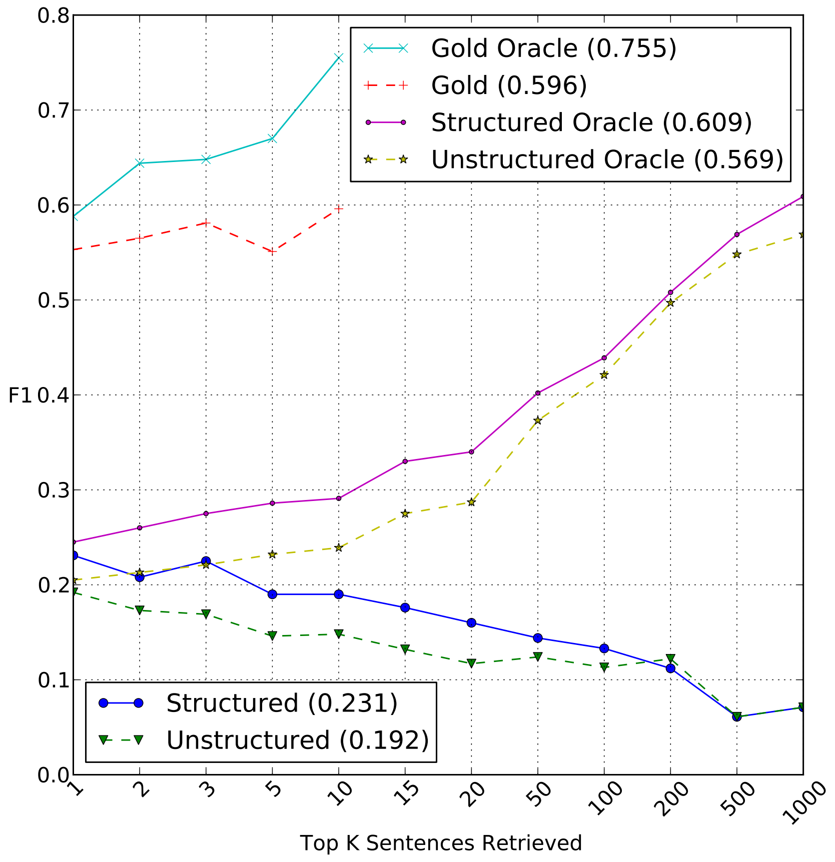


Figure 3.9.: F_1 values for answer extraction on MIT99. Best F_1's for each method are parenthesized in the legend. "Oracle" methods assumed perfect voting of answer candidates (a question is answered correctly if the system ever produced one correct answer for it). "Gold" was tested on TEST$_{\text{gold}}$, which was collected by issuing answer-bearing queries and used Turkers to judge the relevance of only the top 10 retrieved sentences. The figure was generated with `matplotlib` (Hunter, 2007).

3.4.4.4. Answer Extraction

Lastly we sent the retrieved sentences to the downstream QA engine (trained on TRAIN) and computed F_1 per K for the top K retrieved sentences for MIT99,[19] shown in Figure 3.9. The best F_1 with structured sentence retrieval is 0.231, 20% better than F_1 of 0.192 with unstructured retrieval, both at $K = 1$.

The two descending lines (blue and green) at the bottom reflect the fact that the majority-voting mechanism employed by the QA system was too simple to select the best candidate: F_1 drops as K increases. Thus we also computed F_1's assuming perfect voting: a voting oracle that always selects the correct answer as long as the QA system produces one, thus the two ascending lines in the center of Figure 3.9. Still, F_1 with structured retrieval is always better than F_1 with unstructured retrieval: reiterating the fact that structured sentence retrieval covers more answer-bearing sentences.

Finally, to find the upper bound for QA, we drew the two upper lines, testing on $\text{TEST}_{\text{gold}}$ described in Table 3.10. The test sentences were obtained with answer-bearing queries. This is assuming almost perfect IR. The gap between the top two and other lines signals more room for improvements for IR in terms of better coverage and better rank for answer-bearing sentences.

3.4.5. Summary

We described a method to perform structured information retrieval with a prior knowledge of the downstream QA system. Specifically, we coupled structured IR queries with automatically learned answer features from a statistically trained QA system and observed significant improvements in document retrieval, passage retrieval, and boosted F_1 in answer extraction. This method has the merits of not requiring hand-built question and answer templates. Also, it is flexible in incorporating various answer features automatically learned and optimized from the downstream QA system.

[19]Lin (2007), Zhang et al. (2007), and Kaisser (2012) also evaluated on MIT109. However their QA engines used web-based search engines, thus leading to results that are neither reproducible nor directly comparable with ours.

3.5. Conclusion

This chapter is based on the following two published papers:

> Xuchen Yao, Benjamin Van Durme, Peter Clark and Chris Callison-Burch. *Answer Extraction as Sequence Tagging with Tree Edit Distance.* NAACL. 2013.
>
> Xuchen Yao, Benjamin Van Durme and Peter Clark. *Automatic Coupling of Answer Extraction and Information Retrieval.* ACL Short. 2013.

The main ideas and scientific contributions are:

The first to cast answer extraction as a sequence tagging problem. A linear-chain Conditional Random Field (Lafferty et al., 2001) was employed to tag answer candidate with traditional BIO labels (B for B-ANS, beginning of answer; I for I-ANS, inside of answer; O for outside of answer, i.e., not an answer). An example is shown in Figure 3.10:

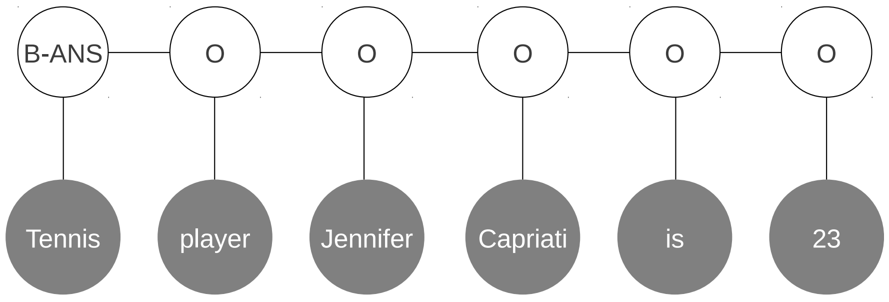


Figure 3.10.: An example of linear-chain CRF for answer sequence tagging in response to the question "`Who was President Cleveland's wife?`". The answer tokens "`Frances Folsom`" are labeled as B-ANS/I-ANS (beginning/inside of answer) and other tokens are labeled as O (outside of answer).

The CRF has the following merits when applied in answer extraction:

1. Its Markovian nature models implicit answer *phrase* patterns, such as the POS transition in the answer phrase Frances/NNP Folsom/NNP, or in/IN 1886/CD if the question asks for a time.

2. Its global decoding nature goes beyond local context and decides the best answer candidate from the sentence, fitting the assumption that usually the answer phrase only appears at most once in the sentence.

3. With the help of efficient toolkit, large-scale discriminative training has become possible when given tens of thousands of QA pairs (further illustrated in § 4.6 on page 180).

The CRF is able to use features extracted from both the sentence and the question and learn those highly associated answer patterns for each question type. This highlights the second idea:

Automatically mining rich features from QA pairs. The objective is to replace traditional manually-written question templates and answer templates with linguistic features, which are learned from the CRF. These linguistic features are based on POS tags, named entity types (NER) and dependency relations (DEP). They are then intersected with question types to learn various answer patterns. For instance, compare the following three features and their learned weights for predicting "yes" on the answer candidate from a maximized objective function in Table 3.14:

	feature	**weight**
1	q=who|NER_0=PER|NER_1=PER	0.5
2	q=when|POS_{-1}=IN|POS_0=CD	0.6
3	q=when|NER_0=PER|NER_1=PER	-0.4

Table 3.14.: Learned feature weights from CRF.

A literal translation of the feature in line 1 reads: if the question is a `who` question, and the current token's entity type is a person (NER_0=PER), followed by another person name (NER_1=PER), then there is a positive feature weight (0.5) learned for this token. This feature actually fires on the token "`Frances`" in the example of Figure 3.10, since "`Frances`" as a first name would be tagged as a person name and the token "`Folsom`" following it is also a person name.

Similarly, in line 2, the CRF learned a good positive weight (0.6) for a feature extracted on the current token, describing the case that there is a `when` question (`q=when`), and the current token's POS tag is a cardinal number (POS_0=CD), preceded by a preposition (POS_{-1}=IN). This feature would fire on the token "`1886`" from the string "`in 1886`" if the question was a `when` question.

On the contrary in line 3, when the question asks for a time (`q=when`) but the token and its context is about a person (NER_0=PER|NER_1=PER), then a negative feature weight is learned. For instance, in Figure 3.10, if the question was `when did President Cleveland marry?`, then the feature would fire on the token `Frances`, discouraging it to be an answer.

The features learned from Table 3.14 are inspiring: it tells us exactly what linguistic annotations are representative for what kind of questions. Thus the third contribution:

The first to *automatically* couple the answer extraction back end with the IR front end using shallow structured information retrieval. QA performance is bounded by its IR front end. However, the IR technique has almost always focused on optimizing quality over keyword search, and ignored the specific needs from the downstream answer extraction component. As a motivating example, when using the question `What year was Alaska purchased?` from TREC2002 QA as the query to the Indri search engine, the top sentence retrieved from the accompanying TREC QA corpus AQUAINT is:

`Eventually Alaska Airlines will allow all travelers who have purchased electronic tickets through any means.`

As a matter of fact, none of the top 1000 returned passages contains the correct answer: `1867`. It is apparent that the question asks for a date. But the IR engine did not take that into account. Luckily, we have already known from row 2 of Table 3.14 that the

POS pattern POS_{-1}=IN|POS_0=CD has a very high weight with when questions. Thus we can naturally index each token in the whole corpus with their corresponding POS tags and named entity labels, then perform structured retrieval to explicitly fetch a number in the results. The general work flow is shown in Figure on page 121.

The whole structured IR for QA setup has the following merits:

- No extra tuning on the IR side. IR maximally reuses techniques from QA. All answer patterns are automatically learned from answer extraction and passed up to IR to retrieve exactly the same pattern from the annotated corpus. So there is also no manual template or tuning.
- No extra query-time overhead on the IR side. There is no more query time overhead than plain text search given that all annotations are preprocessed and indexed.
- IR serves exactly what QA needs. This is the key idea in the work: IR should really search for the *missing* information (represented by structured linguistic annotations) to serve the answer extraction component, rather than searching solely for *existing* information presented in the query.

Conclusion and future work We kick-started a framework for automatic answer extraction and information retrieval. The central idea is that signals that are highly associated with question types and answer patterns can be automatically and effectively learned from data. This overcomes the traditional shortcomings of manually writing templates to match answers and improves both the QA front end (i.e., IR) and back end (i.e., answer extraction).

Several ideas explored in this chapter are worth further investigation. They have also informed subsequent thesis research, specifically:

- *monolingual alignment*: the Tree Edit Distance aligner has brought extra linguistic signals to answer extraction. Ablation test (Table 3.7 and Figure 3.5 on page 108) also shows its effectiveness. But the TED aligner is only rule-based and has very limited potential to be expanded to incorporate more lexical resources for the purposes of recognizing paraphrases/entailment in the QA pairs. Can we do better?

The next chapter proposes a statistical aligner that does better both in terms of alignment accuracy and end QA performance.

- *large-scale discriminative training*: the sequence tagging CRF model is a powerful tool. But how does it scale up, especially when we have tens of thousands questions for training? Section 4.6 on page 180 demonstrates the scale, the speed, and the accuracy when training the CRF learner with about ten thousand Jeopardy! questions. It also compares systematically several off-the-shelf bilingual and monolingual aligners in the task of question answering.

- *feature-driven question answering over structured data*: so far we have shown that the feature-driven idea works on unstructured text. How about structured prediction over structured data? Chapter 5 on page 198 illustrates the same idea and shows that it even performs better than those logic form based statistical parsing approaches. Stay tuned!

4. Discriminative Models for Monolingual Alignment

In the previous chapter I have demonstrated the idea of feature-driven question answering and the usage of a TED (Tree Edit Distance) based aligner to capture the association between the question and the text. To reiterate why alignment is important in question answering (and that it deserves a separate chapter), I use the following example as a motivation:

- Question: Who was **President Cleveland**'s wife?
- Sentence 1: **Cleveland** married Frances [PERSON] Folsom [PERSON] in 1886.
- Sentence 2: **President** Obama married Michelle [PERSON] Obama [PERSON] in 1992.

A naive keyword-based information retrieval engine would rank both sentences equally since each one has exactly one word overlap with the question. Then the answer extraction component has to rule out the second sentence. Table 3.14 on page 130 lists learned features based on the named entity type:

	feature	**weight**
1	q=who|NER_0=PER|NER_1=PER	0.5

This feature would equally fire on both Frances [PERSON] Folsom [PERSON] in sentence 1 and Michelle [PERSON] Obama [PERSON] in sentence 2. It is not discriminative enough and alignment comes to help. As shown in Figure 4.1 on page 136, a good aligner should be able to draw the connection between Cleveland↔Cleveland and married↔wife in the question and sentence 1. Since in the question both the subject (wife) and its modifier (Cleveland) are aligned to sentence 1, this should serve as a strong cue that sentence 1 contains the answer. However, sentence 2 is much less relevant in this regard.

Previously I have demonstrated the usage of the TED aligner to fulfill this purpose. There are a few disadvantages of this aligner:

- The principle of this aligner is minimum tree edit distance, which is not necessarily how sentences should be aligned. To put it in other words, there is no ground to support the theory that minimum tree edit distance yields the best alignment between two sentences.

- The aligner is rule-based; it is not statistically trained and cannot easily incorporate more lexical resources to align paraphrases or semantically related words, which are both challenges in the task of monolingual alignment.

Thus we devise a new tool for better alignment between QA pairs. This chapter presents two statistically trained monolingual aligners for the English language: one based on a Markov model for token alignment (§ 4.3.1) and the other based on a semi-Markov model for phrase alignment (§ 4.3.2). The aligners are able to make use of various lexical resources and have the ability to recognize paraphrase and entailment presented in sentence pairs. We show intrinsic evaluation in terms of alignment quality in § 4.4 and summaries the challenges and future directions in § 4.5. The accompanying implementation is jacana-align, released for general purpose monolingual alignment for the English language. Finally, systematic evaluation of several bilingual and monolingual aligners in the task of question answering is conducted (§ 4.6), with the conclusion that statistical monolingual aligners show consistent superior (up to 25% relative F_1) performance in end QA performance. Thus the task of monolingual alignment in question answering is justified.

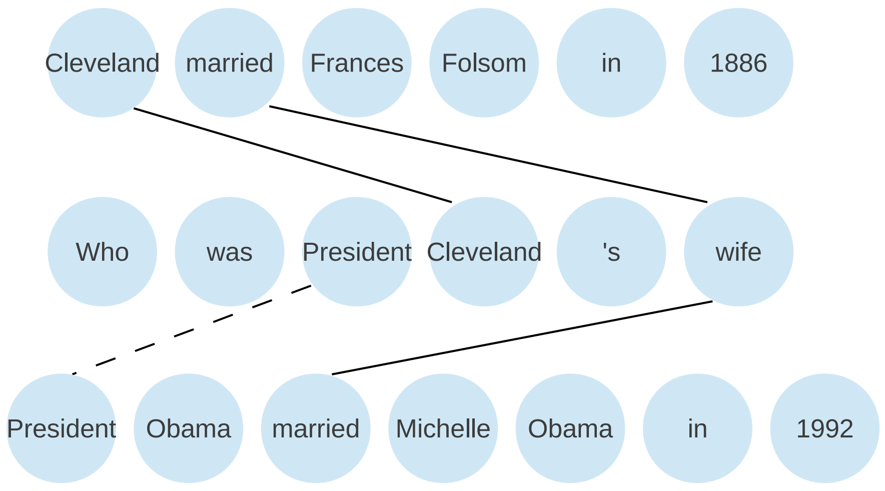


Figure 4.1.: Alignment between the question (middle) and two candidate sentences. The fact that all key concepts in the question are aligned to some words in the top sentence is a good cue to rank the top sentence higher. An intelligent aligner should also decide the two President's should not align (dashed line).

4.1. Introduction

Various NLP tasks can be treated as an alignment problem: machine translation (aligning words in one language with words in another language), question answering (aligning question words with the answer phrase), textual entailment recognition (aligning premise with hypothesis), paraphrase detection (aligning semantically equivalent words), etc. Even though most of these tasks involve only a single language, alignment research has primarily focused on the bilingual setting (i.e., machine translation) rather than monolingual.

Alignment happens at different stages in these two settings. In machine translation, bilingual alignment is majorly used during training to learn the phrase translation table for a language pair: both the source and the target sides are seen; during test only the source language side is given, thus the phrase table generates translation candidates for the target language. In monolingual tasks, alignment often also happens at test time: both the source sentence and the target sentence are presented and need to be aligned. The generation nature of machine translation requires that alignment memorizes lots of translation pairs from a large amount of parallel data. It is an unsupervised task and often the EM algorithm is used for learning the model parameters.

Monolingual alignment on the other hand does not have as much unlabeled parallel data: but a small amount (for the English language) has been annotated for supervised learning. Analysis on these annotated datasets also shows one important characteristic in monolingual alignment: identical word alignment is most prominent type of alignment (contributing 75% to 78% of alignment). In this case, a single feature based on word identity would construct a reasonable baseline. This simple feature is also effective in aligning unseen identical words during *test* time, a phenomenon that bilingual alignment methods based on the IBM models (Brown et al., 1993) are hard to tackle since they are not designed to align unseen data during test. On nonidentical alignments, monolingual alignment can directly benefit from various types of lexical resources, such as WordNet (Fellbaum, 1998) and PPDB (Ganitkevitch et al., 2013), and output from several runs of the Semantic Textual Similarity task (Agirre et al., 2012). Thus discriminative models utilizing features utilizing both word similarity and external resources can be the most helpful.

In this chapter we present two models for the task of monolingual alignment. The first one is a token aligner that focuses on one-to-one alignment. The second one is a phrase aligner that is capable of many-to-many phrase-based alignment.[1] Our work is heavily influenced by the bilingual alignment literature, especially the discriminative model proposed by Blunsom and Cohn (2006). Specifically, we used a Conditional Random Field (Lafferty et al., 2001) model to align from the source sentence to the target sentence. At each source token the CRF inspects the whole target sentence and extracts features for each pair of target word and the current source word. The states of the CRF are target word indices. Unlike CRFs with a fixed state vocabulary, such as the ones for POS tagging (states are for instance 45 Penn Treebank tags) or Named Entity Recognition (states are for instance Beginning/Inside/Outside representation of PERSON, ORG and LOC), the alignment CRF has a dynamic state set depending solely on the length of target sentence. Thus observation features and state transition features must be designed not to be associated with a particular state, but in general any possible state to avoid

[1]In this chapter we use the term *token-based alignment* for one-to-one alignment and *phrase-based* for one-to-many or many-to-many alignment, and *word alignment* in general for both.

data sparsity problems.

Most token-based alignment models can extrinsically handle phrase-based alignment to some extent. For instance, in the case of `NYC` aligning to `New York City`, the single source word `NYC` may align three times separately to the target words: `NYC`↔`New`, `NYC`↔`York`, `NYC`↔`City`. Or in the case of identical alignment, `New York City` aligning to `New York City` is simply `New`↔`New`, `York`↔`York`, `City`↔`City`. However, it is not as clear how to token-align `New York` (as a city) with `New York City`. The problem is more prominent when aligning phrasal paraphrases or multiword expressions, such as `pass away` and `kick the bucket`. This suggests an intrinsically phrase-based alignment model.

Our CRF-based token aligner aligns tokens from the source sentence to tokens in the target sentence by treating source tokens as "observation" and target tokens as "hidden states". However, it is not designed to handle phrase-based alignment, largely due to the *Markov nature* of the underlying model: a state can only span one token each time, making it unable to align multiple consecutive tokens (i.e. a phrase). We extend this model by introducing *semi-Markov states* for phrase-based alignment: a state can instead span multiple consecutive time steps, thus aligning phrases on the source side. Also, we merge phrases on the target side to *phrasal states*, allowing the model to align phrases on the target side as well. To spell out the distinction between Markov and semi-Markov:

- the Markov model has the memory less property: the current state only depends on the previous state *one* time step ago.
- the semi-Markov property introduces states that do not have the Markov property: some "long lasting" state can span *multiple* time steps (say, l). Thus the transition probability of these states depends on the previous state l time steps ago instead of one time step ago. These states do not have the Markov property while other "normal" states do. Thus the whole mixed model is a semi-Markov model.

The token and phrase aligners give state-of-the-art performance on the MSR06 (Brockett, 2007) and Edinburgh++ (Cohn et al., 2008, Thadani et al., 2012) alignment datasets. We also introduce a third dataset, MTReference, with more phrase alignment and report results divided by alignment types.

4.2. Related Work

4.2.1. Bilingual Alignment

Word alignment was first explored in machine translation. The IBM models (Brown et al., 1993) for machine translation can also be used in word alignment. Specifically, it includes 5 models refined by different complexities:

1. IBM Model 1 estimates solely lexical translation probability distributions;
2. IBM Model 2 adds absolute alignment probability based on the word positions of source and target words;
3. IBM Model 3 adds fertility to model how many target words are generated for each source word and distortion to predict the target word positions based on source word positions;
4. IBM Model 4 improves the absolute distortion in Model 3 with relative distortion;
5. IBM Model 5 generates words only into vacant positions and thus eliminates deficiency, a situation where generated words could be put in the same target position according to previous models.

Parameters are estimated using the EM algorithm in an unsupervised fashion. The IBM models allow many-to-one alignment but they are asymmetric, meaning that they only allow one-to-many alignments from French-to-English but not from English-to-French. Phrase-based MT introduced heuristics (Koehn, 2010) to merge two sets of word alignment created in opposite directions for the purposes overcoming this asymmetry. Common heuristics include INTERSECTION, UNION, and GROW-DIAG-FINAL. Liang et al. (2006) proposed alignment by agreement to encourage agreement from both directions during training.

Later, researchers explored non-heuristic phrase-based methods. Among them, Marcu and Wong (2002) described a joint probability model that generates both the source and target sentences simultaneously. All possible pairs of phrases in both sentences are enumerated and then pruned with statistical evidence. Their proposed method was computationally expensive and worked better for short sentences. Deng and Byrne (2008) explored token-to-phrase alignment based on Hidden Markov Models (Vogel et al., 1996) by explicitly modeling the token-to-phrase probability and phrase lengths. However, the token-to-phrase alignment is only in one direction: each target state still only spans one source word, and thus alignment on the source side is limited to tokens. Andrés-Ferrer and Juan (2009) extended the HMM-based method to Hidden Semi-Markov Models (HSMM) (Ostendorf et al., 1996), allowing phrasal alignments on the source side. Finally, Bansal et al. (2011) unified the HSMM models with the alignment by agreement framework (Liang et al., 2006), achieving phrasal alignment that agreed in both directions.

Even though semi-Markov models have been successfully applied in bilingual alignment, they have not been used in discriminative monolingual alignment. Essentially monolingual alignment would benefit more from discriminative models with various feature extractions than generative models without any predefined features. To combine the strengths of both semi-Markov models and discriminative training, we propose to use the semi-Markov Conditional Random Field (Sarawagi and Cohen, 2004), which was first used in information extraction to tag continuous segments of input sequences and outperformed conventional CRFs in the task of named entity recognition. We describe this model in Section 4.3.

4.2.1.1. Evaluation

MT research assumes that annotators make two kinds of alignment: S (sure) alignment and P (possible) alignment (or ambiguous alignment). P alignments are created when multiple annotators label the data and choose different points. P is the label assigned to alignment points created by one of the annotators but not all of them. Sometimes annotation guidelines allow individual annotators to mark alignment points as P when they are approximate alignments. In most literature P alignment is *only* about ambiguous alignment, or the alignment not agreed by annotators. But in evaluation, it is assumed that P is a superset of S ($S \subseteq P$). Suppose A is the prediction, then alignment quality is defined as precision, recall and Alignment Error Rate (AER) (Och and Ney, 2003):

$$\text{precision} = \frac{\mid A \cap P \mid}{\mid A \mid}$$

$$\text{recall} = \frac{\mid A \cap S \mid}{\mid S \mid}$$

$$\text{AER} = 1 - \frac{\mid A \cap S \mid + \mid A \cap P \mid}{\mid A \mid + \mid S \mid}$$

The definition is based on the assumption that a recall error happens when an S alignment is not found and a precision error happens when a found alignment even is not in P.

4.2.1.2. Phrase Extraction

To have better coverage, machine translation models try to extract as many phrase pairs from word alignment as possible. The phrase extraction algorithm (§5.2 in Koehn (2010)) extracts and combines all *consistent* phrase pairs with permission to include unaligned words in the phrase. A *consistent phrase alignment* can be roughly described as: given a phrase pair, for every word on one side, if it is aligned, then all its aligned words on the other side have also to be in the phrase pair; and vice versa. The following are three examples of consistent alignment for `New York`↔`New York (City)`:

1.

	New	York
New	●	
York		●

2.

	New	York
New	●	●
York	●	●

3.

	New	York	City
New	●		
York		●	

Number 2 is defined as a *block alignment*: every word on one side is aligned with every word on the other side, and vice versa. Figure 4.1 gives an illustrative example of extracted phrases (b) given an alignment matrix between an English-German sentence pair (a). Note that the extracted phrases are mostly not natural phrases, for instance, michael assumes, assumes that he, in the. This serves phrase-based machine translation well, but it is not exactly how humans would annotate phrase alignment. In the next section we describe the simple way of using block alignment to extract phrases in monolingual alignment.

4.2.2. Monolingual Alignment

Most work in monolingual alignment employs dependency tree/graph matching algorithms, including:

- Tree Edit Distance (Punyakanok et al., 2004, Kouylekov and Magnini, 2005, Heilman and Smith, 2010, Yao et al., 2013d);
- Particle Swarm Optimization (Mehdad, 2009);
- linear regression/classification models (Chambers et al., 2007, Wang and Manning, 2010);
- min-cut (Roth and Frank, 2012).

These works inherently only support token-based alignment, with phrase-like alignment achieved by first merging tokens to phrases as a *preprocessing* step. Also, most of these approaches used no supervision or indirect supervision. For instance, both Punyakanok et al. (2004) and Kouylekov and Magnini (2005) applied the tree edit distance model with the Zhang and Shasha (1989) algorithm in their separate tasks (QA and RTE). The alignment objective is guided by minimum edit distance and no supervision from data was used. For indirect supervision, Wang and Manning (2010) extended the work of McCallum et al. (2005) and modeled alignment as latent variables. In their case supervision comes from the end application, for instance, whether the premise entails the hypothesis in the task of RTE, or whether the snippet contains an answer to the question in the task of QA. Heilman and Smith (2010) used tree kernels to search for the alignment that yields the lowest tree edit distance.

In a fully supervised fashion, a phrase-base alignment model was proposed by MacCartney et al. (2008), who also evaluated alignment quality intrinsically in terms of precision, recall and F_1. Their MANLI aligner aligns premise and hypothesis sentences for the task of natural language inference. It applies perceptron learning and handles

	michael	geht	davon	aus	,	dass	er	im	hausb	bleibt
michael	●									
assumes		●	●	●						
that						●				
he							●			
will										●
stay										●
in								●		
the								●		
house									●	

(a) word alignment matrix

michael – michael
michael assumes – michael geht davon aus ; michael geht davon aus ,
michael assumes that – michael geht davon aus , dass
michael assumes that he – michael geht davon aus , dass er
michael assumes that he will stay in the house – michael geht davon aus , dass er im haus bleibt
assumes – geht davon aus ; geht davon aus ,
assumes that – geht davon aus , dass
assumes that he – geht davon aus , dass er
assumes that he will stay in the house – geht davon aus , dass er im haus bleibt
that – dass ; , dass
that he – dass er ; , dass er
that he will stay in the house – dass er im haus bleibt ; , dass er im haus bleibt
he – er
he will stay in the house – er im haus bleibt
will stay – bleibt
will stay in the house – im haus bleibt
in the – im
in the house – im haus
house – haus

(b) MT phrase extraction results

three 1x1 alignment blocks:
michael - micheal; that - dass; he - er; house - hausb
one 1x3 alignment block:
assumes - geht davon aus
two 2x1 alignment blocks:
will stay - bleibt; in the - im

(c) Block alignment in monolingual alignment (1x1 blocks take 50% of all alignment)

Table 4.1.: Illustrative examples of how phrases are extracted in MT (b) and how block alignments are extracted in monolingual alignment (c).

phrase-based alignment of arbitrary phrase lengths. Thadani and McKeown (2011) optimized this model by decoding via Integer Linear Programming (ILP). Benefiting from modern ILP solvers, this led to an order-of-magnitude speedup. With extra syntactic constraints added, the exact alignment match rate for whole sentence pairs was also significantly improved. These work used two annotated word alignment datasets, MSR06 (Brockett, 2007) and Edinburgh++ (Cohn et al., 2008, Thadani et al., 2012).

Table 4.2 summarizes some of these approaches and their applied application. Finally, feature and model design in monolingual alignment is often inspired by bilingual work, including distortion modeling, phrasal alignment, syntactic constraints, etc (Och and Ney, 2003, DeNero and Klein, 2007, Bansal et al., 2011).

4.2.2.1. Evaluation

Monolingual alignment has previously discarded the usage of possible alignment since MacCartney et al. (2008), who needed a precise aligner trained on only sure alignments for the entailment end task. Alignment is evaluated in terms of precision, recall, F_1 and exact match rate. If we divide the types of alignment by gold standard vs. real prediction:

		gold standard	
		positive	negative
prediction	pos.	true positive (TP, “hit”)	false positive (FP, “false alarm”)
	neg.	false negative (FN, “miss”)	true negative (TN, “correct rejection”)

then:

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

paper	**structure**	**align**	**method**	**labeled align**	**feature**	**task**
Punyakanok et al. (2004)	dep tree	token	TED	N	no	QA
Kouylekov and Magnini (2005)	dep tree	token	TED	N	no	RTE
Chambers et al. (2007)	dep graph	token	stochastic local search, perceptron	Y	rich	RTE
MacCartney et al. (2008)	flat	phrase	perceptron	Y	rich	RTE
Mehdad (2009)	dep tree	token	Particle Swarm Optimization	N	no	RTE
Heilman and Smith (2010)	dep tree	token	TED	N	no	RTE/PP/QA
Wang and Manning (2010)	dep tree	token	CRF	N	rich	RTE/QA
Thadani and McKeown (2011), Thadani et al. (2012)	dep tree	phrase	perceptron, ILP	Y	rich	alignment
Roth and Frank (2012)	graph	token	min cut	N	similarity	predicate align
Yao et al. (2013b,a)	flat	token/phrase	CRF	Y	rich	RTE/PP/QA

Table 4.2.: Selected publications on monolingual alignment. The meaning of each column is as follows. structure: the required input sentence structure (flat, tree, or graph); align: the alignment type (token or phrase); method: main method used; labeled align: whether the training was supervised; feature: the type of features used; task: in what end tasks were alignment used.

Precision, recall and F_1 *only care about true positives*, i.e., those positive examples by the gold standard. In monolingual alignment, positive examples are those tokens that get aligned and negative examples are those that do not. We usually only care about whether we have correctly aligned those that should have been aligned, thus the measure of F_1 is a good fit.

If correct rejection (true negative) is important, then we compute accuracy instead:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy weighs equally true positives and true negatives. We want to make sure that the classifier recognizes both the positive and negative examples. Specifically in alignment, where most tokens are not aligned, the accuracy value will be very high in general and the difference is hard to tell. In this case we only report F_1 on positive (aligned) examples.

There is also a distinction between *micro* F_1 and *macro* F_1 here. Machine translation alignment computes micro F_1: the judgment of predicted alignment is accumulated over the whole dataset; thus micro F_1 is a measure of alignment performance *per aligned token.* Monolingual alignment computes macro F_1: for each sentence pair, precision and recall are computed; then they are averaged over all sentence pairs and their harmonic mean is computed as the final macro F_1. Thus macro F_1 is a measure of alignment performance *per sentence pair.* § **2.2.3.4 on page 42** gives a real example of computing these values. Micro F_1weighs equally among all tokens, since in MT the purpose is to induce phrase translation tables. Macro F_1weighs equally all sentence pairs, since monolingual alignment serves the end task of RTE or QA, where per-sentence accuracy is important.

4.2.2.2. Phrase/Block Alignment

As previously demonstrated in § 4.2.1.2, the phrase extraction algorithm in machine translation does not respect natural phrase boundaries. The extracted phrases are also not mutually exclusive. To give an estimation of alignment size distribution in a dataset, we simply count the number of MxN alignment blocks. For instance, in Table 4.1 (c), we count three 1x1 alignment blocks, one 1x3 alignment block, and two 2x1 alignment blocks. An MxN alignment block is a continuously fully filled M-by-N block. It does not necessarily represent natural phrases (NP, VP, PP, etc), but in many cases is good enough approximation. For instance, the following is a sample of alignment blocks from one of the monolingual alignment datasets described later:

united states . $\leftrightarrow$ u.s.

crashed into each other $\leftrightarrow$ collision

king norodom sihanouk $\leftrightarrow$ sihanouk

this $\leftrightarrow$ the matter

a telephone conversation $\leftrightarrow$ phone

information on this assassination attempt $\leftrightarrow$ information

In later sections we use alignment blocks to estimate the alignment distribution of several datasets.

4.2.3. Open-source Aligners

In this section we describe three open-source aligners. The first one is GIZA++, originally designed for MT but adapted for the monolingual alignment task. The other two are specifically designed for the monolingual case.

4.2.3.1. GIZA++ (Adapted)

GIZA++ (Och and Ney, 2003) is most commonly used in the task of bilingual alignment for machine translation. It implements the IBM models (Brown et al., 1993) and the HMM model (Vogel et al., 1996) for word alignment. Due to its nature of a generative model and language independence, it does not use any features based on lexical similarity.

To adapt GIZA++ to monolingual alignment, specifically, to monolingual alignment *evaluation*, one should organize the training data in a way that not only parallel sentences construct alignment pairs, but also each word's stem forms an identical and parallel pair. This is to enforce GIZA++'s belief to align lexically similar pairs. To be more specific, for a monolingual alignment task, suppose data is divided in two portions: TRAIN and TEST, and TRAIN contains minimally the following one pair of sentences:

`Casey loves Robin` $\leftrightarrow$ `Robin adores Casey too`

then the training data for GIZA++ consists of (all words are lowercased and stemmed):

- sentence pairs from both TRAIN *and* TEST, e.g., `casey love robin` $\leftrightarrow$ `robin adore casey too`. Since GIZA++ uses unsupervised training and favors more data, one should specifically include also TEST.
- identical word stem pairs formed from both TRAIN *and* TEST, such as `casey` $\leftrightarrow$ `casey`.
- (only in the alignment task with manually aligned data) labeled alignment pairs from *only* TRAIN, such as `love` $\leftrightarrow$ `adore` (if they are annotated as aligned).

The test data for GIZA++ consists of only the sentence pairs from TEST. However, since the training data for GIZA++ has already included TEST, one only needs to extract the already aligned TEST part after training is done. Usually alignment is run in both directions and heuristics are applied on the results to obtain the best result.

4.2.3.2. Meteor

The Meteor (Denkowski and Lavie, 2011) system is designed for evaluating machine translation performance by aligning the output with reference sentences. Meteor supports English, Czech, German, French, Spanish, and Arabic. In this chapter we only use it to align English sentences. It makes pairwise comparison between two strings of words and finds the alignment of highest cardinality with minimal number of crossing branches. Meteor inherently supports phrase-based alignment. String similarity is measured by mostly lexical/stem identity and a predefined paraphrase tables (5.27 million pairs) extracted using the foreign language pivoting technique (Bannard and Callison-Burch, 2005).

4.2.3.3. TED

The TED aligner is based on Tree Edit Distance, which computes the minimal cost of transforming from tree T_1 to tree T_2 under a set of predefined edit operations (such as insertion, deletion and substitution) and their associated cost (usually uniform cost). The TED aligner accepts tree-structured input. So the sentence pairs have to be parsed first.

The most popular algorithm to search for the aligning sequence with the minimal cost is the dynamic programming method of Zhang and Shasha (1989). Augsten et al. (2010) implemented for aligning database entries and I adapted it with dependence parse trees (§ 3.2 on page 90 gives a detailed description). The algorithm pairwisely compares each node pair from the two trees in a bottom-up and post-order manner. Word similarity is measured by stemming and WordNet synset relations.

4.2.3.4. Example

Table 4.3 on page 151 shows a real alignment example selected from the manually annotated MSR06 entailment corpus (described later in 4.4.1) for aligning the following sentence pair:

- source sentence: Three days after PeopleSoft bought JD Edwards in June 2003 , Oracle began its offer for PeopleSoft .
- target sentence: JD Edwards belongs to PeopleSoft .

Human annotators agreed on the following alignment pairs:

JD↔JD, Edwards↔Edwards, belongs to↔bought, PeopleSoft↔PeopleSoft, .↔.

Note that there is one 2x1 phrasal alignment, and there are two words of PeopleSoft in the source sentence. All three aligners failed to align belongs to↔bought, which is a difficult case and needs the context for inferring the alignment. The interesting case lies in the alignment between PeopleSoft↔PeopleSoft:

- GIZA++ made the correct alignment. It was very likely that the distortion model encouraged the first PeopleSoft in the source sentence to be aligned since it was relatively closer to other alignment.

- Meteor made the wrong alignment by aligning the second PeopleSoft in the source sentence. This can be explained by the minimal cross alignment principle employed by Meteor: the final alignment in Table Table 4.3 on page 151(b) looks monotonic without any cross alignment when aligning the second PeopleSoft in the source sentence.

- TED also made made the wrong alignment by aligning the second PeopleSoft in the source sentence. This can be explained by the order of how the minimum tree edit distance principle works: bottom-up and post-order of tree traversal. Thus the second PeopleSoft in the source sentence, a low-hanging leaf node in the parse tree, was aligned first.

The deficiencies of the three aligners above encourage us to design better aligner models that are able to make phrasal alignment with the ability to allow reasonable distortion and access more lexical resources.

4.3. Our Alignment Model

4.3.1. Markov Token Alignment

Our work is heavily influenced by the bilingual alignment literature, especially the discriminative model proposed by Blunsom and Cohn (2006). Given a source sentence $\mathbf{s}$ of length M, and a target sentence $\mathbf{t}$ of length N, the alignment from $\mathbf{s}$ to $\mathbf{t}$ is a sequence of target word indices $\mathbf{a}$, where $a_{i\in[1,M]} \in [0, N]$. We specify that when $a_i = 0$, source word s_i is aligned to a NULL state, i.e., deleted. This models a many-to-one alignment from source to target. Multiple source words can be aligned to the same target word, but not vice versa. One-to-many alignment can be obtained by running the aligner in the other direction. The probability of alignment sequence $\mathbf{a}$ conditioned on both $\mathbf{s}$ and $\mathbf{t}$ is then:

$$p(\mathbf{a} \mid \mathbf{s}, \mathbf{t}) = \frac{\exp(\sum_{i,k} \lambda_k f_k(a_{i-1}, a_i, \mathbf{s}, \mathbf{t}))}{Z(\mathbf{s}, \mathbf{t})}$$

	Three	days	after	PeopleSoft	bought	JD	Edwards	in	June	2003	,	Oracle	began	its	offer	for	PeopleSoft	.
JD						●												
Edwards							●											
belongs					🔴													
to					🔴													
PeopleSoft				●														
.																		●

(a) Gold standard: JD↔JD, Edwards↔Edwards, belongs to↔bought (red dots), PeopleSoft↔PeopleSoft, .↔..

	Three	days	after	PeopleSoft	bought	JD	Edwards	in	June	2003	,	Oracle	began	its	offer	for	PeopleSoft	.
JD						●												
Edwards							●											
belongs																		
to																		
PeopleSoft				●														
.																		●

(b) GIZA++, which failed to align belongs to↔bought.

	Three	days	after	PeopleSoft	bought	JD	Edwards	in	June	2003	,	Oracle	began	its	offer	for	PeopleSoft	.
JD						●												
Edwards							●											
belongs																		
to																		
PeopleSoft																	🟢	
.																		●

(c) Meteor/TED, which failed to align belongs to↔bought and made wrong alignment in PeopleSoft↔PeopleSoft (green dot).

Table 4.3.: GIZA++, Meteor and TED alignment for the sentence pair Three days after PeopleSoft bought JD Edwards in June 2003 , Oracle began its offer for PeopleSoft . ↔JD Edwards belongs to PeopleSoft . . Example comes from real alignment on one sentence pair from the MSR06 corpus, described later in § 4.4.1.

This assumes a first-order Conditional Random Field (Lafferty et al., 2001). The word alignment task is evaluated over F_1. Instead of directly optimizing F_1, we employ softmax-margin training (Gimpel and Smith, 2010) and add a cost function to the normalizing function $Z(\mathbf{s}, \mathbf{t})$ in the denominator, which becomes:

$$\begin{aligned} Z(\mathbf{s}, \mathbf{t}) &= \sum_{\hat{\mathbf{a}}} \exp(\sum_{i,k} \lambda_k f_k(\hat{a}_{i-1}, \hat{a}_i, \mathbf{s}, \mathbf{t}) \\ &\quad + cost(\mathbf{a_y}, \hat{\mathbf{a}})) \end{aligned}$$

where $\mathbf{a_y}$ is the true alignments. $cost(\mathbf{a_y}, \hat{\mathbf{a}})$ can be viewed as special "features" that encourage decoding to be consistent with true labels. It is only computed during training in the denominator because in the numerator $cost(\mathbf{a_y}, \mathbf{a_y}) = 0$. Hamming cost is used in practice without learning the weights (i.e., uniform weights). The more inconsistence there is between $\mathbf{a_y}$ and $\hat{\mathbf{a}}$, the more penalized is the decoding sequence $\hat{\mathbf{a}}$ through the cost function.

One distinction of this alignment model compared to other commonly defined CRFs is that the input is two dimensional: at each position i, the model inspects both the entire sequence of source words (as the observation) and target words (whose offset indices are states). The other distinction is that the size of its state space is not fixed (e.g., unlike POS tagging, where states are for instance 45 Penn Treebank tags), but depends on N, the length of target sentence. Thus we cannot "memorize" what features are mostly associated with what states. For instance, in the task of tagging mail addresses, a feature of "5 consecutive digits" is highly indicative of a POSTCODE. However, in the alignment model, it does not make sense to design features based on a hard-coded state, say, a feature of "source word lemma matching target word lemma" fires for state index 6.

To avoid this data sparsity problem, all features are defined *implicitly* with respect to the state. For instance:

$$f_k(a_{i-1}, a_i, \mathbf{s}, \mathbf{t}) = \begin{cases} 1 & \text{lemmas match: } s_i, t_{a_i} \\ 0 & \text{otherwise} \end{cases}$$

Thus this feature fires for, e.g.: $(s_3 = \text{sport}, t_5 = \text{sports}, a_3 = 5)$, and: $(s_2 = \text{like}, t_{10} = \text{liked}, a_2 = 10)$.

4.3.1.1. Symmetrization

To expand from many-to-one alignment to many-to-many, we ran the model in both directions and applied the following symmetrization heuristics (Koehn, 2010): INTERSECTION, UNION, GROW-DIAG-FINAL. However, this is not an ideal solution. Thus in the following section we describe how to expand the token-based model to phrase-based model.

4.3.2. Semi-Markov Phrase Alignment

The token-based model supports $1:1$ alignment. We first extend it in the direction of $l_s : 1$, where a target state spans l_s words on the source side (l_s source words align to 1 target word). Then we extend it in the direction of $1 : l_t$, where l_t is the target phrase length a source word aligns to (1 source word aligns to l_t target words). The final combined model supports $l_s : l_t$ alignment. Throughout this section we use Figure 4.2 as an illustrative example, which shows phrasal alignment between the source sentence: `Shops are closed up for now until March` and the target sentence: `Shops are temporarily closed down`.

$1:1$ alignment is a special case of $l_s : 1$ alignment where the target side state spans $l_s = 1$ source word, i.e., at *each* time step i, the source side word s_i aligns to one state a_i and the next aligned state a_{i+1} only depends on the current state a_i. This is the Markovian property of the CRF. When $l_s > 1$, during the time frame $[i, i+l_s)$, all source words $[a_i, a_{i+l_s})$ share the same state a_i. Or in other words, the state a_i "spans" the following l_s time steps. The Markovian property still holds "outside" the time frame l_s,

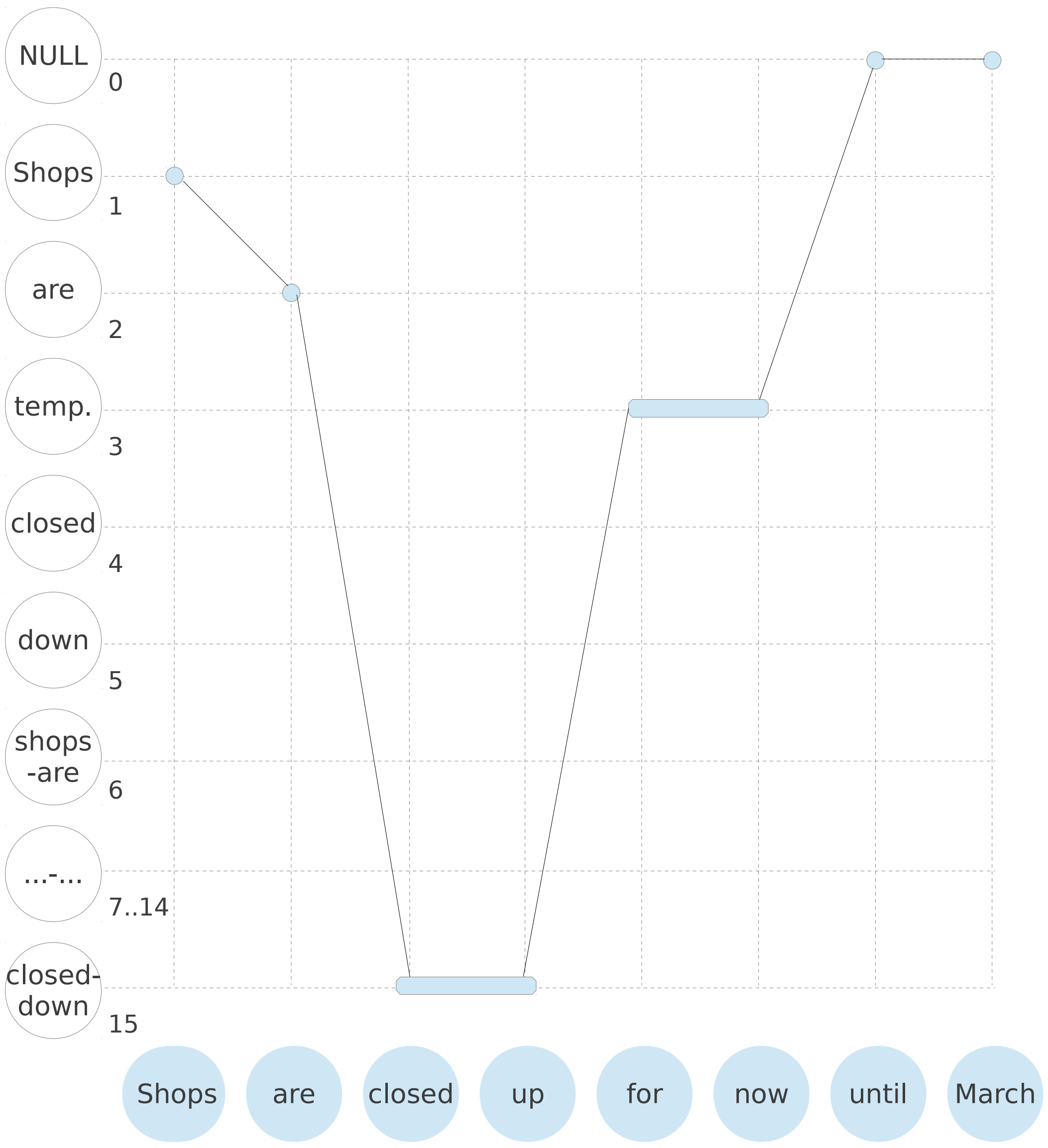


Figure 4.2.: A semi-Markov phrase-based model example and the desired Viterbi decoding path. Shaded horizontal circles represent the source sentence (`Shops are closed up for now until March`) and hollow vertical circles represent the hidden states with state IDs for the target sentence (`Shops are temporarily closed down`). State 0, a NULL state, is designated for deletion. One state (e.g. state 3 and 15) can span multiple consecutive source words (a semi-Markov property) for aligning phrases on the source side. States with an ID larger than the target sentence length indicate "phrasal states" (states 6-15 in this example), where consecutive target tokens are merged for aligning phrases on the target side. Combining the semi-Markov property and phrasal states yields for instance, a 2×2 alignment between `closed up` in the source and `closed down` in the target.

i.e., a_{i+l_s} still only depends on a_i, the previous state l_s time steps ago. But "within" the time frame l_s, the Markovian property does not hold any more: $[a_i, ..., a_{i+l_s-1}]$ are essentially the same state a_i. This is the semi-Markov property . States can be distinguished by this property into two types: *semi-Markovian states* and *Markovian states.*

We have generalized the regular CRF to a semi-Markov CRF. Now we define it by generalizing the feature function:

$$p(\mathbf{a} \mid \mathbf{s}, \mathbf{t}) = \frac{\exp(\sum_{i,k,l_s} \lambda_k f_k(a_{i-l_s}, a_i, \mathbf{s}, \mathbf{t}))}{Z(\mathbf{s}, \mathbf{t})}$$

At time i, the k-th feature function f_k mainly extracts features from the pair of source words $(s_{i-l_s}, ..., s_i]$ and target word t_{a_i} (still with a special case that $a_i = 0$ marks for deletion). Inference is still Viterbi-like: except for the fact during maximization, the Viterbi algorithm not only checks the previous *one* time step, but *all* l_s time steps. Suppose the allowed maximal source phrase length is L_s, define $V_i(a \mid \mathbf{s}, \mathbf{t})$ as the highest score along the decoding path until time i ending with state a:

$$V_i(a \mid \mathbf{s}, \mathbf{t}) = \max_{a_1, a_2, \ldots a_{i-1}} p(a_1, a_2, \ldots, a_i = a \mid \mathbf{s}, \mathbf{t})$$

then the recursive maximization is:

$$\begin{aligned} V_i(a \mid \mathbf{s}, \mathbf{t}) &= \max_{a'} \max_{l_s = 1 \ldots L_s} [V_{i-l_s}(a' \mid \mathbf{s}, \mathbf{t}) \\ &\quad + \Psi_i(a', a, l_s, \mathbf{s}, \mathbf{t})] \end{aligned}$$

with factor:

$$\Psi_i(a', a, l_s, \mathbf{s}, \mathbf{t}) = \sum_k \lambda_k f_k(a'_{i-l_s}, a_i, \mathbf{s}, \mathbf{t})$$

and the best alignment $\mathbf{a}$ can be obtained by backtracking the last state a_M from $V_M(a_M \mid \mathbf{s}, \mathbf{t})$.

Training a semi-Markov CRF is very similar to the inference, except for replacing maximization with summation. The forward-backward algorithm should also be used to dynamically compute the normalization function $Z(\mathbf{s}, \mathbf{t})$. Compared to regular CRFs, a semi-Markov CRF has a decoding time complexity of $O(L_s M N^2)$, a constant factor L_s (usually 3 or 4) slower.

To extend from $1 : 1$ alignment to $1 : l_t$ alignment with one source word aligning to l_t target words, we simply explode the state space by L_t times with L_t the maximal allowed target phrase length. Thus the states can be represented as an $N \times L_t$ matrix. The state at (j, l_t) represents the target phrase $[t_j, ..., t_{j+l_t})$. In this paper we distinguish states by three types: NULL state $(j = 0, l_t = 0)$, *token state* $(l_t = 1)$ and *phrasal state* $(l_t > 1)$.

To efficiently store and compute these states, we linearize the two dimensional matrix with a linear function mapping uniquely between the state ID and the target phrase offset/span. Suppose the target phrase t_j of length $l_{t_j} \in [1, L_t]$ holds a position $p_{t_j} \in [1, N]$, and the source word s_i is aligned to this state (p_{t_j}, l_{t_j}), a tuple for (position, span). Then state ID a_{s_i} is computed as:

$$a_{s_i}(p_{t_j}, l_{t_j}) = \begin{cases} p_{t_j} & l_{t_j} = 1 \\ N + (p_{t_j} - 1) \times L_t + l_{t_j} & 1 < l_{t_j} \leq L_t \end{cases}$$

Assume in Figure 4.2, $L_t = 2$, then the state ID for the phrasal state (5, 2) `closed-down` with $p_{t_j} = 5$ for the position of word `down` and $l_{t_j} = 2$ for the span of 2 words (looking "backward" from the word `down`) is: $5 + (5 - 1) \times 2 + 2 = 15$.

Similarly, given a state id a_{s_i}, the original target phrase position and length can be recovered through integer division and modulation. Thus during decoding, if one output state is 15, we would know that it uniquely comes from the phrasal state (5,2), representing the target phrase `closed down`.

This two dimensional definition of state space expands the number of states from $1+N$ to $1 + L_t N$. Thus the decoding complexity becomes $O(M(L_t N)^2) = O(L_t^2 M N^2)$ with a usual value of 3 or 4 for L_t.

Now we have defined separately the $l_s : 1$ model and the $1 : l_t$ model. We can simply merge them to have an $l_s : l_t$ alignment model. The semi-Markov property makes it possible for any target states to align phrases on the source side, while the two dimensional state mapping makes it possible for any source words to align phrases on the target side. For instance, in Figure 4.2, the phrasal state a_{15} represents the two-word phrase `closed down` on the target side, while still spanning for two words on the source side, allowing a 2×2 alignment. State a_{15} is phrasal, and at source word position 3 and 4 (spanning `closed up`) it is semi-Markovian. The final decoding complexity is $O(L_s L_t^2 M N^2)$, a factor of $30 \sim 60$ times slower than the token-based model (with a typical value of 3 or 4 for L_s and L_t).

In the following we describe features.

4.3.3. Feature Design

The token version of jacana-align was originally (Yao et al., 2013b) equipped with a compact set of features and minimal lexical resources (only WordNet was used) for fast run time. The phrase version (Yao et al., 2013a) introduced more lexical resources based on the Paraphrase Database (PPDB) and semantic relatedness (Han et al., 2013). In November 2013 jacana-align went through a series of "upgrades", for the purpose of *real world* usage. The aim also includes platform-independence: it runs on any system with a Java virtual machine (from version 6) and it does not have any external dependencies. The following lists the current feature set:

String Similarity Features include the following similarity measures: Jaro Winkler, Dice Sorensen, Hamming, Jaccard, Levenshtein, NGram overlapping and common prefix matching.[2] Also, two binary features are added for identical match and identical match ignoring case.

POS Tags Features are binary indicators of whether the POS tags of two words match. Also, a "$\text{pos}_{\text{src}}2\text{pos}_{\text{tgt}}$" feature fires for each word pair, with respect to their POS tags. This would capture, e.g., "vbz2nn", when a verb such as *arrests* aligns with a noun such as *custody*.

Positional Feature is a real-valued feature for the positional difference of the source

[2]Of these features the trained aligner preferred Dice Sorensen and NGram overlapping.

and target word (abs($\frac{m}{M} - \frac{a_m}{N}$)). This allows to learn a preference over alignment near the diagonal.

WordNet Features indicate whether two words are of the following relations of each other: hypernym, hyponym, synonym, derived form, entailing, causing, members of, have member, substances of, have substances, parts of, have part; or whether their lemmas match. We also found that each word has to be POS tagged to get an accurate relation, otherwise this feature will not help. Specifically, the "Snow version"[3] of WordNet with 400 thousand automatically expanded synsets from Snow et al. (2006) replaced the original WordNet (Fellbaum, 1998). In my quick internal evaluation it improved the final F_1 by 0.2%.

Distortion Features measure how far apart the aligned target words of two consecutive source words are: abs($a_m + 1 - a_{m-1}$). This learns a general pattern of whether these two target words aligned with two consecutive source words are usually far away from each other, or very close. We also added special features for corner cases where the current word starts or ends the source sentence, or both the previous and current words are deleted (a transition from NULL to NULL).

Contextual Features indicate whether the left or the right neighbor of the source word and aligned target word are identical or similar. This helps especially when aligning functional words, which usually have multiple candidate target functional words to align to and string similarity features cannot help. We also added features for neighboring POS tags matching.

Chunking Features are binary indicators of whether the phrase types of two phrases match. Also, we added indicators for mappings between source phrase types and target phrase types, such as "vp2np", meaning that a verb phrase in the source is mapped to a noun phrase in the target.

Moreover, we introduced the following lexical features:

PPDB Features (Ganitkevitch et al., 2013) utilize PPDB (roughly size L),[4] with 270 thousand paraphrase pairs. Originally various paraphrase conditional probability

[3]http://ai.stanford.edu/~rion/swn/
[4]http://paraphrase.org

was employed. For instance, for the ADJP/VP phrase pair `capable of` and `able to`, there are the following minus-log probabilities:

$$p(lhs|e1) = 0.1, p(lhs|e2) = 0.3, p(e1|lhs) = 5.0$$
$$p(e1|e2) = 1.3, p(e2|lhs) = 6.7, p(e2|e1) = 2.8$$
$$p(e1|e2, lhs) = 0.6, p(e2|e1, lhs) = 2.3$$

where $e1/e2$ are the phrase pair, and lhs is the left hand side syntactic non-terminal symbol. Later I discovered that a simple indicator feature of whether a pair of words exist in PPDB worked better than the original features based on $p(\cdot \mid \cdot)$ probabilities. Thus the final features based on PPDB were all binary indicator features.

Nicknames: a list of common nicknames mined from the web.[5] A few examples: Aaron–Erin, Ron, Ronnie and Abigail–Abby, Nabby, Gail.

Wiktionary: 90 thousand English words from Wiktionary. It contained WordNet-like relations, such as that the adjective form of minute has synonyms infinitesimal, insignificant, tiny, etc, and antonyms big, enormous, colossal, etc.

word2vec: a similarity measure based on continuous bag of words (Mikolov et al., 2013), trained on the first 100MB of English Wikipedia.

The following two types of features were *removed*:

Semantic Relatedness feature outputs a single scaled number in $[0, 1]$ from the best performing system (Han et al., 2013) of the *Sem 2013 Semantic Textual Similarity (STS) task from UMBC. This feature mainly deals with cases where "related" words cannot be well measured by either paraphrases or distributional similarities. For instance, in one alignment dataset annotators aligned `married` with `wife`. Adding a few other words as comparison, the Han et al. (2013) system gives the following similarity scores:

```
married/wife:  0.85
married/husband:  0.84
married/child:  0.10
married/stone:  0.01
```

However, this feature depends on the web service of UMBC. I removed it for two

[5] `http://usgenweb.org/research/nicknames.shtml`

reasons: 1. web API calling severely reduced the system speed; 2. there is no guarantee that this service will outlive jacana-align.

Name Phylogeny feature (Andrews et al., 2012) uses a string transducer to model how one name evolves to another. Examples below show how similar is the name `Bill` associated with other names in log probability:

```
Bill/Bill:  -0.8
Bill/Billy:  -5.2
Bill/William:  -13.6
Bill/Mary:  -18.6
```

However, the model file from Andrews et al. (2012) was only compatible on Mac and Linux systems, but not on Windows. Thus the name phylogeny model was replaced by a fixed set of common nick names (described above).

Finally, one decision we made during feature design was not using any parsing-based features, with a permissive assumption that the input might not be well-formed English, or even not complete sentences (such as fragmented snippets from web search). The "deepest" linguistic processing stays at the level of tagging and chunking, making the model more easily extendible to other languages.

4.3.3.1. Feature Value for Phrase Alignment

In the phrase-based model, the width of a state span over the source words depends on the competition between features fired on the phrases as a whole vs. the consecutive but individual tokens. We found it critical to assign feature values "fairly" among tokens and phrases to make sure that semi-Markov states and phrasal states fire up often enough for phrasal alignments.

To illustrate this in a simplified way, take `closed up`↔`closed down` in Figure 4.2, and assume the only feature is the normalized number of matching tokens in the pair. Then this feature firing on the following pairs would have values (the normalization factor is the maximal phrase length):

`closed↔closed`	1.0
`closed up↔closed`	0.5
`closed up↔up`	0.5
`closed up↔closed down`	0.5
...↔...	...

The desired alignment `closed up↔closed down` would not have survived the state competition due to its weak feature value. In this case the model would simply prefer a token alignment `closed↔closed` and up↔... (probably NULL).

Thus we upweighted feature values by the maximum source or target phrase length to encourage phrasal alignments, in this case `closed up ↔closed down`:1.0. Then this alignment would have a better chance to be picked out with additional features, such as with the PPDB and chunking features, which are also upweighted by maximum phrase lengths.

4.3.4. Implementation and Training

Since no generic off-the-shelf CRF software is designed to handle the special case of dynamic state indices and feature functions, we implemented this aligner model in the Scala programming language, which is fully interoperable with Java. For the Semi-Markov CRF, we referenced the implementation of Sarawagi and Cohen (2004)[6] and also implemented it in Scala. We used the L2 regularizer and LBFGS for optimization. OpenNLP[7] provided the POS tagger and JWNL[8] interfaced with WordNet (Fellbaum, 1998).

[6]`http://crf.sf.net`
[7]`http://opennlp.apache.org/`
[8]`http://jwordnet.sf.net`

4.4. Experiments

4.4.1. Datasets

4.4.1.1. MSR06 and Edinburgh++

Two annotated datasets have been previously published in alignment research. **MSR06**[9] (Brockett, 2007) has annotated alignments on the 2006 PASCAL RTE2 development and test corpora, with 1600 pairs in total. Semantically equivalent words and phrases in the premise and hypothesis sentences are aligned in a manner analogous to alignments in statistical machine translation. This dataset is asymmetric: on average the premises contain 29 words and the hypotheses 11 words. **Edinburgh++**[10] is a version of the Edinburgh paraphrase corpus (Cohn et al., 2008) that was revised by Thadani et al. (2012) such that some annotation errors were further cleaned. The Edinburgh corpus contains English-English sentence pairs from the following resources:

1. the Multiple-Translation Chinese (MTC) corpus;
2. Jules Verne's novel Twenty Thousand Leagues Under the Sea.
3. the Microsoft Research paraphrase corpus (Dolan et al., 2004).

The corpus is more balanced and symmetric: the source and target sentences are both 22 words long on average. Before it was manually aligned, it was pre-aligned with GIZA++ trained on $54,615$ translation sentence pairs from the MTC corpus. Table 4.4 shows some statistics.

Both corpora contain mostly token-based alignment. For MSR06, MacCartney et al. (2008) showed that setting the allowable phrase size to be greater than one only increased F_1 by 0.2%. For Edinburgh++, the annotation guideline[11] explicitly instructs to "prefer smaller alignments whenever possible". Statistics shows that single token alignment counts 98% and 95% of total alignments in these two corpora separately. With such

[9] `http://www.cs.biu.ac.il/nlp/files/RTE2006Aligned.zip`
[10] `http://www.ling.ohio-state.edu/scott/#edinburgh-plusplus`
[11] `http://staffwww.dcs.shef.ac.uk/people/T.Cohn/paraphraseguidelines.pdf`

	train	**test**	**length**	**%aligned**	**%1x1 align**	**GIZA++**
MSR06	800	800	29/11	36%	96%	No
Edinburgh++	714	306	22/22	78%	95%	Yes
MTReference	2000	1998	22/17	79%	88%	Yes

Table 4.4.: Statistics of the three manually aligned datasets, divided into training and test in sentence pairs. The length column shows average lengths of source and target sentences in a pair. %aligned is the percentage of aligned words over all words. %1x1 align is the percentage of one-to-one alignment. GIZA++ indicates whether the corpus was pre-aligned by GIZA++ before presenting to annotators.

a heavy imbalance towards only token-based alignment, a phrase-based aligner would learn feature weights that award token alignments more than phrasal alignments.

To compensate for the fact of overwhelming 1x1 alignment, we first enabled *possible* alignment in the two datasets. Both MSR06 and Edinburgh++ were aligned with multiple annotators. The unanimously agreed alignment was taken as *sure* alignment, and the rest *possible*. Research in monolingual alignment has diverged from bilingual alignment research, in that monolingual alignment research sometimes discards possible alignments and uses only sure alignment. This has the unintended side effect of reducing the number of non-identical alignments and the number of many-one/many-many alignments. As we will show later, these are among the most challenging parts of the monolingual alignment task. When we include the possible alignment in the two datasets, percentage of 1x1 alignment drops to 97% (MSR06) and 92% (Edinburgh++) respectively. Table 4.5 summarizes the percentage of various alignment sizes.

4.4.1.2. New Dataset: MTReference

Even with possible alignment, the two aforementioned datasets still contain more than 90% 1x1 alignment. Thus we created a third dataset, **MTReference**. The sentences came from multiple references in a machine translation task. The dataset was initially created for a sentence compression task: the length ratio between target and source sentences is between 0.7 and 0.8. The whole dataset contains 3998 sentence pairs. To

	1x1	**1x2**	**1x3**	**2x2**	**2x3**	**3x3**	**more**
MSR06	97.9	1.7	0.4	0.0	0.0	0.0	0.0
MSR06(possible)	96.9	2.6	0.4	0.0	0.0	0.0	0.1
Edinburgh++	94.6	3.1	1.2	0.4	0.5	0.2	0.0
Edinburgh++(possible)	91.8	4.2	1.8	0.8	0.8	0.6	0.0
MTReference	87.8	8.4	3.4	0.0	0.0	0.0	0.4
MTReference(possible)	73.3	13.9	8.4	1.0	2.0	1.4	0.0

Table 4.5.: Percentage of various alignment sizes (undirectional, e.g., 1x2 and 2x1 are merged) in the *training* portion of various datasets. A size of MxN is defined as M continuous tokens in one sentence all align with N continuous tokens in the other sentence. M and N are only measures for the *size* of continuous alignment. They are not necessarily defined by natural phrase boundaries.

ease the burden of annotation, all sentences were pre-aligned by GIZA++. Then alignment was corrected by Amazon Mechanical Turk (two Turkers per HIT). After enabling possible alignment, the percentage of 1x1 alignment dropped from 88% to 73%.

4.4.2. Notes on Evaluation and Datasets

4.4.2.1. Datasets

There are two threads of work on alignment evaluation due to the order of published papers and datasets. Careful attention should be paid for future researchers in spirit of fair comparison among scientific works.

The first thread includes that of (MacCartney et al., 2008, Thadani and McKeown, 2011, Yao et al., 2013b), which solely used the MSR06 dataset. This is mainly due to that the other corpus, Edinburgh (Cohn et al., 2008), published at the same time of MacCartney et al. (2008) and the latter was not able to use this dataset. Thadani and McKeown (2011) improved over MacCartney et al. (2008) by applying ILP during decoding. Thus the themes of evaluation in these work include both alignment F_1 and speed comparison. On the token-alignment work, we (Yao et al., 2013b) followed the previous papers and evaluated also on F_1 and run time, with results reported in section

§ 4.4.3 on page 166.

The second thread includes that of (Thadani et al., 2012, Yao et al., 2013a). Thadani et al. (2012) revised the Edinburgh dataset by correcting wrong annotations. However, the final released dataset, Edinburgh++, was further improved and was not exactly the same one reported in Thadani et al. (2012). We (Yao et al., 2013a) reported numbers from both token-alignment and phrase-alignment on both the MSR06 and Edinburgh++ datasets. The results on MSR06 are directly comparable with previous results, but the results on Edinburgh++ are only roughly comparable with Thadani et al. (2012).

4.4.2.2. Evaluation Metrics: Token vs. Phrase

Even though the most of the work in comparison does phrase alignment, evaluation has been consistently performed on *token alignment macro* F_1 since MacCartney et al. (2008). In some cases, this measure rates down phrase alignment. For instance, if the gold annotation token-aligns New York with New York (`New`$\leftrightarrow$`New`, `York`$\leftrightarrow$`York`), a token aligner's perfect output (`New`$\leftrightarrow$`New`, `York`$\leftrightarrow$`York`) will get a full score:

$$\begin{array}{ccc} & \text{New} & \text{York} \\ \text{New} & \bullet & \\ \text{York} & & \bullet \end{array}$$

However, a phrase aligner's phrase-based output (`New`$\leftrightarrow$`New`, `New`$\leftrightarrow$`York`, `York`$\leftrightarrow$`York`, `York`$\leftrightarrow$`New`):

$$\begin{array}{ccc} & \text{New} & \text{York} \\ \text{New} & \bullet & \bullet \\ \text{York} & \bullet & \bullet \end{array}$$

will only have two-thirds of the full score (precision=0.5, recall=1, $F_1 = 2/3$) according to the gold-standard token alignment.

Concrete evaluation metrics definition can be found in § 4.2.2.1 on page 144.

4.4.2.3. "Natural Phrase" Alignment

One other reason for evaluating with token alignment F_1 instead of phrase alignment F_1 is that there is no gold standard data with naturally annotated phrase boundaries. When annotators were asked to align words, they did not intentionally create "block" alignment (e.g., `New`↔`New`, `New`↔`York`, `York`↔`York`, `York`↔`New`), but mostly "stripe" alignment (e.g., `New`↔`New`, `York`↔`York`), just like the examples shown above. The question boils down to: given a token-aligned dataset, how can we properly extract phrase alignment?

Originally I had used a phrase chunker to create phrase alignment. The OpenNLP chunker was first run through the sentences. Then for each phrase pair, if each token in the source phrase is aligned to a token in the target phrase in a monotonic way, and vice versa, these alignments are merged to form one single phrasal alignment. One example is:

$$\begin{array}{ccc} & \text{New} & \text{York} \\ \text{New} & \bullet & \\ \text{York} & & \bullet \end{array} \longrightarrow \begin{array}{ccc} & \text{New} & \text{York} \\ \text{New} & \bullet & \bullet \\ \text{York} & \bullet & \bullet \end{array}$$

Some other examples that came from synthesized phrases include: `two Atlanta-based companies`↔`two Atlanta companies`, `the UK`↔`the UK`, `the 17-year-old`↔`the teenager`, `was held`↔`was held`. The phrase boundaries in these alignments are licensed by the OpenNLP chunker.

Table 4.6 lists the percentage of various alignment block sizes after the merge and compares phrase sizes extracted from possible alignment. Three observations can be made:

1. the portion of non 1x1 alignments increases to 10% $\sim$ 20% after merging, showing that this is an effective way for reducing 1x1 alignment;

2. allowing a maximal phrase length of 3 covers 98% $\sim$ 99% of total alignments, thus a phrase length larger than 3 would be a bad trade-off for coverage vs. speed;

3. the phrase synthesizing method creates more block alignment (mainly 2x2 and 3x3) than including the possible alignment.

However, one argument against the phrase synthesizing method is that the phrases created come from a chunker – they are not *natural* phrases. Thus in the following evaluation, we did not evaluate on the phrase synthesized datasets, but only evaluated on sure

	1x1	**1x2**	**1x3**	**2x2**	**2x3**	**3x3**	**more**
MSR06	97.9	1.7	0.4	0.0	0.0	0.0	0.0
MSR06(possible)	96.9	2.6	0.4	0.0	0.0	0.0	0.1
MSR06(chunker)	**89.2**	1.9	0.3	**5.7**	0.0	**1.9**	0.8
Edinburgh++	94.6	3.1	1.2	0.4	**0.5**	0.2	0.0
Edinburgh++(possible)	91.8	4.2	1.8	0.8	**0.8**	0.6	0.0
Edinburgh++(chunker)	**81.9**	3.5	0.8	**8.3**	**0.4**	**3.0**	2.1

Table 4.6.: Comparison of different methods aiming to create more phrase alignment from the original datasets. (possible) enables extra alignments not unanimously agreed by annotators. (chunker) synthesized blocked phrase alignments from monotonic token-alignment by running the OpenNLP chunker to get phrase boundaries from only the sure alignment. The (chunker) method creates more 2x2/3x3 block alignment.

alignment and sure+possible alignment. Readers can still refer to Yao et al. (2013a) for results on the phrase synthesized datasets.

4.4.3. Evaluation: the General Picture

To make results directly comparable, we closely followed the setup of MacCartney et al. (2008), Thadani and McKeown (2011) and Thadani et al. (2012). For the sentence pairs in the three datasets, we took the longer sentence as the source sentence and the shorter one as the target, then ran the aligners in both directions, S2T and T2S. Experiments showed that the T2S direction consistently outperformed the S2T direction. For post-processing heuristics, INTERSECTION had better precision, UNION gave better recall, but neither of them yielded better F_1's than T2S. Thus in the following we only report scores from running in the T2S direction.

4.4.3.1. Baselines

The baseline systems were GIZA++, Meteor, and TED, all of which were described in § 4.2.3 on page 147. Among them, GIZA++ was trained on each dataset separately (with specific procedures described in § 4.2.3.1 on page 147) while Meter and TED were used out of the box without training.

Additionally, we cited the numbers from the Stanford RTE system (Chambers et al., 2007) and the MANLI family of aligners. MANLI was first developed by MacCartney et al. (2008), and then improved by Thadani and McKeown (2011), Thadani et al. (2012) with faster and exact decoding via ILP. There are five versions to be compared here:

- **MANLI**: the original version.
- **MANLI-approx.**: re-implemented version by Thadani and McKeown (2011).
- **MANLI-exact**: decoding via ILP solvers.
- **MANLI-constraint**: MANLI-exact with hard syntactic constraints, mainly on common "light" words (determiners, prepositions, etc.) attachment to boost exact match rate.
- **MANLI-joint** (Thadani et al., 2012): an improved version of MANLI-constraint that not only models phrasal alignments, but also alignments between dependency arcs, with reported numbers on the original Edinburgh paraphrase corpus.

4.4.3.2. Results

Performance was evaluated by macro-averaged precision, recall, F_1 of aligned token pairs, and exact (perfect) match rate for a whole pair, shown in Table 4.7. The following observation can be made across different datasets:

Pre-alignment affects end performance, both human's and machine's Out of all three datasets, only MSR06 was not pre-aligned when presenting to the annotators:

	MSR06	Edinburgh++	MTReference
pre-aligned by GIZA++?	N	Y	Y
GIZA++ score	78.3%	85.5%	77.4%
best score	88.3%	86.7%	77.4%

GIZA++ had an F_1 of 78.3% on this dataset, 10 points below (78.3% vs. 88.3%) the best performing system, our token aligner. However, the other two datasets, Edinburgh++ and MTReference, were all pre-aligned by GIZA++. On these two datasets, GIZA++ was only either slightly suboptimal (85.5% vs. 86.7% on Edinburgh++) or on par with the best system (77.4% on MTReference).

This shows that the final annotation can be heavily influenced by pre-alignment. Or in other words, human annotators might have made different alignment when being presented with an not-aligned sentence pair. We show some of these examples in Appendix A on page 247.

Best alignment scores reflect dataset quality The best scores on the three datasets and their annotators are:

MSR06	88.3%	3 professional annotators from the Butler Hill Group
Edinburgh++	86.7%	2 linguistics graduate students
MTReference	77.4%	Amazon Mechanical Turk

We hypothesize that the better scores a statistically-trained aligner obtains, the more consistently the dataset was annotated. This hypothesis also might correspond to the types of annotators: MSR06 was obtained with intersecting results from 3 annotators, thus the highest quality; Edinburgh++ followed: a fixed set of two annotators can still make consistent alignment; MTReference outsourced the task to MTurk, whose quality cannot be constantly guaranteed even after vetting.

Discriminative aligners outperform other aligners There are three types of aligners in the evaluation: rule-based (TED: minimal edit distance; Meteor: minimal cross alignment), generative (GIZA++), discriminative (jacana-align and the MANLI* family). The discriminative aligners outperformed other aligners on MSR06. jacana-align also outperformed other aligners on Edinburgh++ and MTReference, where MANLI's performance was either not directly comparable or missing (MANLI* is not open-source).

jacana-align also did better than the MANLI family on the MSR06 dataset. There are two possible explanations here: jacana-align uses more lexical resources, or the CRF model of jacana-align works better than the perceptron model of MANLI on this dataset. We think it is the latter, for the following reasons:

- we have shown in Yao et al. (2013b) that even with only access to WordNet, jacana-align also outperformed MANLI, which accessed about 5GB of lexical resources;
- we later show in ablation test that the alignment performance on these datasets are not sensitive to lexical resources;
- the CRF model we have employed has one merit over MANLI: it performs global decoding over the whole sentence to find the optimal alignment; this rules out ambiguous alignments especially in the case of multiple stop words; the perceptron model employed by MANLI does not carry this merit.

jacana-align is a high precision aligner Both the token and phrase version of jacana-align had the best precision scores in all datasets, with moderate recall scores. This shows that jacana-align is conservative about making new alignments. We later in § 4.6 on page 180 show this property suits well in downstream applications which require high precision, such as question answering.

The more non 1x1 alignment, the better the phrase aligner works There are two reasons that the phrase aligner might not work better in a dataset with majorly 1x1 alignment (such as MSR06: 97.9% 1x1 alignment):

- extra phrase-based features might degrade the aligner performance when most alignment types are 1x1 blocks;
- the token-based evaluation metrics might rate down phrase alignment, as shown in § 4.4.2.2 on page 164.

The following table shows a trend that, when there are more non 1x1 alignment blocks, the phrase aligner's performance goes up and eventually beats the token aligner. We continue to show in the next section that this trend holds when we evaluate on the possible alignment of each dataset, where 1x1 alignment is even less.

Corpus	**System**	**P %**	**R %**	**F$_1$ %**	**E %**
MSR06 (1x1: 97.9%)	GIZA++	82.5	74.4	78.3	14.0
	TED	80.6	79.0	79.8	13.5
	Stanford RTE$*$	82.7	75.8	79.1	-
	Meteor	82.5	81.2	81.9	15.0
	MANLI$*$	85.4	85.3	85.3	21.3
	MANLI-approx.$\triangleleft$	87.2	**86.3**	86.7	24.5
	MANLI-exact$\triangleleft$	87.2	86.1	86.8	24.8
	MANLI-constraint$\triangleleft$	89.5	86.2	87.8	33.0
	our token aligner	**93.3**	83.8	**88.3**	**34.8**
	our phrase aligner	91.5	83.5	87.3	32.0
Edinburgh++ (1x1: 94.7%, pre-aligned by GIZA++)	GIZA++	89.7	81.7	85.5	13.1
	TED	79.0	60.7	68.7	4.2
	Meteor	88.3	80.5	84.2	12.7
	MANLI-joint$\triangleright$	76.6	**83.8**	79.2	12.2
	our token aligner	**90.8**	82.9	**86.7**	15.7
	our phrase aligner	90.4	81.9	85.9	13.7
MTReference (1x1: 87.8%, pre-aligned by GIZA++)	GIZA++	70.1	**86.3**	**77.4**	1.6
	TED	80.5	50.9	62.4	0.4
	Meteor	85.1	66.7	74.8	0.8
	our token aligner	**86.8**	68.4	77.1	1.2
	our phrase aligner	85.7	70.6	**77.4**	1.9

Table 4.7.: Results on the three datasets. E% stands for exact (perfect) match rate. Systems marked with $*$ are reported by MacCartney et al. (2008), with $\triangleleft$ by Thadani and McKeown (2011), with $\triangleright$ by Thadani et al. (2012) (but not directly comparable since they used the original Edinburgh dataset, not Edinburgh++). Results of GIZA++ were obtained after applying the INTERSECTION heuristic on the GIZA++ alignments run in both directions. Results of the token and phrase aligners are from the direction of T2S.

corpus	%1x1	token aligner F_1	phrase aligner F_1
MSR06	97.9%	88.3%	87.3%
Edinburgh++	94.7%	86.7%	85.9%
MTReference	87.8%	77.1%	77.4%

4.4.4. Evaluation: Identical vs. Nonidentical

One way to investigate what aligners have learned from the data is to divide the scores by (easy) identical and (hard) nonidentical alignment. We selected the two best performing open-source aligners, GIZA++ and Meteor, and compared them with the token aligner and the phrase aligner. Identical alignment was simply judged by word matching while nonidentical alignment was the rest. Result is shown in Table 4.8, with subscript i showing identical alignment scores and n showing nonidentical. Due to the complexity of the table with various results on 6 datasets, we describe the observations verbosely in the following.

Identical alignment performance reflects the quality of alignment models. Aligners do not need any external resources for making most identical alignment. But the structured prediction can still make a difference, especially in the case of multiple possible alignment for the same word, such as stop words. The following table shows the best F_{1i} values for identical alignment in the 3 datasets (with sure or sure+possible alignment).

	MSR06		Edinburgh++		MTReference	
	sure	s+p	sure	s+p	sure	s+p
% identical	69.7	68.0	70.8	62.2	55.3	34.8
GIZA++ F_{1i}%	91.5	91.9	96.6	96.9	**94.3**	**96.6**
Meteor F_{1i}%	93.5	93.9	95.9	96.3	92.7	94.4
token/phrase F_{1i}%	**95.8**	**96.0**	**96.7**	**97.2**	93.5	94.8

On the relatively higher-quality datasets of MSR06 and Edinburgh++, both the token and phrase version of jacana-align had a stable F_{1i} around 96% ∼ 97%. Meteor followed and GIZA++ was the third. MTReference is noisier and GIZA++ had the best performance. We think this is because GIZA++ aligned multiple stop words in the sentence pair and Turkers were not able to correct all of them. Overall, the global decoding nature of CRF-based jacana-align showed its power in an alignment task that needs smart decisions on ambiguous alignment.

Nonidentical alignment performance is moderate and needs improvement. Making correct nonidentical alignment mostly needs the help of external knowledge. The following table summarizes the best F_{1n}:

	MSR06		Edinburgh++		MTReference	
	sure	s+p	sure	s+p	sure	s+p
% nonidentical	30.3	32.0	29.2	37.8	44.7	65.2
GIZA++ F_{1n}%	27.6	28.1	41.6	37.8	**53.9**	**48.3**
Meteor F_{1n}%	30.5	31.8	34.7	29.5	33.8	20.6
token/phrase F_{1n}%	**71.2**	**70.5**	**64.8**	**59.7**	46.1	42.6

The same pattern as the identical alignment can be observed. On MSR06 and Edinburgh++, jacana-align had F_{1n}'s around 60% ∼ 70% for nonidentical alignment, which consists of 30% ∼ 40% total alignment in these datasets. We think this performance is reasonable given the amount of external lexical resources jacana-align employs. GIZA++ and Meteor on the other hand could barely get to 40%. However, again, on MTReference, GIZA++ had F_{1n}'s around 50%, with jacana-align closely followed while Meteor still had a hard problem of coping with nonidentical alignment.

4.4.5. Evaluation: Token vs. Phrasal

Another way to investigate what aligners have learned from the data is to divide the scores by (easy) 1x1 and (hard) non-1x1 alignment. Table 4.9 shows the divided scores,

System	Corpus	P%	R%	F1%	E%	Corpus	P%	R%	F1%	E%
		P_i/P_n	R_i/R_n	F_{1i}/F_{1n}			P_i/P_n	R_i/R_n	F_{1i}/F_{1n}	
GIZA++	MSR06 sure (identical: 69.7%)	82.5 93.4/36.1	74.4 89.7/22.4	78.3 91.5/27.6	14.0	MSR06 sure+possible (identical: 68.0%)	84.0 94.5/39.7	72.6 89.4/21.7	77.9 91.9/28.1	13.6
Meteor		82.5 89.9/39.9	81.2 97.3/24.6	81.9 93.5/30.5	15.0		84.5 91.0/45.1	79.5 97.0/24.6	81.9 93.9/31.8	14.5
our token aligner		93.3 96.0/88.0	83.8 95.7/59.7	88.3 95.9/71.2	34.8		94.8 96.7/91.6	81.9 95.2/57.3	87.9 96.0/70.5	31.5
our phrase aligner		91.5 95.9/78.7	83.5 95.8/58.4	87.3 95.8/67.1	32.0		93.4 96.7/84.4	82.1 95.2/57.2	87.4 96.0/68.2	29.1
GIZA++	Edinburgh++ sure (identical: 70.8%)	89.7 97.0/56.3	81.7 96.1/33.0	85.5 96.6/41.6	13.1	Edinburgh++ sure+possible (identical: 62.2%)	93.0 97.9/72.4	74.5 95.9/25.6	82.8 96.9/37.8	10.5
Meteor		88.3 94.0/61.4	80.5 97.8/24.1	84.2 95.9/34.7	12.7		91.0 94.9/77.2	72.9 97.6/18.3	80.9 96.3/29.5	9.8
our token aligner		90.8 96.0/77.8	82.9 97.8/55.6	86.7 96.9/64.8	15.7		90.3 97.0/74.9	79.7 97.8/51.1	84.7 97.4/60.7	11.4
our phrase aligner		90.4 95.8/72.5	81.9 97.6/55.2	85.9 96.7/62.7	13.7		89.4 96.7/71.5	79.4 97.8/51.3	84.1 97.2/59.7	12.1
GIZA++	MTReference sure (identical: 55.3%)	70.1 89.6/45.3	86.3 99.5/66.4	77.4 94.3/53.9	1.6	MTReference sure+possible (identical: 34.8%)	82.6 94.8/64.8	64.7 98.6/38.6	72.6 96.6/48.3	0.9
Meteor		85.1 88.7/63.7	66.7 97.0/23.0	74.8 92.7/33.8	0.8		92.2 93.3/80.8	46.2 95.6/11.8	61.5 94.4/20.6	0.2
our token aligner		86.8 91.4/73.1	69.4 96.0/31.2	77.1 93.7/43.8	1.2		86.2 94.1/69.8	52.7 94.0/22.9	65.4 94.1/34.5	0.2
our phrase aligner		85.7 90.9/70.5	70.6 96.4/34.2	77.4 93.5/46.1	1.9		81.8 93.8/64.3	59.2 95.9/31.9	68.7 94.8/42.6	0.7

Table 4.8.: Results on the 3 datasets with sure and sure+possible alignment, where (x%) indicates how much alignment is identical alignment, such as `New↔New`. E% stands for exact (perfect) match rate. Subscript i stands for corresponding scores for "identical" alignment and n for "non-identical".

with subscript t showing 1x1 "token" alignment scores and p showing non-1x1 "phrasal" scores. The following observation can be made:

jacana-align excelled in 1x1 alignment The following table summaries each aligner's performance on 1x1 alignment:

	MSR06		Edinburgh++		MTReference	
	sure	s+p	sure	s+p	sure	s+p
% 1x1	97.9	96.9	94.7	91.8	87.8	73.3
GIZA++ F_{1t}%	83.2	82.9	90.9	90.4	76.4	73.2
Meteor F_{1t}%	85.9	85.7	91.2	90.1	78.5	74.1
token aligner F_{1t}%	**91.4**	**90.9**	**92.7**	90.6	80.9	71.6
phrase aligner F_{1t}%	90.8	90.6	92.2	**91.1**	**81.7**	**75.2**

jacana-align took the first place in all datasets. Also, the gap between the token aligner and the phrase aligner closed in as the percentage of 1x1 alignment became smaller in the datasets. We think the reason is that when the dataset is mostly ($> 95\%$) about 1x1 alignment, the phrase aligner was "confused" to learn phrasal alignment. But as non-1x1 alignment increased, the phrasal aligner captured the regularities in the data and thus performed better than the token aligner.

Non-1x1 alignment: a big challenge The following table summaries each aligner's performance on non-1x1 "phrasal" alignment:

	MSR06		Edinburgh++		MTReference	
	sure	s+p	sure	s+p	sure	s+p
% non-1x1	2.1	3.1	5.3	8.2	12.2	26.7
GIZA++ F_{1p}%	0.0	0.0	0.0	0.0	**35.0**	**22.9**
Meteor F_{1p}%	**49.4**	**42.3**	11.0	6.3	7.2	3.8
token aligner F_{1p}%	0.0	0.0	16.3	21.6	4.5	4.7
phrase aligner F_{1p}%	13.8	1.7	**40.8**	**30.4**	19.0	17.4

It was surprising to see that Meteor performed the best on the MSR06 dataset. But given that non-1x1 alignment only take 2.1% − 3.1% on MSR06, it is very likely that the paraphrase list of Meteor happened to have included some phrases in MSR06. On the Edinburgh++ dataset, the phrase aligner outperformed all other aligners by a large margin. However, on the MTReference dataset, GIZA++ still had the best scores, with the phrase aligner taking the second place. Overall, alignment quality on non-1x1 blocks was not satisfying: none of the systems went close to 50% in F_1.

4.4.6. Error Analysis

Concrete examples of alignment from GIZA++, Meteor, and the token/phrase version of jacana-align are shown in Appendices A on page 247 and C on page 272. There were three primary categories of error:[12]

1. Paraphrases that are not covered by current lexical resources, such as dubbed↔called, program↔software and signed a contract↔struck a deal.
2. Words that are semantically related judging by the context but not exactly paraphrases, such as married↔wife, beat↔victory, shared↔sentiment among and Cuba↔the island. This is a difficult challenge because the decision to make an alignment depends on a case-by-case basis given the context.
3. Annotation errors or inconsistencies in the datasets.

The last point needs further elaboration. The point of machine learning a statistically trained aligner is to capture the annotation consistences in datasets. In theory using a few annotators consistently across the whole dataset would lead to higher alignment evaluation numbers than using a lot of annotators on a lot of small pieces of the dataset. This is a bias vs. variance trade-off in annotating the datasets. But it also indirectly determines the performance upper bound of a statistically trained aligner.

[12]The jacana-align source code contains a browser in JavaScript (AlignmentBrowser.html) that compares the gold alignment and test output; readers are encouraged to try it out.

System	Corpus	P% P_t/P_p	R% R_t/R_p	F1% F_{1t}/F_{1p}	E%	Corpus	P% P_t/P_p	R% R_t/R_p	F1% F_{1t}/F_{1p}	E%
GIZA++	MSR06 sure (1x1: 97.9%)	82.5 85.7/0.0	74.4 80.9/0.0	78.3 83.2/0.0	14.0	MSR06 sure+possible (1x1: 96.9%)	84.0 87.1/0.0	72.6 79.2/0.0	77.9 82.9/0.0	13.6
Meteor		82.5 85.1/73.8	81.2 86.7/37.1	81.9 85.9/49.4	15.0		84.5 86.5/64.9	79.5 84.9/31.4	81.9 85.7/42.3	14.5
token aligner		93.3 93.8/0.0	83.8 89.2/0.0	88.3 91.4/0.0	34.8		94.8 94.9/0.0	81.9 87.2/0.0	87.9 90.9/0.0	31.5
phrase aligner		91.5 94.2/19.9	83.5 87.6/10.6	87.3 90.8/13.8	32.0		93.4 95.1/3.1	82.1 86.6/1.2	87.4 90.6/1.7	29.1
GIZA++	Edinburgh++ sure (1x1: 94.7%)	89.7 89.4/0.0	81.7 92.5/0.0	85.5 90.9/0.0	13.1	Edinburgh++ sure+possible (1x1: 91.8%)	93.0 90.3/0.0	74.5 90.5/0.0	82.8 90.4/0.0	10.5
Meteor		88.3 90.5/26.2	80.5 91.9/7.0	84.2 91.2/11.0	12.7		91.0 90.8/22.7	72.9 89.3/3.6	80.9 90.1/6.3	9.8
token aligner		90.8 91.7/49.0	82.9 93.7/9.8	86.7 92.7/16.3	15.7		90.3 89.3/45.1	79.7 92.0/14.2	84.7 90.6/21.6	11.4
phrase aligner		90.4 91.8/55.0	81.9 92.6/32.4	85.9 92.2/40.8	13.7		89.4 91.6/43.5	79.4 90.6/23.3	84.1 91.1/30.4	12.1
GIZA++	MTReference sure (1x1: 87.8%)	70.1 65.8/35.1	86.3 91.2/35.0	77.4 76.4/35.0	1.6	MTReference sure+possible (1x1: 73.3%)	82.6 60.4/34.0	64.7 93.1/17.3	72.6 73.2/22.9	0.9
Meteor		85.1 76.0/26.0	66.7 81.2/4.2	74.8 78.5/7.2	0.8		92.2 68.2/25.1	46.2 81.1/2.1	61.5 74.1/3.8	0.2
token aligner		86.8 77.3/39.9	69.4 84.9/2.4	77.1 80.9/4.5	1.2		86.2 61.0/35.4	52.7 86.7/2.5	65.4 71.6/4.7	0.2
phrase aligner		85.7 80.6/44.9	70.6 82.8/12.0	77.4 81.7/19.0	1.9		81.8 69.0/27.8	59.2 82.6/12.6	68.7 75.2/17.4	0.7

Table 4.9.: Results on the 3 datasets with 1x1 and non-1x1 alignment. Subscript t stands for corresponding scores for "token" 1x1 alignment and p for "phrasal" non-1x1 alignment.

4.4.7. Ablation Test and Feature Weights

We have demonstrated that the CRF nature of jacana-align helps with globally optimized alignment over the sentence pairs. The discriminative nature of jacana-align also helps with automatically optimizing over multiple lexical resources. Thus an ablation test of various lexical resources was also conducted: specifically, Wiktionary, WordNet Snow, word2vec, and PPDB, all described in § 4.3.3 on page 156.

Recall that jacana-align's performance in F_1 on MSR06 and Edinburgh++ is around 86% $\sim$ 88%, while on MTReference around 77%, with MTReference a noisier dataset. For real-world usage, we trained the aligner on the full set of MSR06 and Edinburgh++ for it's open-source release. The ablation test was also performed on this mixed dataset, which all together has 2620 manually aligned sentence pairs. We divided them by 80/20 into 2100 training pairs and 520 test pairs. Table 4.10 on the next page shows the ablation test result of the token aligner. Overall, WordNet Snow contributed the most to precision, and PPDB to recall and F_1. However, with all the features enabled, jacana-align's performance was only 1.2% better (85.3% vs. 86.5%). This shows that coverage is still an issue for lexical resources.

Table 4.11 shows a selection of 80 features and their optimized weights. Among all lexical resources, PPDB had the largest weight (2.8348). WordNet and Wiktionary had features spread across the table depending on specific relations. For instance, WiktionaryRelation.SYNONYM had a positive feature weight of 1.3061 while WordnetPartsOf had a negative feature weight of -0.8883. Finally, word2vec had a mediocre weight of 1.0565. This is due to that the notion of "semantic relatedness" in alignment is different than in distributional similarity. For instance, Beijing and Tokyo would be ranked highly similar as they are both capital cities of Asian countries. But they should be mutually exclusive in alignment.

Features	**Precision**	**Recall**	F_1
basic	90.3	80.8	85.3
+Wiktionary	89.7 (-0.6)	80.0 (-0.8)	85.0 (-0.3)
+word2vec	89.8 (-0.5)	81.0 (+0.2)	85.2 (-0.1)
+WordNet Snow	**90.7** (+0.4)	81.3 (+0.5)	85.7 (+0.4)
+PPDB	90.5 (+0.2)	**82.1** (+1.3)	**86.1** (+0.8)
full	90.7	82.6	86.5
-Wiktionary	90.5 (-0.2)	82.4 (-0.2)	86.3 (-0.2)
-word2vec	90.7 (0.0)	82.3 (-0.3)	86.3 (-0.2)
-WordNet Snow	**90.2** (-0.5)	82.2 (-0.4)	86.0 (-0.5)
-PPDB	90.3 (-0.4)	**81.4** (-1.2)	**85.6** (-0.9)

Table 4.10.: Ablation test of the token aligner with various lexical features on 520 test sentences from a mixture of MSR06 and Edinburgh++. Basic features were described in § 4.3.3 on page 156.

4.5. Summary on Alignment

We presented two discriminative models for the task of monolingual alignment, one based on Conditional Random Field for token alignment, the other based on semi-Markov CRF for phrase alignment. The token aligner is based on the model proposed by Blunsom and Cohn (2006). It is only able to make 1-to-1 alignment in a sentence pair. To extend it for phrase alignment, we introduced semi-Markov states and phrasal states to the CRF to make possible phrasal alignment on both the source and target sides. Both aligners access an extensive list of lexical resources and have state-of-the-art performance on several monolingual datasets.

Through a broad-range evaluation of alignment types, we also have found that human annotators can be affected by whether the sentences were pre-aligned by GIZA++. The influence of this pre-alignment directly affects the end performance of several monolingual aligners. Given that GIZA++ was commonly used for word alignment before the age of monolingual alignment, and the high precision characteristics of our monolingual aligners, we recommend in the future the replacement of GIZA++ with jacana-align if pre-alignment is needed for aligning more sentence pairs.

In terms of alignment types, the current state-of-the-art monolingual aligners have excellent performance on identical alignment (96% F_1), moderate performance on non-

align_position.pos2null	5.6821	WordnetDerived	0.6484
identicalMatchIgnoreCase	2.8447	WordnetEntailing	0.6307
pos.map..-.	2.8348	DiceSorensen	0.5789
PPDBsimple	2.8053	pos.map.EX-EX	0.5441
pos.map.VBD-VBD	2.4356	pos.map..-OtherPOS	0.5248
pos.map.DT-null	2.1037	pos.map.CC-OtherPOS	0.5102
pos.match	1.7985	WiktionaryRelation.RELATED_TERM	0.5082
pos.map.PRP-null	1.7352	WiktionaryRelation.ANTONYM	0.5048
pos.map.IN-null	1.6961	similar.left	0.4621
Ngram4	1.4464	pos.map.JJR-JJR	0.4401
WiktionaryRelation.SYNONYM	1.3061	pos.map.TO-TO	0.4258
pos.map.$-$	1.2433	WordnetHyponym	0.3923
WordnetHaveSubstance	1.2189	pos.map.IN-IN	0.3904
pos.map.JJ-null	1.1923	WordnetHypernym	0.3401
word2vec	1.0565	Ngram3	0.2823
pos.map.-LRB—LRB-	1.0486	JaroWinkler	0.2738
WordnetSubstancesOf	1.0204	pos.map.”-null	0.2704
WiktionaryRelation.MERONYM	1.0089	match.right	0.2591
start.2null	0.9967	numCommonSuffix	0.2395
end.2null	0.9967	pos.no_match	0.1805
edge.null2null	0.9967	pos.map.WDT-OtherPOS	0.1736
edge.null2align	0.9967	pos.map.PRP-OtherPOS	0.1499
edge.monotonic.zero	0.9967	Jaccard	0.1498
edge.monotonic.positive	0.9967	similar.functional.right	0.111
edge.monotonic.negative	0.9967	pos.map..-null	0.1071
edge.align2null	0.9967	Levenshtein	0.0148
WiktionaryRelation.COORDINATE_TERM	0.9148	Hamming	0.0127
WordnetCausing	0.9095	align_position.pos2pos	-0.0846
pos.map.MD-MD	0.9017	WordnetLemmaMatch	-0.0923
pos.map.JJS-OtherPOS	0.9013	WiktionaryRelation.HYPONYM	-0.2125
WordnetHavePart	0.8923	pos.map.DT-DT	-0.2317
pos.map.WDT-WDT	0.8815	identicalMatch	-0.2793
similar.functional.left	0.8714	WordnetHaveMember	-0.4198
WiktionaryRelation.SEE_ALSO	0.8699	nicknames	-0.4224
pos.map.-null	0.7855	pos.map.VBD-OtherPOS	-0.5229
match.functional.right	0.7468	match.functional.left	-0.6493
pos.map.JJR-OtherPOS	0.7439	WordnetMembersOf	-0.6698
match.left	0.7302	WordnetPartsOf	-0.8883
WordnetSynonym	0.7086	WiktionaryRelation.DERIVED_TERM	-1.1102
similar.right	0.7028	align_position.pos2pos.relative	-4.9388

Table 4.11.: Selected alignment features and their optimized weights. There were 190 features in total. This table presents 80 of them, with the other 110 features mostly based on “pos.map.*”.

identical alignment (60% $\sim$ 70% F_1), good performance on token alignment (90% F_1) and very poor performance on phrasal alignment (40% F_1 or less). All three datasets we have used do not contain information about natural phrase boundaries. Thus if in the future should more annotated alignment dataset be created, it is suggested to make a focus on annotating more phrase alignment.

So far we have only evaluated the aligners' performance intrinsically. Next we apply them in an NLP application: Question Answering.

4.6. QA with Alignment

4.6.1. Motivation

One fundamental problem in the task of Question Answering (QA) is first knowing whether a retrieved sentence contains the answer to the question, then extracting it. A good hint to approaching this is to judge whether the retrieved sentence is relevant (via lexical overlap, paraphrasing, entailing, etc) to the question, through some mechanism of mapping the QA pairs, such as synchronous parsing, parse tree matching or alignment.

We focus on how natural language alignment can help the task of QA. Intuitively (factoid) QA can be treated as a natural language alignment task, with the objective of aligning the question word with the answer fragment, provided that the factoid answer is usually a short phrase in the sentence. For more open-ended QA, such as those seeking a paragraph describing an opinion, a process, etc, alignment should still help answering the question by drawing the connection between the missing information in the question and the existing information in the text. In this section, instead of directly aligning the question word with answer candidates (i.e., QA *as* alignment), we take a more general approach to incorporate alignment-based features into existing QA engines (i.e., QA *with* alignment), making use of both the power of existing QA engines and natural language aligners.

Due to recent development of monolingual alignment (MacCartney et al., 2008, Thadani and McKeown, 2011, Denkowski and Lavie, 2011, Yao et al., 2013b), many off-the-shelf aligners have been made available for NLP usage. We selected one open-source bilingual aligner and three monolingual ones for this task:

1. GIZA++ (Och and Ney, 2003), adapted from Machine Translation (MT) word alignment.
2. TED, based on Tree Edit Distance (Zhang and Shasha, 1989), described in detail in § 3.2 on page 90.
3. Meteor (Denkowski and Lavie, 2011), used for MT evaluation.
4. jacana-align (Yao et al., 2013b), a discriminative aligner trained on labeled word alignment data (§ 4.3 on page 150).

Note that the first aligner, GIZA++, was originally designed for bilingual word alignment but here adapted to the monolingual case, a common approach (Quirk et al., 2004, Fader et al., 2013, Xu et al., 2013) prior to the introduction of monolingual aligners.

We have already shown in Table 4.7 on page 170 the comparison of alignment performance among these four aligners. In general jacana-align achieved the best results on two different datasets among the four aligners, mainly due to the fact that it was trained on these datasets and was able to optimize over other lexical resources.

Next we conduct extensive experiments on multiple QA datasets, collected from previous QA challenges, i.e., TREC QA and the Jeopardy! game show. The Jeopardy data is of slightly different nature than the TREC data: the questions (called "clues" in the game show) are formed in declarative sentences rather than interrogative; the questions are also usually longer, harder and more realistic since they are designed to test humans as opposed to machines in the TREC QA. Nonetheless, we report consistent findings from our experiments on both datasets.

4.6.2. Using Alignment

All experiments were run with jacana-qa (see Chapter 3 for a detailed introduction) with the full feature set. The way the other three aligners (GIZA++, Meteor, jacana-align) were applied was to replace the original TED aligner that comes with jacana-qa with each of them, then draw alignment features (§ 3.3.2 on page 99) from these new alignments. Final QA performance will be affected if there is a significant difference in alignment quality.

Figure 4.3 shows an example. The sentence pair:

- Question (target): Who was President Cleveland's wife?
- Sentence (source): Cleveland married Frances Folsom in 1886.

can be aligned by any aligners that accept string input. Then the dependency trees of the source and target sentences can be obtained via parsing. Previously the TED aligner gives an edit script that transforms the source *tree* to the target *tree*. Now with the two strings first aligned and then parsed, we can still retrieve such an edit script by walking from the source tree in a bottom-up post-order manner, just like how the TED works, or even more simply retrieve the edit script that transforms the source *string* to the target *string*, with its tree structure encoded:

1. INS_LEAF (Who/WP/dep)
2. INS (was/VBD/root)
3. INS_LEAF (President/NNP/nn)
4. REN_DEP (Cleveland/NNP/nsubj, Cleveland/NNP/poss)
5. INS_LEAF ('s/POS/possessive)
6. REN_POS_DEP (married/VBD/root, wife/NN/nsubj)
7. DEL_LEAF (Frances/NNP/nn)
8. DEL (Folsom/NNP/dobj)
9. DEL (in/IN/prep)
10. DEL_LEAF (1886/CD/pobj)

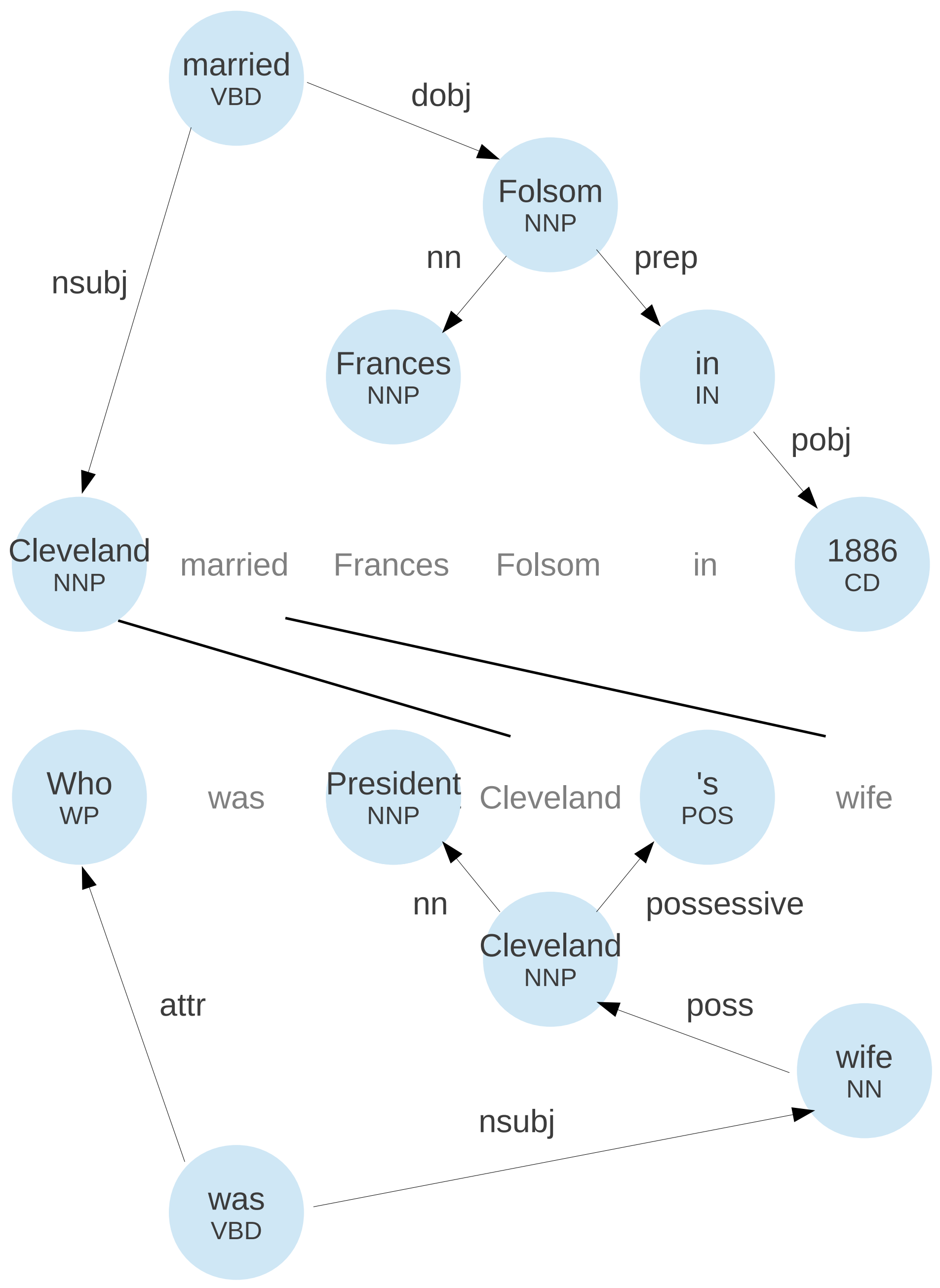


Figure 4.3.: An alignment example of a QA pair and the pair's dependency parse trees. The aligner can accept flat string input. By parsing the sentences post alignment, we can derive features based on each node's dependency labels and alignment status.

Here the operations are insertion, deletion, or substitution (rename). When two tokens with different POS tags or dependency relations are aligned, it is treated as a substitution edit to rename the different POS tag or dependency relation, as in REN_DEP (Cleveland/NNP/nsubj, Cleveland/NNP/poss).

In this way we can extract edit scripts from any aligner's output and post-alignment parsing trees. With some edit type defined for each token, we extract alignment based features defined in Table 3.4 on page 101. jacana-qa further optimizes these features and performs answer extraction. Experiments in the next two sections focus on how these alignment features from various aligners affect the final QA performance.

4.6.3. QA on TREC

We re-used the TREC dataset described in Table 3.2 on page 96 by merging the DEV and TEST part into a bigger TEST, with new statistics shown in Table 4.12.

The final evaluation metrics are traditional Precision, Recall, and F_1 values, shown in Table 4.13. The baseline system only enabled the chunking-like and question type features. Then alignment based features were added to the baseline with different aligners. Table 4.13 shows that alignment added by GIZA++ did not help QA performance. A hand check on the alignment output showed too many misalignments. We think this is mostly due to that the whole data for GIZA++ was too small to draw statistical patterns of co-occurrence. In the later section we report numbers on a much bigger training set. The three monolingual aligners, TED, Meteor and jacana-align, all helped with F_1 significantly.

Overall, adding alignment features help F_1 by about 10% relatively (47.0% vs. 51.5%). However, there is no major difference in terms of final F_1 among different monolingual aligners. Note that this comparison is completely fair: the QA engine used exactly the same set of features, just from different aligners. The result comparison seems to suggest that in general alignment does help QA performance but it does not matter which monolingual aligner is used. Since the test set is not large enough to show a significant performance difference, we move to experiment on a test set 100 times bigger.

set	**source**	**#question**	**#sents**	**%positive**
TRAIN	TREC8-12	1229	53417	12.0
TEST	TREC13	171	2665	19.0

Table 4.12.: TREC dataset distribution. %positive marks the percentage of sentences containing an answer.

features	$P\%$	$R\%$	$F_1\%$
baseline	57.9	39.5	47.0
with GIZA++, INTERSECTION	54.6	38.9	45.5
with GIZA++, GROW-DIAG-FINAL	54.2	38.3	44.9
with GIZA++, UNION	55.2	38.3	45.2
with TED aligner	**60.5**	**44.9**	**51.5**
with Meteor	58.9	43.7	50.2
with jacana-align	59.1	**44.9**	51.0

Table 4.13.: Performance of jacana-qa with alignment features from different aligners. The baseline system did not use any alignment features.

4.6.4. QA on Jeopardy!

4.6.4.1. Data Preparation

Although IBM has not released the Jeopardy data that was used with Watson, we have created a novel dataset that replicates the data as closely as possible. We crawled the J! Archive[13] website and downloaded 237,367 Jeopardy questions with standard answers (each answer is a word or short phrase). The Jeopardy questions (called "clues" in the game) are all formed in declarative sentences where the question focus word is not as obvious as those in the TREC questions. For instance, one question is `Peaches are more than 80% this compound` and the standard answer is `water` (or `What is water`? in the actual game). We wrote some simple rules based on POS tags and keywords, most of which just detect the pronouns in the sentence, to extract the focus word in each

[13]`http://j-archive.com/`

question. For instance, `compound` is the focus word from `Peaches are more than 80% this compound`. In this way we collected 92,354 questions with a focus word found. This counts as roughly 1/3 of total questions. We did not further collect the other 2/3 since it is more time-consuming and specific question analysis is needed to have a full coverage. During feature extraction we used these Jeopardy-style focus words as the question types.

For each of the 92,354 questions, we queried a popular search engine and collected the top 10 snippets (each snippet is roughly 150 words long) and ruled out those coming from Jeopardy-related websites, which usually disclose the answers. This constructs the final training and evaluation dataset. A simple word matching from the standard Jeopardy answer against the snippet was performed to identify positive snippet. Overall, 72,197 (79%) of the questions had a match. This was only to check how well the search engine returned with results. We still used all questions for training and testing and randomly split them into TRAIN (80%, 73,629) and TEST (20%, 18,725). All questions and snippets were processed with Stanford CoreNLP for POS tagging, named entity recognition and dependency parsing. For alignment, we applied TED, Meteor and jacana-align directly. GIZA++ was trained on the whole set.

4.6.4.2. Training and Decoding

On average, for each token in the snippet, there were 250 different features extracted (or fired). The total number of feature *types* extracted from TRAIN was about 35 million. This posted a relatively large-scale machine learning problem. To effectively reduce the dimensionality of the feature space, we applied L1-regularization during CRF training, which encourages drawing feature weight towards zero. It took about 2 days to run CRF-suite (written in C++) on a single core for 300 iterations. We did not find a significant difference in final evaluation when training for a longer time (say, 1000 iterations). After training, the total number of features with learned non-zero weight reduced to about 600 thousand (300 iterations) to 500 thousand (1,000 iterations), roughly a 50-to-70-fold reduction.

Decoding was fast even with 600 thousands of non-zero-weight features (recall that only 250 features fired per token): to tag answers from all 10 snippets (or $1,500$ tokens) for one question, CRFsuite took 0.4 second (note that this process can be easily parallelized to reduce the decoding time by 10 times). Overall, the speed bottleneck during test was the Stanford CoreNLP pipeline (POS tagging + NER + parsing) running at about 25 tokens per second.

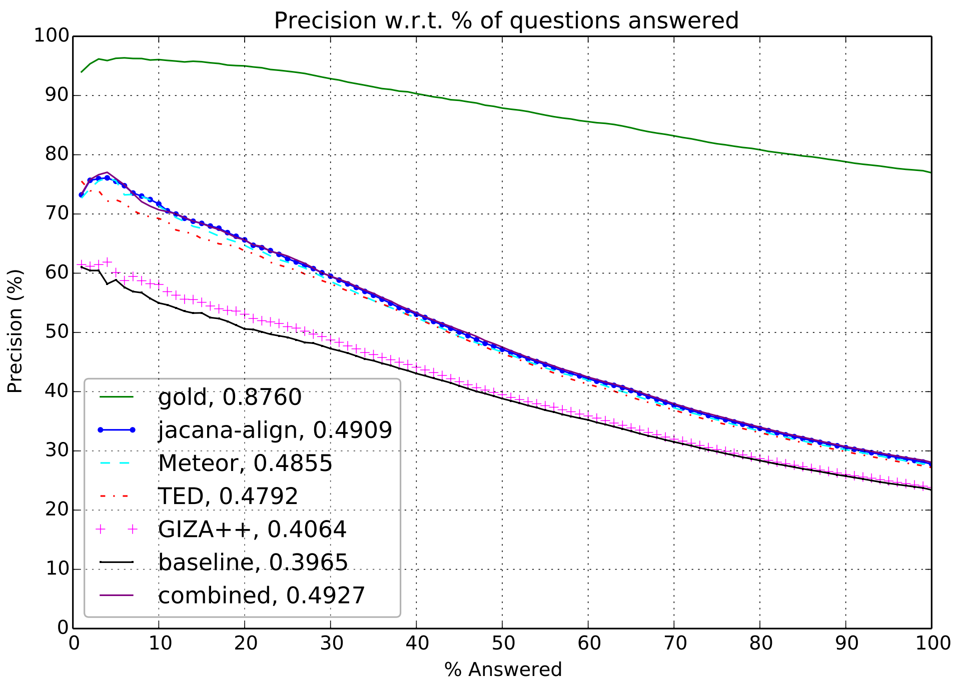
4.6.4.3. Evaluation

Given the great amount of test set, a single report of $P/R/F_1$ scores does not depict the whole picture of QA performance. Thus we resort to a 2-dimensional precision@proportion curve, which shows the precision of a QA engine on any proportions of the whole test set. The curve basically answers one question: at a fixed precision, what is the proportion of questions that can be answered? A better system should be able to answer more questions correctly with the same precision.

The curve is drawn in the following way. For each question, we select the best answer candidate with the highest confidence score. Then for the whole test set, we have a list of (question, highest ranked answer, confidence score) tuples. Say the largest confidence score of all tuples is MAX and the smallest MIN. Running a threshold from MAX down to MIN, we select those questions with an answer confidence score above this threshold and compute the precision at this point. The X axis indicates the percentage of questions above this threshold and the Y axis the precision, shown in Figure 4.4(a). We also computed the area under the curve (or average precision) to show a precise number for each system.

The baseline system, with no alignment features enabled, had an average precision of roughly 40%. GIZA++ (with the intersection heuristics) only helped marginally (but still significant, explained later) over the baseline. However, there is a clear jump with the three monolingual aligners. The top performing aligner, jacana-align, almost helped boost the precision to 50%, a 25% relative gain from the baseline.

Due to the large number of test instances, even a difference of 0.5% could be significant. Thus we used the paired permutation test (Smucker et al., 2007) on the close competitors,



(a) precision at % answered

null hypothesis	p
jacana-align vs. Meteor	0.039
jacana-align vs. TED	0.006
jacana-align vs. baseline	0.000
Meteor vs. TED	0.230
GIZA++ vs. baseline	0.000
combined vs. jacana-align	0.007

(b) paired permutation significance test

Figure 4.4.: Performance of jacana-qa in terms of precision at various proportion of attempted questions. The baseline system had no alignment features enabled. The gold system assumed oracle answer extraction with respect to the proportion of questions answered by the best system (blue curve). The float numbers in the legend are average precision (or area under the curve). The "combined" curve uses features from the baseline system and all aligners.

with p-values shown in Figure 4.4(b). The curve of GIZA++ constantly lies above the baseline curve and the area under the curve is about 1% better: permutation test shows that this difference is significant ($p = 0.000$). In the top tier, the differences between jacana-align and Meteor (0.4909 vs. 0.4855) and between Meteor and TED (0.4855 vs. 0.4792) are 0.54% and 0.63% separately. Interestingly, the former difference is significant ($p = 0.039$, close to 0.05 though) but the latter is not ($p = 0.23$).

A close examination shows that in terms of precision *at* each proportion, the difference between jacana-align and Meteor is actually larger than that between Meteor and TED. The graph however shows the precision *up to* each proportion. It just happened that the system with Meteor answered more instances correctly between 0% and 10% of answers than with TED, then due to this early advantage, the average precision *up to* other proportion of questions is also better than TED. To put it in other words: the system with jacana-align is better than that with Meteor at almost all proportions of questions, while the system with Meteor is only mostly better than that with TED at the first 10% of all questions. That explains why the former difference was significant while the latter was not. We show the difference between jacana-align and Meteor with concrete examples in Appendix B on page 260.

We were also interested to see whether the aligners made orthogonal decisions in the types of alignment that could lead to better QA performance. Thus we combined all alignment features from different aligners, plus the baseline features, and trained a "combined" model with the union of all features. This combined model outperformed the single best result from jacana-align by only 0.18% (0.4927 vs. 0.4909). However, significance test showed that the difference was significant ($p = 0.007$): the combined model covered not only the correct cases jacana-align was right about, but also some additional questions.

Note that since there are only answers for 79% of all questions in the retrieved top 10 snippets, for the other 21% of questions, no matter how hard the QA system tries, the answer is never correct. Thus we also drew the "ceiling" for this curve in green with respect to the confidence score from the best performing curve in blue (jacana-align). For instance, the blue point at $(30\%, 60\%)$ shows that the system is able to answer 30%

of all questions with 60% precision at some confidence threshold. However, among all the 30% of questions answered by the system, 7% of them do not have a correct answer retrieved in the snippets. Thus assuming perfect answer extraction, the best a system can do is 93% precision. Thus the green line lies at $(30\%, 93\%)$ at the same proportion value. Clearly, there is still large room for improvement.

4.6.5. Discussion

Figure 4.4 shows a major difference between QA systems without alignment, and with alignment from bilingual aligners and monolingual aligners. We were surprised to see that GIZA++ did not help much. The reason could be two-fold: firstly, not enough training data. However, the total number of words in the parallel training corpus exceeded 100 million: we think this is unlikely the reason. Secondly, not good-enough accuracy. QA is a task that requires highly precise linguistic analysis. Any miss-or-over-alignment could affect the final end performance easily. Our manual check also showed that the output from GIZA++ was too noisy. The monolingual aligners, on the other hand, provided mostly accurate alignment, with some cases of lower coverage. Still, they significantly helped QA. Overall, we reached the conclusion that monolingual aligners are much better fit for the task of question answering.

Among the monolingual aligners, the difference in average precision was only between $0.5\% \sim 1.5\%$. However, when used in real-world applications, they had their distinctions. The TED aligner is the fastest (roughly $10,000$ alignment per second) if we do not count parsing time. It does not only require parsing, but also tree-structured input due to constraints of the Zhang and Shasha (1989) algorithm. This has implications that the collapsed graph form of Stanford dependency (De Marneffe and Manning, 2008) could not be used (note that both Meteor and jacana-align do not require parsing). Finally, it only uses WordNet for aligning synonyms, hypernyms, etc. Since the system is not statistically trained, it needs very high quality lexical resources if going beyond WordNet.

The Meteor system shows both the merits of performance and speed. It includes a larger lexical resource for paraphrases (than TED) and runs in the speed of roughly 100 alignments per second. Finally, jacana-align is the most accurate but also slowest (10 alignments per second). Training on manually aligned data of high quality helped a precision task like QA the most. For instance, manual check showed that Meteor sometimes aligned just the stopwords in QA pairs, which might give some false information in the form of alignment features in the final answer extraction decision. jacana-align on the other hand tended not to do so when there was no contextual evidence to support this kind of stopwords alignment during CRF decoding. Finally, despite the minor difference in terms of precision for QA among the three aligners, when applied in large scale (such as in the Jeopardy dataset), a better precision of only 0.5% more can lead to answering 100 more questions correctly.

A hand check over all learned features show some useful patterns of alignment-based features. For instance, one feature fired at a deleted direct object carried a large positive weight with respect to the B-ANS class, indicating that the token could be an answer. Some other features fired at deleted adverbs or punctuations carried a large positive weight with respect to the O class, indicating that these tokens are not likely to be answers. Interestingly, of all 600 thousand non-zero features learned from training, only about $1,500$ (0.25%) of them were alignment features. These less popular but effective features, however, boosted the final QA performance by 25% relatively.

4.6.6. Summary

In this section we have systematically examined the relationship between question answering and natural language alignment, with evidence from experimenting on two popular QA challenges: TREC and the Jeopardy! game show. We found that high-precision aligners help QA the most, since QA is also a task that requires highly precise linguistic analysis. When applied in QA, monolingual aligners are much more helpful than the bilingual aligner (specifically, GIZA++) we used: they have all shown good performance and helped boosting the average precision by up to 25% relatively, even though the total number of alignment features consisted only 0.25% of all features. This means the automatically generated and optimized alignment features took an important role in helping identify the correct answers. Overall, with the development of monolingual aligners in recent years, natural language (esp. monolingual) alignment has matured, in theory, precision, and speed, into a worthy component in the task of question answering.

4.7. Discussion: Various Aligners

We have employed and evaluated four open-source aligners in this chapter: GIZA++, TED, Meteor, jacana-align. Table 4.14 gives a brief summary and comparison of each of them. Specifically, my experience from using each of them is:

- GIZA++: even though commonly used as a tool for bilingual word alignment, GIZA++ is actually a tool set for employing the EM algorithm to performing unsupervised learning of the IBM and HMM models for any input pairs. It is not sensitive to input and can be simply used as a handy tool for computing conditional probabilities. The input can be any form as long as there are co-occurring regularities. However, on the other hand, GIZA++ is not equipped with any linguistic knowledge. Thus when used for monolingual alignment, one has to prepare the training corpus with great care. One simple example is that identical words have to appear in parallel in the training corpus for GIZA++ to memorize them. Still, GIZA++ lacks the ability to recognizing unknown words (even if they are identical) and making use of more lexical resources. Our analysis in § 4.4.3.2 on page 167 shows that on human annotated datasets without being pre-aligned by GIZA++, the performance difference in terms of F_1 values is about 10% between GIZA++ and the best monolingual aligner.

- TED: the Tree Edit Distance algorithm is a dynamic algorithm for aligning a pair of tree input. It runs very fast when the input is pre-parsed: in my experiments it aligns 10 thousand tree pairs a second. It has limited ability to incorporate lexical resources and requires parsing. Thus it is only suggested to be used when input is parsed and alignment speed is a top priority.

- Meteor: Meteor is specifically designed for aligning multiple references in machine translation, with a built-in paraphrase list. It is somehow rule-based: the objective is to minimize cross alignment. But this objective is not the best policy for aligning

aligner	model	input structure	lexical resources	application
GIZA++	IBM/HMM models	flat	none	build co-occurrence
TED	minimal edit distance	tree	WordNet	tree alignment
Meteor	minimal cross align	flat	paraphrases	fast mono align
jacana-align	Conditional Random Field	flat	various	precise mono align

Table 4.14.: Summary and comparison of the four open-source aligners used in this Chapter.

English sentence pairs. I show in Appendix B on page 260 a few examples of Meteor failing to align to the correct stop words. Often Meteor tends to over align: given a pair of irrelevant sentences, Meteor usually makes some alignment between punctuations and stop words. This is OK in the machine translation setting: it is assumed that the output of an MT system is relevant to the reference to some degree. But it is not applicable when using Meteor in other monolingual alignment tasks where this assumption does not hold. Meteor is fast, aligning up to a hundred sentence pairs a second.

- jacana-align: it is specifically designed for monolingual alignment, with access to various lexical resources. It easily recognizes identical word alignment and its global decoding nature tries to assign the best alignment sequence in case of multiple ambiguous alignment. jacana-align is a high-precision aligner, suitable in tasks that require high precision, such as question answering. Currently jacana-align is designed for aligning the English language. It can be adapted to aligning closely-related languages. jacana-align aligns a few to a dozen sentences per second.

4.8. Conclusion

This chapter is based on the following two published papers:

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch and Peter Clark. *A Lightweight and High Performance Monolingual Word Aligner.* ACL Short. 2013.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch and Peter Clark. *Semi-Markov Phrase-based Monolingual Alignment.* EMNLP. 2013.

The main ideas and scientific contributions are:

The first open-source and state-of-the-art monolingual word aligner. The traditional way to aligning monolingual parallel sentences (e.g., Casey loves Kim $\leftrightarrow$ Casey is fond of Kim) is using GIZA++ (Och and Ney, 2003), the most used bilingual aligner. However, bilingual aligners do not suit the task of monolingual alignment for two major reasons:

1. Bilingual aligners need a lot of parallel sentences to train on. Bilingual parallel sentences commonly exist in translations of multilingual documents (especially those from government documents and bilingual news papers). But there are not as many for the monolingual case.
2. There are a lot of monolingual lexical resources that can help alignment, such as thesaurus, WordNet, distributional similarity measures. Bilingual aligners are not designed to incorporate these extra resources.

All these call for a discriminatively trained monolingual aligner. We adopted the idea of Blunsom and Cohn (2006), and trained a CRF-based aligner for English sentences. We defined features based on string similarities, contextual matching, relative positions of tokens, POS tag matching, and WordNet relations etc. Training and testing on a word-aligned corpus (Brockett, 2007) of 1600 sentences shows superior performance over other strong baselines from GIZA++, tree edit distance, and the perceptron-based MANLI aligners (MacCartney et al., 2008, Thadani and McKeown, 2011).

The token aligner only handles one-to-one alignment. It does not work for phrases, idioms, or multi-word expressions. For instance, it is very hard for the token aligner to align pass away ↔ kick the bucket. Thus on top of the token aligner, we continue to develop the phrase-based aligner.

The first to use semi-Markov CRF models for phrase-based alignment. To improve the one-to-one token aligner to phrase-based many-to-many aligner, we need to be able to align to phrases both on the source and target side. On the target side, we merge consecutive states together to get phrasal states. For instance, the words pass away represent two separate *token states* in the token aligner, but in the phrase aligner, we merge them to get a *phrasal state*. Any words align to this merged state indicates a phrase alignment *on the target side*.

On the source side, we break the Markovian property of a CRF and make one state last more than one time step (thus the current state does not necessarily depend on the previous state). In this case, during the multiple time steps where one state aligns to multiple tokens on the source side, we have achieved phrasal alignment *on the source side*. The regular CRF has also been generalized to a semi-Markov CRF. To be more clear, the difference between CRF and the semi-Markov CRF is that in the CRF the current state depends on the previous state one time step ago, while in the semi-Markov CRF the current state depends on the state l_s time steps ago, where l_s is the phrase length on the source side. Semi-CRF models are able to do phrase-based alignment with a sacrifice of speed.

The first to systematically justify monolingual alignment in question answering. There are two aspects in the above statement. First, monolingual alignment as a separate NLP task has been justified in natural language inference (MacCartney, 2009) but not formally in question answering. Second, researchers have more or less used alignment (with either monolingual or bilingual tool) in QA but never stated how monolingual alignment alone helps QA. In this chapter we have developed our own theory for monolingual alignment, and conducted systematic experiments on two alignment datasets and two QA datasets to show that: it is desirable to develop a separate aligner for monolingual data and monolingual aligners show consistent superior performance in both alignment accuracies and end QA performance. Thus the task of monolingual alignment in question answering has been justified.

Conclusion and Future Work We have introduced two state-the-art monolingual aligners (jacana-align) in this chapter. Experiments show that these aligners have better performance than traditional bilingual aligners and other monolingual aligners on two annotated word-aligned corpora. With both theory and dataset matured, monolingual alignment has gradually developed into its own NLP task. It has been so far shown effective in the task of recognizing textual entailment (MacCartney, 2009). We further showed that our aligner also helps the most among all other aligners in final QA performance on two datasets (TREC and Jeopardy!). Due to its ease of usage and effectiveness, we strongly encourage a QA system to have linguistic features based on alignment between the question and snippet, beyond other traditional features such as named entity labels.

Looking forward from the current stage of monolingual alignment, I can imagine future work in the following directions:

- Recognizing contradictions in sentences. Currently we have only focused on recognizing word pairs that are of similar meaning, such as synonyms, paraphrases, hypernyms. But we also want to recognize mutually exclusive things in a sentence pair. To use one example given by Dekang Lin, when searching emperor of China, a search engine might return snippets about emperor of Japan. A naive aligner would align the emperor part together, but a better aligner would tell that they should really not align since they are about different things.

- Going further beyond lexical similarities. For identical word alignment, we are able to get around 97% in F_1 but for non-identical alignment, we have trouble getting over 50% (c.f. Table 4.8 on page 173). Non-identical alignment is the most difficult problem. But they are essential in NLP applications: we have shown that the jacana-qa system, when equipped with the jacana-align aligner, has the best performance than with other aligners (TED and Meteor) that have very limited ability to incorporate lexical resources. How can we then get past the 50% F_1 for non-identical alignments after exploiting as many lexical resources as possible?

This is still an open question.

- Recreating the natural language logic (NatLog) system with jacana-align as the backbone. The MANLI system was created for the work of recognizing textual entailment in the more general framework of natural language logic (MacCartney, 2009). Very detailed linguistic rules are defined over types of alignments and these rules are shown very precise in recognizing inference. Unfortunately neither MANLI nor NatLog is open-source. Now that we have created an aligner that is even more accurate than MANLI, it is promising to develop a natural language inference engine on top of the aligner. I imagine it with great implication in a lot of AI tasks that require some level of reasoning.

- Using monolingual alignment in other NLP tasks. Alignment has been used in question answering and recognizing textual entailment. What about other areas? Here is a list of tasks I think applicable:

  – Tweet normalization. With some tweets-dependent features defined, it can be used to align deliberately shortened or scrambled tweets to normal text, for instance outa biz $\leftrightarrow$ out of business.

  – Information retrieval. Alignment tells how closely related two sentences are. Potentially it can be used by a search engine to rank the retrieved snippets with respect to the query.

  – Entity linking. We want to find out the same events described in multiple sources, very commonly used in news aggregation, summarization; or we want to find out all lifetime events about the same person over various sources. A lot of these work falls into the category of entity linking with predicate-argument alignment. I have also co-authored a short paper with other researchers in this area, see Wolfe et al. (2013).

  – Interface to database, or textual schema matching. When using natural language to query a database, such as the Internet Movie Database (IMDb), there is a mismatch between spoken words and database relations, such as played in $\leftrightarrow$ /film/casting. We can view it as a monolingual alignment task: aligning natural language words with database schemas in the same language. I will introduce a solution in the next chapter using web-scale data mining.

5. Feature-driven QA from Structured Data: Freebase

In this chapter we apply the same *feature-driven* idea on a different source: structured knowledge base. Factoid question answering relies on answer extraction, but the source that provides answer candidates could either be structured (knowledge base) or unstructured (text). I have shown in previous chapters that the feature-driven technology was successfully used in answer extraction from free text. Now I show that it also achieves state-of-the-art performance in answer extraction from Freebase. With the heated interest in question answering from knowledge bases in the NLP community, we provide a novel technique based on information extraction and quite surprisingly, this technique outperforms (in terms of macro F_1) semantic parsing approaches while using less heavy machinery. I give an overview of the approach in § 5.2 with background explained (§ 5.3) and automatic feature coupling in § 5.4. The alignment problem between natural language words and knowledge base relations is tackled in § 5.5. Finally the experiments (§ 5.6) and discussion (§ 5.7) compare the performance between our approach and semantic parsing. The successful application of QA from Freebase completes the idea of feature-driven technologies I have been trying to communicate in this dissertation. The accompanying implementation is jacana-freebase.

5.1. Introduction

Question Answering (QA) from a knowledge base (KB) has a long history within natural language processing, going back to the 1960s and 1970s, with systems such as

Baseball (Green et al., 1961) and Lunar (Woods, 1977). These systems were limited to closed-domains due to a lack of knowledge resources, computing power, and ability to robustly understand natural language. With the recent growth in KBs such as `DBPedia` (Auer et al., 2007), `Freebase` (Bollacker et al., 2008) and `Yago2` (Hoffart et al., 2011), it has become more practical to consider answering questions across wider domains, with commercial systems including `Google Now`, based on Google's `Knowledge Graph`, and `Facebook Graph Search`, based on social network connections.

The AI community has tended to approach this problem with a focus on first understanding the intent of the question, via shallow or deep forms of semantic parsing (c.f. § 5.3 for a discussion). Semantic parsing is the process of converting natural language sentences into meaning representations. Assuming appropriate question analysis, such systems are then faced with forward searching from the question representation to a match within potentially large KBs (for example, Freebase contains more than two-billion facts). Typically questions are converted into some meaning representation (e.g., the lambda calculus), then mapped to database queries. Performance is thus bounded by the accuracy of the original semantic parsing, and the well-formedness of resultant database queries.[1]

The Information Extraction (IE) community approaches QA differently: first performing relatively coarse information retrieval as a way to triage the set of possible answer candidates, and only then attempting to perform deeper analysis. To some degree it is also how humans find answers: first find relevant information from a search engine, Wikipedia, or textbook, then isolate the precise answer.

Researchers in semantic parsing have recently explored QA over Freebase as a way of moving beyond closed domains such as GeoQuery (Tang and Mooney, 2001). While making semantic parsing more robust is a laudable goal, here we provide a more rigorous IE baseline against which those efforts should be compared: we show that "traditional" IE methodology achieved similar performance in terms of average F_1 and better performance in terms of macro F_1 as compared to Berant et al. (2013), while using a much simpler

[1]As an example, 50% of errors of the CCG-backed (Kwiatkowski et al., 2013) system were contributed by parsing or structural matching failure.

model.

5.2. Approach

We will view a KB as an interlinked collection of "topics". When given a question about one or several topics, we can select a "view" of the KB concerning only involved topics, then inspect every related node within a few hops of relations to the topic node in order to extract the answer. We call such a view a *topic graph* and assume answers can be found within the graph. We aim to maximally automate the answer extraction process, by massively combining discriminative features for both the question and the topic graph. With a high performance learner we have found that a system with millions of features can be trained within hours, leading to intuitive, human interpretable features. For example, we learn that given a question concerning money, such as: what money is used in ukraine, the expected answer type is likely currency. We formalize this approach in §5.4.

One challenge for natural language querying against a KB is the relative informality of queries as compared to the grammar of a KB. For example, for the question: who cheated on celebrity A, answers can be retrieved via the Freebase relation celebrity.infidelity.participant, but the connection between the phrase cheated on and the formal KB relation is not explicit. To alleviate this problem, the best attempt so far is to map from `ReVerb` (Fader et al., 2011) predicate-argument triples to Freebase relation triples (Cai and Yates, 2013a, Berant et al., 2013). Note that to boost precision, `ReVerb` has already pruned down less frequent or credible triples, yielding not as much coverage as its text source, `ClueWeb`. Here we instead directly mine relation mappings from `ClueWeb` and show that both direct relation mapping precision and indirect QA F_1 improve by a large margin. Details in §5.5.

Finally, we tested our system, jacana-freebase, on a realistic dataset generously contributed by Berant et al. (2013), who collected thousands of commonly asked questions by crawling the `Google Suggest` service. Our method achieves state-of-the-art performance in terms of macro and average F_1 over answered questions.

5.3. Background

QA from a KB faces two prominent challenges: model and data. The model challenge involves finding the best meaning representation for the question, converting it into a query and executing the query on the KB. Most work approaches this via the bridge of various intermediate representations, including combinatory categorial grammar (Zettlemoyer and Collins, 2005, 2007, 2009, Kwiatkowski et al., 2010, 2011, 2013), synchronous context-free grammars (Wong and Mooney, 2007), dependency trees (Liang et al., 2011, Berant et al., 2013), string kernels (Kate and Mooney, 2006, Chen and Mooney, 2011), and tree transducers (Jones et al., 2012). These works successfully showed their effectiveness in QA, despite the fact that most of them require hand-labeled logic annotations. More recent research started to minimize this direct supervision by using latent meaning representations (Berant et al., 2013, Kwiatkowski et al., 2013) or distant supervision (Krishnamurthy and Mitchell, 2012). § 2.3.4 on page 60 gives a very detailed description.

We instead attack the problem of QA from a KB from an IE perspective: we learn directly the pattern of QA pairs, represented by the dependency parse of questions and the Freebase structure of answer candidates, without the use of intermediate, general purpose meaning representations.

The data challenge is more formally framed as ontology or (textual) schema matching (Hobbs, 1985, Rahm and Bernstein, 2001, Euzenat and Shvaiko, 2007): matching structure of two ontologies/databases or (in extension) mapping between KB relations and NL text. In terms of the latter, Cai and Yates (2013a) and Berant et al. (2013) applied pattern matching and relation intersection between Freebase relations and predicate-argument triples from the `ReVerb` OpenIE system (Fader et al., 2011). Kwiatkowski et al. (2013) expanded their CCG lexicon with Wiktionary word tags towards more domain independence. Fader et al. (2013) learned question paraphrases from aligning multiple questions with the same answers generated by `WikiAnswers`. The key factor to their success is to have a huge text source. Our work pushes the data challenge to the limit by mining directly from `ClueWeb`, a 5TB collection of web data.

Finally, the KB community has developed other means for QA without semantic parsing (Lopez et al., 2005, Frank et al., 2007, Unger et al., 2012, Yahya et al., 2012, Shekarpour et al., 2013). Most of these work executed SPARQL queries on interlinked data represented by RDF (Resource Description Framework) triples, or simply performed triple matching. Heuristics and manual templates were also commonly used (Chu-Carroll et al., 2012). We propose instead to learn discriminative features from the data with shallow question analysis. The final system captures intuitive patterns of QA pairs automatically.

5.3.1. SEMPRE

The major system of comparison in this chapter is SEMPRE (Semantic Parsing with Execution) by Berant et al. (2013). In this section we give a detailed description.

Given a natural language sentence, SEMPRE directly parses into lambda DCS (Dependency-based Compositional Semantics). Using the example given by Liang (2013):

utterance: people who have lived in Seattle

lambda calculus: $\lambda x.\exists e.\mathsf{PlacesLived}(x, e) \wedge \mathsf{Location}(e, \mathsf{Seattle})$

lambda DCS: PlacesLived.Location.Seattle

Lambda DCS attempts to remove variables explicitly while encoding implicit relation with relations and entities defined by the knowledge base. For instance, one can find out from Freebase that Seattle is a single entity while both Location and PlacesLived are binary relations. The three terms here are joined by the "." joint operator. Other common operators in lambda DCS include intersection, union, negation, count and argmax. For instance:

Intersection

utterance: scientists born in Seattle

lambda DCS: Profession.Scientist $\sqcap$ PlaceOfBirth.Seattle

Union

utterance: Oregon, Washington and Canadian provinces

lambda DCS: Oregon $\sqcup$ Washington $\sqcup$ Type.CanadianProvince

Negation

utterance: states not bordering California

lambda DCS: Type.USState $\sqcap \neg$ Border.California

count

utterance: the number of states in the US

lambda DCS: count(Type.USState)

argmax

utterance: largest state by area

lambda DCS: argmax(Type.USState, Area)

The resulting logic form based on lambda DCS is converted into database queries, such as SPARQL, and executed. How this conversion has been done was neither described in detail in Berant et al. (2013) nor in Liang (2013).

The input to SEMPRE contains only the utterance and the answer to the utterance, no logic forms (also called derivations) are used. The discriminative log-linear model over derivations $d \in D(x)$ given utterances x is defined as:

$$p_\theta(d \mid x) = \frac{\exp\{\phi(x,d)^\intercal \theta\}}{\sum_{d' \in D(x)} \exp\{\phi(x,d')^\intercal \theta\}}$$

where $\phi(x,d)$ is a feature vector and θ is the vector of its parameters to be optimized. Given a set of n QA pairs (x_i, y_i) the objective is to maximize the total log likelihood of the correct answer ($[\![d.z]\!] = y_i$) over all training instances:

$$\mathcal{O}(\theta) = \sum_{i=1}^{n} \log \sum_{d \in D(x) : [\![d.z]\!] = y_i} p_\theta(d \mid x)$$

In order to map from words in utterances to knowledge base entities and relations, alignment was used with features drawn from text similarity, co-occurrence, ReVerb (Fader et al., 2011), etc. In order to connect aligned predicates together to form denotations, bridging was used to generate binary predicates based on neighboring logical predicates. On the annotated WebQuestions dataset, Sempre achieved a 31.4% average F_1, which was later revised to 35.7% by Berant and Liang (2014) via bug fixing.

5.4. Graph Features

Our model is inspired by an intuition on how everyday people search for answers. If you asked someone: what is the name of justin bieber brother,[2] and gave them access to Freebase, that person might first determine that the question is about Justin Bieber (or his brother), go to Justin Bieber's Freebase page, and search for his brother's name. Unfortunately Freebase does not contain an exact relation called brother, but instead sibling. Thus further inference (i.e., brother $\leftrightarrow$ male sibling) has to be made. In the following we describe how we represent this process based on features extracted from both the question and the Freebase graph.

5.4.1. Question Graph

In answering our example query a person might take into consideration multiple constraints. With regards to the question, we know we are looking for the name of a person based on the following:

- the dependency relation nsubj(what, name) and prep_of(name, brother) indicates that the question seeks the information of a name;[3]
- the dependency relation prep_of(name, brother) indicates that the name is about a brother (but we do not know whether it is a person name yet);

[2]All examples used come from the training data crawled from `Google Suggest`. They are lowercased and some contain typos.

[3]We use the Stanford collapsed dependency form.

- the dependency relation nn(brother, bieber) and the facts that, (i) Bieber is a person and (ii) a person's brother should also be a person, indicate that the name is about a person.

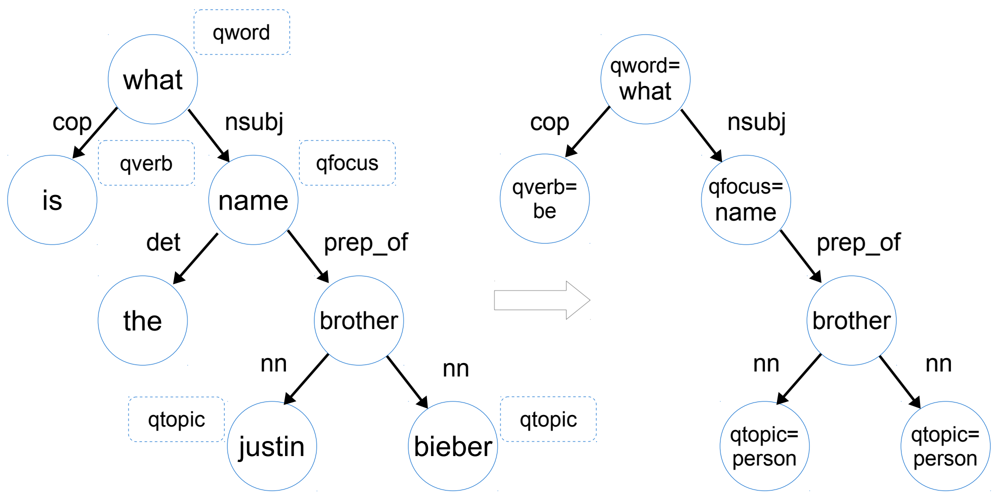
This motivates the design of dependency-based features. We show one example in Figure 5.1(a), left side. The following linguistic information is of interest:

- question word (*qword*), such as what/who/how many. We use a list of 9 common qwords: who, when, what, where, how, which, why, whom, whose.
- question focus (*qfocus*), a cue of expected answer types, such as name/money/time. We keep our analysis simple and do not use a question classifier, but simply extract the noun dependent of qword as qfocus.
- question verb (*qverb*), such as is/play/take, extracted from the main verb of the question. Question verbs are also good hints of answer types. For instance, the verb play is likely to be followed by an instrument, a movie or a sports team.
- question topic (*qtopic*). The topic of the question helps us find relevant Freebase pages. We simply apply a named entity recognizer to find the question topic. Note that there can be more than one topic in the question.

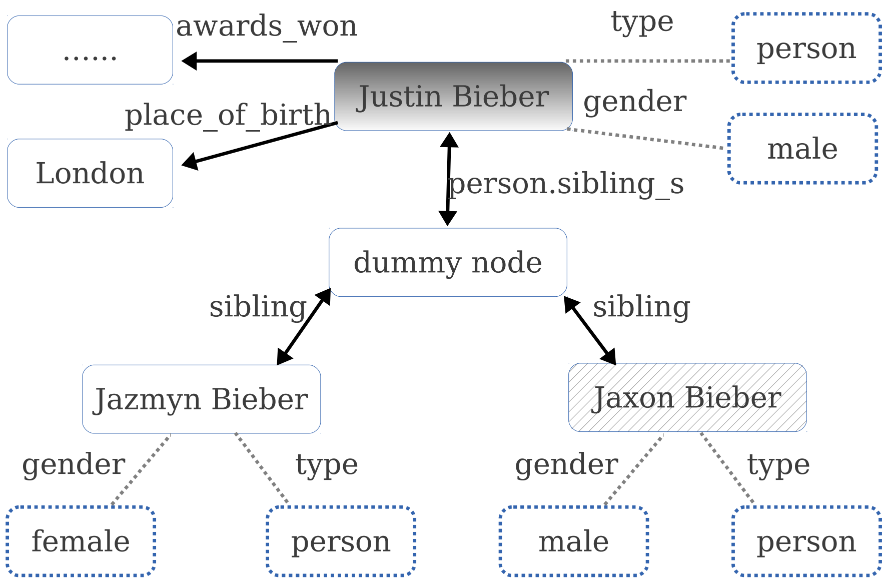
Then we convert the dependency parse into a more generic question graph, in the following steps:

1. if a node was tagged with a question feature, then replace this node with its question feature, e.g., what $\rightarrow$ qword=what;
2. (special case) if a qtopic node was tagged as a named entity, then replace this node with its its named entity form, e.g., bieber $\rightarrow$ qtopic=person;
3. drop any leaf node that is a determiner, preposition or punctuation.

The converted graph is shown in Figure 5.1(a), right side. We call this a *question feature graph*, with every node and relation a potential feature for this question. Then



(a) Dependence parse with annotated question features in dashed boxes (left) and converted feature graph (right) with only relevant and general information about the original question kept. Note that the left is a real but incorrect parse.



(b) A view of Freebase graph on the Justin Bieber topic with nodes in solid boxes and properties in dashed boxes. The hatching node, Jaxon Bieber, is the answer. Freebase uses a dummy parent node for a list of nodes with the same relation.

Figure 5.1.: Dependency parse and excerpted Freebase topic graph on the question what is the name of justin bieber brother.

features are extracted in the following form: with s the source and t the target node, for every edge $e(s,t)$ in the graph, extract s, t, $s \mid t$ and $s \mid e \mid t$ as features. For the edge, prep_of(qfocus=name, brother), this would mean the following features: qfocus=name, brother, qfocus=name|brother, and qfocus=name|prep_of|brother.

We show with examples why these features make sense later in §5.6 Table 5.9. Furthermore, the reason that we have kept some lexical features, such as brother, is that we hope to learn from training a high correlation between brother and some Freebase relations and properties (such as sibling and male) if we do not possess an external resource to help us identify such a correlation.

5.4.2. Freebase Topic Graph

Given a topic, we selectively roll out the Freebase graph by choosing those nodes within a few hops of relationship to the *topic node*, and form a *topic graph*. Besides incoming and/or outgoing relationships, nodes also have *properties*: a string that describes the attribute of a node, for instance, node type, gender or height (for a person). One major difference between relations and properties is that both arguments of a relation are nodes, while only one argument of a property is a node, the other a string. Arguments of relations are usually interconnected, e.g., London can be the place_of_birth for Justin Bieber, or capital_of the UK. Arguments of properties are attributes that are only "attached" to certain nodes and have no outgoing edges. Figure 5.1(b) shows an example.

Both relationship and property of a node are important to identifying the answer. They connect the nodes with the question and describe some unique characteristics. For instance, without the properties type:person and gender:male, we would not have known the node Jaxon Bieber represents a male person. These properties, along with the sibling relationship to the topic node, are important cues for answering the question. Thus for the Freebase graph, we use relations (with directions) and properties as features for each node. Suppose a node has n relations and properties in total, then the total feature number is n (single features) plus $\binom{n}{2}$ (combination of any two single features). More feature combinations can be used if the computation is affordable.

Additionally, we have analyzed how Freebase relations map back to the question. Some of the mapping can be simply detected as paraphrasing or lexical overlap. For example, the person.parents relationship helps answering questions about parenthood. However, most Freebase relations are framed in a way that is not commonly addressed in natural language questions. For instance, for common celebrity gossip questions like who cheated on celebrity A, it is hard for a system to find the Freebase relation celebrity.infidelity.participant as the target relation if it had not observed this pattern in training.

Thus assuming there is an alignment model that is able to tell how likely one relation maps to the original question, we add extra alignment-based features for the incoming and outgoing relation of each node. Specifically, for each relation *rel* in a topic graph, we compute $P(rel \mid question)$ to rank the relations. Finally the ranking (e.g., top 1/2/5/10/100 and beyond) of each relation is used as a feature instead of a pure probability. We describe such an alignment model in § 5.5.

5.4.3. Feature Production

We combine question features and Freebase features (per node) by doing a pairwise concatenation. In this way we hope to capture the association between question patterns and answer nodes. For instance, in a loglinear model setting, we expect to learn a high feature weight for features like:

qfocus=money|node_type=currency

and a very low weight for:

qfocus=money|node_type=person.

This combination greatly enlarges the total number of features, but owing to progress in large-scale machine learning such feature spaces are less of a concern than they once were (concrete numbers in § 5.6 Model Tuning).

5.5. Relation Mapping

In this section we describe a “translation” table between Freebase relations and NL words was built.

5.5.1. Formula

The objective is to find the most likely relation a question prompts. For instance, for the question who is the father of King George VI, the most likely relation we look for is people.person.parents. To put it more formally, given a question Q of a word vector $\mathbf{w}$, we want to find out the relation R that maximizes the probability $P(R \mid Q)$.

More interestingly, for the question who is the father of the Periodic Table, the actual relation that encodes its original meaning is law.invention.inventor, rather than people.person.parents. This simple example points out that *every* part of the question could change what the question inquires eventually. Thus we need to count for *each* word w in Q. Due to the bias and incompleteness of any data source, we approximate the true probability of P with $\tilde{P}$ under our specific model. For the simplicity of computation, we assume conditional independence between words and apply Naive Bayes:

$$\begin{aligned}\tilde{P}(R \mid Q) &\propto \tilde{P}(Q \mid R)\tilde{P}(R) \\ &\approx \tilde{P}(\mathbf{w} \mid R)\tilde{P}(R) \\ &\approx \prod_{w} \tilde{P}(w \mid R)\tilde{P}(R)\end{aligned}$$

where $\tilde{P}(R)$ is the prior probability of a relation R and $\tilde{P}(w \mid R)$ is the conditional probability of word w given R.

It is possible that we do not observe a certain relation R when computing the above equation. In this case we back off to the “sub-relations”: a relation R is a concatenation of a series of sub-relations $R = \mathbf{r} = r_1.r_2.r_3.\ldots$. For instance, the sub-relations of people.person.parents are people, person, and parents. Again, we assume conditional independence between sub-relations and apply Naive Bayes:

$$\begin{aligned}
\tilde{P}_{\text{backoff}}(R \mid Q) &\approx \tilde{P}(\mathbf{r} \mid Q) \\
&\approx \prod_r \tilde{P}(r \mid Q) \\
&\propto \prod_r \tilde{P}(Q \mid r)\tilde{P}(r) \\
&\approx \prod_r \prod_w \tilde{P}(w \mid r)\tilde{P}(r)
\end{aligned}$$

One other reason that we estimated $\tilde{P}(w \mid r)$ and $\tilde{P}(r)$ for sub-relations is that Freebase relations share some common structures in between them. For instance, both people.person.parents and fictional_universe.fictional_character.parents indicate the parent relationship but the latter is much less commonly annotated. We hope that the shared sub-relation, parents, can help better estimate for the less annotated. Note that the backoff model would have a much smaller value than the original, due to double multiplication $\prod_r \prod_w$. In practice we normalize it by the sub-relations size to keep it at the same scale with $\tilde{P}(R \mid Q)$.

Finally, to estimate the prior and conditional probability, we need a massive data collection.

5.5.2. Implementation

The ClueWeb09[4] dataset is a collection of 1 billion webpages (5TB compressed in raw HTML) in 10 languages by Carnegie Mellon University in 2009. FACC1, the Freebase Annotation of the ClueWeb Corpus version 1 (Gabrilovich et al., 2013), contains index and offset of Freebase entities within the English portion of ClueWeb. Out of all 500 million English documents, 340 million were automatically annotated with at least one entity, with an average of 15 entity mentions per document. The precision and recall of annotation were estimated at $80-85\%$ and $70-85\%$ (Orr et al., 2013).[5] Figure 5.2 on the next page shows a snippet.

[4]http://lemurproject.org/clueweb09/

[5]http://googleresearch.blogspot.com/2013/07/11-billion-clues-in-800-million.html

- Like [FREEBASE mid=/m/0gnbw]*Michael Caine*[/FREEBASE] at the end of the [FREEBASE mid=/m/0j9hg]**Italian Job**[/FREEBASE] , where all looks lost, I said "hold on lads".
- Method Acting and [FREEBASE mid=/m/0bj9k]*Pacino*[/FREEBASE]'s [FREEBASE mid=/m/05ss0g]**Looking for Richard**[/FREEBASE]
- [FREEBASE mid=/m/030vnj]Luke Wilson[/FREEBASE] needs some heat: He's in a new movie with [FREEBASE mid=/m/0c7xjb]*Jessica Simpson*[/FREEBASE] called "[FREEBASE mid=/m/025sjcr]**Blonde Ambition**[/FREEBASE]
- February 27 2006: [FREEBASE mid=/m/0c7xjb]*Jessica Simpson*[/FREEBASE] pitching on the movie set of [FREEBASE mid=/m/0gf18g]**Employee of the month**[/FREEBASE] in [FREEBASE mid=/m/05fjy]New Mexico[/FREEBASE].
- February 25 2006: [FREEBASE mid=/m/0c7xjb]*Jessica Simpson*[/FREEBASE] and [FREEBASE mid=/m/03lzxk]Greg Coolidge[/FREEBASE] on set of the new movie "[FREEBASE mid=/m/0gf18g Employee Of The Month]**Employee Of The Month**[/FREEBASE]" - thanks to Criss for these!
- February 18 2006: [FREEBASE mid=/m/0c7xjb]*Jessica Simpson*[/FREEBASE] just returned to [FREEBASE mid=/m/030qb3t]Los Angeles[/FREEBASE] last night from [FREEBASE mid=/m/0f25y]Santa Fe[/FREEBASE] where she is filming "[FREEBASE mid=/m/0gf18g]**Employee of the Month**[/FREEBASE]".

Figure 5.2.: Snippets from Annotated Clueweb. **Blue bold** words and *red italic* words are the first and second arguments of the film.film.starring..film.performance.actor relation. The relation was determined by querying whether any two of the annotated entities had a direct relation according to Freebase. All entities were annotated with unique machine id (mid) in Freebase. These snippets were later aligned with the relation to find out the mostly likely natural language words that express the relation.

Given these two resources, for each binary Freebase relation, we can find a collection of sentences each of which contains both of its arguments, then simply learn how words in these sentences are associated with this relation, i.e., $\tilde{P}(w \mid R)$ and $\tilde{P}(w \mid r)$. By counting how many times each relation R was annotated, we can estimate $\tilde{P}(R)$ and $\tilde{P}(r)$. The learning task can be framed in the following short steps:

1. We split each HTML document by sentences (Kiss and Strunk, 2006) using NLTK (Bird and Loper, 2004) and extracted those with at least two Freebase entities which has at least one direct established relation according to Freebase.

2. The extraction formed two parallel corpora, one with "relation - sentence" pairs (for estimating $\tilde{P}(w \mid R)$ and $\tilde{P}(R)$) and the other with "subrelations - sentence" pairs (for $\tilde{P}(w \mid r)$ and $\tilde{P}(r)$). Each corpus has 1.2 billion pairs. All words were stemmed with the Snowball stemmer (Porter, 1980).

3. The tricky part was to align these 1.2 billion pairs. Since the relations on one side of these pairs are not *natural* sentences, we ran the most simple IBM alignment Model 1 (Brown et al., 1993) to estimate the translation probability with GIZA++ (Och and Ney, 2003). To speed up, the 1.2 billion pairs were split into 100 even chunks. We ran 5 iterations of EM on each one and finally aligned the 1.2 billion pairs from both directions. To symmetrize the alignment, common MT heuristics INTERSECTION, UNION, GROW-DIAG-FINAL, and GROW-DIAG-FINAL-AND (Koehn, 2010) were separately applied and evaluated later.

4. Treating the aligned pairs as *observation*, the co-occurrence matrix between aligning relations and words was computed. There were 10,484 relations and subrelations in all, and we kept the top 20,000 words.

5. From the co-occurrence matrix we computed $\tilde{P}(w \mid R)$, $\tilde{P}(R)$, $\tilde{P}(w \mid r)$ and $\tilde{P}(r)$.

Hand-checking the learned probabilities shows both success, failure and some bias. For instance, for the film.actor.film relation (mapping from film names to actor names), the top words given by $\tilde{P}(w \mid R)$ are won, star, among, show. For the film.film.directed_by relation, some important stop words that could indicate this relation, such as by and

sentence number:	**0**	**≤ 10**	**$\leq 10^2$**	**$\leq 10^3$**	**$\leq 10^4$**	**$> 10^4$**
relation percentage:	7.0%	0.7%	1.2%	0.4%	1.3%	89.5%

Table 5.1.: Percentage of answer relations (the incoming relation connected to the answer node) with respect to how many sentences we learned this relation from CluewebMapping. For instance, the first column says there are 7% of answer relations for which we cannot find a mapping (so we had to use the backoff probability estimation); the last column says there are 89.5% of answer relations that we were able to learn the mapping between this relation and text based on more than 10 thousand relation-sentence pairs. The total number of answer relations is 7886.

with, rank directly after director and direct. However, due to significant popular interest in certain news categories, and the resultant catering by websites to those information desires, then for example we also learned a heavily correlated connection between Jennifer Aniston and celebrity.infidelity.victim, and between some other you-know-who names and celebrity.infidelity.participant.

We next formally evaluate how the learned mapping help predict relations from words.

5.5.3. Evaluation

Both ClueWeb and its Freebase annotation has a bias. Thus we were firstly interested in the coverage of mined relation mappings. As a comparison, we used a dataset of relation mapping contributed by Berant et al. (2013) and Lin et al. (2012). The idea is very similar: they intersected Freebase relations with predicates in (arg1, predicate, arg2) triples extracted from `ReVerb` to learn the mapping between Freebase relations and triple predicates. Note the scale difference: although `ReVerb` was also extracted from ClueWeb09, there were only 15 million triples to intersect with the relations, while we had 1.2 billion alignment pairs. We call this dataset **ReverbMapping** and ours **CluewebMapping**.

The evaluation dataset, WebQuestions, was also contributed by Berant et al. (2013). It contains 3778 training and 2032 test questions collected from the Google Suggest service. All questions were annotated with answers from Freebase. Some questions have more than one answer, such as what to see near sedona arizona?.

We evaluated on the training set in two aspects: coverage and prediction performance. We define *answer node* as the node that is the answer and *answer relation* as the relation from the answer node to its direct parent. Then we computed how much and how well the answer relation was triggered by ReverbMapping and CluewebMapping. Thus for the question, who is the father of King George VI, we ask two questions: does the mapping, 1. (coverage) contain the answer relation people.person.parents? 2. (precision) predict the answer relation from the question?

Table 5.1 shows the coverage of CluewebMapping, which covers 93.0% of all answer relations. Among them, we were able to learn the rule mapping using more than 10 thousand relation-sentence pairs for *each* of the 89.5% of all answer relations. In contrast, ReverbMapping covers 89.7% of the answer relations.

Next we evaluated the prediction performance, using the evaluation metrics of information retrieval. For each question, we extracted all relations in its corresponding topic graph, and ranked each relation with whether it is the answer relation. For instance, for the previous example question, we want to rank the relation people.person.parents as number 1. We computed standard MAP (Mean Average Precision) and MRR (Mean Reciprocal Rank), shown in Table 5.2(a) (defined in detail in § 2.2.3.2 on page 40). As a simple baseline, "word overlap" counts the overlap between relations and the question. CluewebMapping ranks each relation by $\tilde{P}(R \mid Q)$. ReverbMapping does the same, except that we took a uniform distribution on $\tilde{P}(w \mid R)$ and $\tilde{P}(R)$ since the contributed dataset did not include co-occurrence counts to estimate these probabilities.[6] Note that the median rank from CluewebMapping is only 12, indicating that half of all answer relations are ranked in the top 12.

Table 5.2(b) further shows the percentage of answer relations with respect to their ranking. CluewebMapping successfully ranked 19% of answer relations as top 1. A sam-

[6]The way we used ReverbMapping was not how Berant et al. (2013) originally used it: they employed a discriminative log-linear model to judge relations and that might yield better performance. As a fair comparison, ranking of CluewebMapping under uniform distribution is also included in Table 5.2(a).

	Median Rank	**MAP**	**MRR**
word overlap	471	0.0380	0.0590
ReverbMapping	60	0.0691	0.0829
CluewebMapping	12	0.2074	0.2900
with uniform distribution	61	0.0544	0.0561

(a) Ranking on answer relations. Best result on CluewebMapping was under the GROW-DIAG-FINAL-AND heuristics (row 3) when symmetrizing alignment from both directions. The last row shows ranking of CluewebMapping under uniform distribution (assuming counting on words and relations is not known).

	1	**≤ 5**	**≤ 10**	**≤ 50**	**≤ 100**	**> 100**
word overlap	3.5	4.7	2.5	3.9	4.1	81.3
ReverbMapping	2.6	9.1	8.6	26.0	13.0	40.7
CluewebMapping	19.0	19.9	8.9	22.3	7.5	22.4

(b) Percentage of answer relations w.r.t. ranking number (header).

Table 5.2.: Evaluation on answer relation ranking prediction on 3778 training questions.

ple of these includes person.place_of_birth, location.containedby, country.currency_used, regular_tv_appearance.actor, etc. These percentage numbers are good clue for feature design: for instance, we may be confident in a relation if it is ranked top 5 or 10 by CluewebMapping.

To conclude, we found that CluewebMapping provides satisfying coverage on the 3778 training questions: only 7% were missing, despite the biased nature of web data. Also, CluewebMapping gives reasonably good precision on its prediction, despite the noisy nature of web data. A sample of the top relations with their most aligned words (in stemmed form) is shown in Table 5.3. We move on to fully evaluate the final QA F_1.

relation	**top words in stemmed form**
location.location.contains	in, at, citi, locat, area, town, near, hotel, of, center
location.adjoining_relationship.adjoins	and, state, follow, are, includ, from, or, we, all, border
military.military_combatant_group.combatants	and, countri, are, from, includ, have, or, other, world, were
location.imports_and_exports.imported_from	and, are, or, countri, have, than, world, we, over, from
location.country.administrative_divisions	in, is, from, a, year, old, inc, deliveri, florist, guy
people.person.nationality	presid, his, s, was, who, minist, that, said, by, prime
government.government_position_held.jurisdiction_of_office	presid, that, s, minist, said, inc, prime, deliveri, florist, transworld
award.award_nomination.award_nominee	star, as, with, by, s, freebas, produc, cast, featur, film
location.mailing_address.citytown	at, in, freebas, base, campus, depart, headquart, univers, attend, museum
government.government_position_held.office_holder	presid, to, s, that, senat, minist, prime, said, governor, his
award.award_honor.award_winner	s, as, with, by, star, freebas, on, produc, jennif, anistonjennif
location.mailing_address.country	regist, iso, at, by, compani, it, has, is, research, largest
location.country.capital	in, capit, on, hotel, citi, said, base, intern, offici, held
government.government_position_held.district_represented	is, a, inc, deliveri, florist, transworld, from, state, of, senat
award.award_nomination.nominated_for	s, for, by, with, film, star, on, movi, as, his
film.performance.actor	star, as, movi, film, role, with, s, play, his, who
people.place_lived.location	was, his, born, s, mayor, he, governor, who, said, former
sports.sports_team_roster.player	for, with, score, s, player, his, goal, sign, point, season

Table 5.3.: Top Freebase relations determined by their argument appearance counts in the annotated Clueweb and their top aligned words (stemmed form).

5.6. Experiments

We evaluate the final F_1 in this section. The system of comparison is that of Berant et al. (2013) and Berant and Liang (2014).[7]

5.6.1. Notes on Evaluation Metrics

There have been debate and inconsistencies about how the final QA system should be evaluated. The SEMPRE homepage [8] finally released an evaluation script. The evaluation metrics are:

- macro P: averaged precision on each answered question;
- macro R: averaged recall on each answered question;
- macro F_1: harmonic mean of macro P and macro R;
- average F_1: for each question, compute F_1 ($F_1 = 0$ for not answered questions), then average all F_1's over either only answered questions, or all questions.

The steps to compute these values are:

1. compute $P/R/F_1$ for each question;
2. average all answered questions to obtain macro P/R;
3. compute the harmonic mean of macro P/R to get macro F_1;
4. average all questions or all answered questions to obtain average F_1 over all questions, or all answered questions;

Note that both Berant et al. (2013) and Berant and Liang (2014) used the term *accuracy* but what they actually computed was average F_1 over all questions. The reason is that the semantic parsing community has a long tradition of using accuracy to evaluate their systems. In previous datasets systems answered *all* questions and each answer is either

[7]Berant and Liang (2014) was published simultaneously with our work (Yao and Van Durme, 2014) at the same conference. At the time of writing, we had no knowledge about Berant and Liang (2014).

[8]`www-nlp.stanford.edu/software/sempre/`

totally correct or totally wrong. Thus accuracy was a good measure. However, on the WebQuestions dataset, partial credits were allowed: in order to compute partial credit, F_1 for each question was computed, then averaged – or so called "accuracy".

§ 2.2.3.4 on page 42 gives concrete examples on how to compute these values. In the follow sections the scores on the TEST set are based on this new evaluation script. The scores on the DEV set are still based on my own evaluation script, which is quite similar to macro F_1. But unfortunately I have lost the DEV set result and cannot produce the new scores. Even though, the comparisons within the DEV set are still valid.

5.6.2. Data

We re-used WebQuestions, a dataset collected by Berant et al. (2013). It contains 5810 questions crawled from the Google Suggest service, with answers annotated on Amazon Mechanical Turk. All questions contain at least one answer from Freebase. A sample is shown in Table 5.4. This dataset has been split by 65%/35% into TRAIN-ALL and TEST. We further randomly divided TRAIN-ALL by 80%/20% to a smaller TRAIN and development set DEV. Note that our DEV set is different from that of Berant et al. (2013), but the final result on TEST is directly comparable. Results are reported in terms of macro F_1 with partial credit (following Berant et al. (2013)) if a predicted answer list does not have a perfect match with all gold answers, as a lot of questions in WebQuestions contain more than one answer.

5.6.3. Search

With an Information Retrieval (IR) front end, we need to locate the exact Freebase topic node a question is about. For this purpose we used the Freebase Search API.[9] All named entities [10] in a question were sent to this API, which returned a ranked list of relevant topics. Table 5.5 shows an example. We also evaluated how well the search

[9] `https://developers.google.com/freebase/v1/search-overview`

[10] When no named entities are detected, we fall back to noun phrases.

question	**answer**
what are king charles spaniels?	Dog
what club team is diego forlan on?	Uruguay national football team
what years did the red sox win the world series?	1903 World Series, ..., 2007 World Series
what character did john noble play in lord of the rings?	Denethor II
what is the most practiced religion in the united states?	Christianity
where do the orioles play spring training?	Baltimore
what currency do the ukraine use?	Ukrainian hryvnia
what countries do people speak portuguese?	Brazil, Canada, Angola, ...
where did the assyrian empire start?	Middle East
what is the currency of germany in 2010?	Euro
what year did allen iverson get married?	8/3/2001
what is the political system in england?	Constitutional monarchy
where are the mines in victoria?	Victoria
what to see outside of paris?	Stade de France, 20th arrondissement, ...
what type of government does israel?	Parliamentary system
what were some inventions of leonardo da vinci?	Double hull, Viola organista
who killed vincent chin dvd?	Ronald Ebens, Michael Nitz
what are the gods of islam?	Allah
who has won the most fa cup?	Liverpool F.C.
how many major dialects are there in china?	Nepali Language, Standard Tibetan, ...
how many australian states and territories?	City of Moreland, Northern Territory, ...
where did english numbers originate from?	Latin alphabet
who did jason segel date?	Linda Cardellini

Table 5.4.: A sample of the WebQuestions dataset (training only).

Freebase Topic	**Score**
natalie_portman	722.30
star_wars	233.01
saturday_night_live_season_31	56.36
clone_wars	51.47
lego_star_wars	38.73
star_wars_music	37.78
star_wars_episode_iv_a_new_hope	36.67
star_wars_episode_i_the_phantom_menace	35.12
star_wars_clone_wars	33.58
star_wars_galaxies	29.21

Table 5.5.: Returned Freebase topics with scores from query the Freebase Search API with natalie portman and star wars. The two queries were fired separately and the returned snippets were merged and ordered according to their scores. The two entities came from the question who did natalie portman play in star wars?. They were tagged as either person name or noun phrase by the Stanford CoreNLP suite.

API served the IR purpose. WebQuestions not only has answers annotated, but also which Freebase topic nodes the answers come from.[11] Thus we evaluated the ranking of retrieval with the gold standard annotation on train-all, shown in Table 5.6. The top 2 results of the Search API contain gold standard topics for more than 90% of the questions and the top 10 results contain more than 95%. We took this as a "good enough" IR front end and used it on test.

Once a topic is obtained we query the Freebase Topic API [12] to retrieve all relevant information, resulting in a topic graph. The API returns almost identical information as displayed via a web browser to a user viewing this topic. Given that Turkers annotated answers based on the topic page via a browser, this supports the assumption that the same answer would be located in the topic graph, which is then passed to the QA engine for feature extraction and classification.

[11]an example: for the question what is the name of justin bieber brother, WebQuestions has `www.freebase.com/view/en/justin_bieber` as the topic node.
[12]`https://developers.google.com/freebase/v1/topic-overview`

top	**1**	**2**	**3**	**5**	**10**
#	3263	3456	3532	3574	3604
%	86.4	91.5	93.5	94.6	95.4

Table 5.6.: Evaluation on the Freebase Search API: how many questions' top n retrieved results contain the gold standard topic. Total number of questions is 3778 (size of TRAIN-ALL). There were only 5 questions with no retrieved results.

5.6.4. Model Tuning

We treat QA on Freebase as a binary classification task: for each node in the topic graph, we extract features and judge whether it is the answer node. Every question was processed by the Stanford CoreNLP suite with the caseless model. Then the question features (§5.4.1) and node features (§5.4.2) were combined (§5.4.3) for each node. The learning problem is challenging: for about 3000 questions in TRAIN, there are 3 million nodes (1000 nodes per topic graph), and 7 million feature types. We employed a high-performance machine learning tool, `Classias` (Okazaki, 2009). Training usually took around 4 hours in total. We experimented with various discriminative learners on DEV, including logistic regression, perceptron and SVM, and found L1 regularized logistic regression to give the best result. The L1 regularization encourages sparse features by driving feature weights towards zero, which was ideal for the over-generated feature space. After training, we had around 30 thousand features with non-zero weights, a 200 fold reduction from the original features.

Also, we did an ablation test on DEV about how additional features on the mapping between Freebase relations and the original questions help, with three feature settings: 1) "basic" features include feature productions read off from the feature graph (Figure 5.1); 2) "+ word overlap" adds additional features on whether sub-relations have overlap with the question; and 3) "+ CluewebMapping" adds the ranking of relation prediction given the question according to CluewebMapping. Table 5.7 shows that the additional CluewebMapping features improved overall F_1 by 5%, a 13% relative improvement: a remarkable gain given that the model already learned a strong correlation between question types and answer types (explained more in discussion and Table 5.9 later).

Finally, the ratio of positive vs. negative examples affect final F_1: the more positive examples, the lower the precision and the higher the recall. Under the original setting, this ratio was about 1 : 275. This produced (under old evaluation script) precision around 60% and recall around 35% (c.f. Table 5.7). To optimize for F_1, we down-sampled the negative examples to 20%, i.e., a new ratio of 1 : 55. This boosted the final F_1 on DEV to 48%. We report the final TEST result under this down-sampled training. In practice the precision/recall balance can be adjusted by the positive/negative ratio.

5.6.5. Test Results

Table 5.8 gives the final F_1's on TEST using the shared evaluation script. "Gold Retrieval" always ranked the correct topic node top 1, a perfect IR front end assumption. In a more realistic scenario, we had already evaluated that the Freebase Search API returned the correct topic node 95% of the time in its top 10 results (c.f. Table 5.6), thus we also tested on the top 10 results returned by the Search API. To keep things simple, we did not perform answer voting, but simply extracted answers from the first (ranked by the Search API) topic node with predicted answer(s) found.

There are two ways to view Table 5.8:

- performance on the questions the system was confident about: this is the macro F_1. The logistic regression ranker has a default threshold of 0.5 for positive prediction. This threshold enabled answering 1605 of 2032 questions. The macro F_1 of 46.5% was system performance on these 1605 questions the system was confident about.[13] Since this only counts as roughly 80% of all questions, the average F_1 over all questions was significantly lower.

- performance on all questions: this is the average F_1 over all. SEMPRE chose to answer every question in the dataset, thus its average F_1 was higher. To compare our system fairly on this level, we lowered the threshold to force an answer out of all questions, obtaining the average F_1 of 35.4%, which was comparable (35.4%

[13]In retrospect: the best way to optimize F_1 is to draw the precision/recall curve (c.f. § 2.2.3.3 on page 41) on the DEV set; select the threshold that gives the best macro F_1; then use this threshold to compute macro F_1 on TEST.

	P	**R**	$\mathbf{F_1}$
basic	57.3	30.1	39.5
+ word overlap	56.0	31.4	40.2
+ CluewebMapping	59.9	35.4	44.5
+both	59.0	35.4	44.3

Table 5.7.: F_1 (using old evaluation script) on DEV with different feature settings.

	#Q	**macro P**	**macro R**	**macro** $\mathbf{F_1}$	**average** $\mathbf{F_1}$ all/answered
our method	1605	38.8	58.0	**46.5**	33.0/41.8
force to answer all questions	2032	33.5	48.0	39.5	35.4
with Gold Retrieval	1957	45.4	52.2	48.6	36.2/37.6
Berant et al. (2013) (bug fix)	2032	48.0	41.3	44.4	35.7
Berant and Liang (2014)	2032	40.5	46.6	43.3	**39.9**

Table 5.8.: F_1 on TEST (2032 questions in total). Gold Retrieval always returns the correct topic node as the first retrieved result. #Q is the number of answered questions. Macro F_1 is the harmonic mean of macro P/R. Average F_1 averages each individual F_1 computed for each question over either all 2032 questions or only those answered questions.

vs. 35.7%) to the bug corrected version of Berant et al. (2013).

The other question of interest is that whether our system has acquired some level of "machine intelligence": how much does it know what the question inquires? We discuss it below through feature and error analysis.

5.6.6. Error Analysis

The combination between questions and Freebase nodes captures some real gist of QA pattern typing, shown in Table 5.9 with sampled features and weights. A full list of the top 40 positive and negative features are shown in Table 5.10 and 5.11 separately. Our system learned, for instance, when the question asks for geographic adjacency information (qverb=border), the correct answer relation to look for is location.adjoins.

weight	**feature**
5.56	qfocus=money|type=Currency
5.11	qword=when|type=datetime
4.56	qverb=border|rel=location.adjoins
3.90	qword=why|incoming_relation_rank=top_3
2.94	qverb=go|qtopic=location|type=Tourist_attraction
-3.94	qtopic=location|rel=location.imports_exports.date
-2.93	qtopic=person|rel=education.end_date

Table 5.9.: A sample of the top 50 most positive/negative features. Features are production between question and node features (c.f. Figure 5.1).

Even with perfect retrieval, the final macro F_1 was still less than 50%. We found that list questions were harder to answer perfectly. Our system is also weak in two types of questions: 1. questions with constraints on topics, such as what is the new orleans hornets *new* name and what was reagan *before* president. Our features did not cover these temporal constraints such as new and before. 2. counting questions (how many) which require a special *count()* operator on the answer candidates. These types of questions appear more prominently in the other FREE917 dataset (Cai and Yates, 2013a) and might be better handled by the semantic parsing approaches. We provide a detailed comparison with the output from Berant et al. (2013) in the next section.

5.7. Comparison: Information Extraction vs. Semantic Parsing

We have shown in Table 5.8 that jacana-freebase, a system based on information extraction, outperforms SEMPRE (Berant et al., 2013), a system based on semantic parsing, on the test set of WEBQUESTIONS. We characterize semantic parsing as the task of deriving a representation of meaning from language, sufficient for a given task. Traditional information extraction from text may be coarsely characterized as representing a certain level of semantic parsing, where the goal is to derive enough meaning in order to

weight	feature
8.60426	qfocus=religions|/type/object/type=Religion
6.22916	qfocus=sports|/type/object/type=Sports_Team
6.04425	qfocus=religions|/type/object/type=Religious_Organization
5.88405	qfocus=currency|/type/object/type=Currency
5.56413	qfocus=money|/type/object/type=Currency
5.43707	qfocus=sport|/type/object/type=Sports_Team
5.40111	qtopic_ner=ORGANIZATION|out_rel=/sports/sports_league_participation/team
5.34641	qverb=die|qword=what|/type/object/type=Cause_Of_Death
5.34273	music|/type/object/type=Musical_genre
5.11456	qtopic_ner=DURATION|nodetype=datetime
5.05355	qtopic_ner=LOCATION|out_rel=/location/administrative_division.../capital
4.92154	qfocus=timezone|/type/object/type=Time_Zone
4.8497	qfocus=party|/type/object/type=Political_party
4.74702	qtopic_ner=LOC|out_rel=/location/administrative_division_.../administrative_division
4.59445	qverb=live|out_rel=/people/place_lived/location
4.5645	qverb=border|out_rel=/location/adjoining_relationship/adjoins
4.54962	qfocus=music|/type/object/type=Musical_genre
4.5315	qfocus=airport|/type/object/type=Airport
4.45438	qverb=have|/type/object/type=Form_of_Government
4.4374	qverb_tag=VB|dobj|qword=what|out_rel=/location/imports_exports.../industry
4.27125	qverb=happen|dobj|qtopic_ner=PERSON|/type/object/type=Profession
4.23816	in|/type/object/type=Soundtrack
4.20004	cruis|/type/object/type=Transport_terminus
4.19823	govern|/type/object/type=Form_of_Government
4.10507	qtopic_ner=MISC|out_rel=/business/company_brand_relationship/company
4.0946	qfocus=education|incoming_relation_text_overlap=some_overlap
4.015	qfocus=continent|/type/object/type=Continents
3.97087	qverb_tag=VB|prep_on|qtopic_ner=DATE|nodetype=compound
3.89806	qword=why|incoming_relation_rank=top_3
3.88312	qfocus=music|/type/object/type=Composition_type
3.8393	citi|node_text_overlap=total_subsume
3.82019	qtopic_ner=LOCATION|out_rel=/location/imports_and_exports/exported_to
3.79286	qword=when|/type/object/type=Top_Level_Domain
3.77478	qverb=influence|/type/object/type=Influence_Node
3.6726	qfocus=albums|/type/object/type=Musical_Album
3.64795	qfocus=teams|/type/object/type=Sports_Team
3.58191	qfocus=city|/type/object/type=City/Town/Village
3.43911	qfocus=religion|/type/object/type=Religion
3.43494	scienc|/type/object/type=Namesake
3.42661	qtopic_ner=MISC|/type/object/type=Recurring_Period
3.36592	qword=when|nodetype=datetime
3.32011	qverb_tag=VBP|/type/object/type=Color

Table 5.10.: Full list of top 40 positive features.

weight	feature
-6.8883	qword=what|/type/object/type=Reporting_Unit
-4.1717	qverb=influence|incoming_relation_text_overlap=no_overlap
-3.94465	qtopic_ner=LOCATION|out_rel=/location/imports_and_exports/date
-3.9102	qfocus=money|is_island=true
-3.53065	qtopic_ner=PERSON|out_rel=/film/performance/character
-3.41437	parent|incoming_relation_text_overlap=no_overlap
-3.26789	qtopic_ner=DATE|/type/object/type=Tournament_Competitor
-2.95402	qword=what|/type/object/type=Olympic_games
-2.93283	qtopic_ner=PERSON|out_rel=/education/education/end_date
-2.82172	high|node_text_overlap=no_overlap
-2.80396	qtopic_ner=PERSON|out_rel=/people/place_lived/start_date
-2.7888	qverb=play|qtopic_ner=PERSON|/type/object/type=Fictional_Character_Creator
-2.59511	qfocus=currency|parent_node_text_overlap=some_overlap
-2.51765	qtopic_ner=LOCATION|prep_in|qtopic_ner=DATE|is_island=true
-2.47958	qverb=inspire|qtopic_ner=PERSON|incoming_relation_text_overlap=no_overlap
-2.39718	qfocus=religion|incoming_relation_text_overlap=no_overlap
-2.39195	qverb=use|prep_in|qtopic_ner=LOCATION|incoming_relation_rank=top_1
-2.33476	qverb=be|out_rel=/common/topic/notable_types
-2.31186	qtopic_ner=PERSON|/type/object/type=Extended_institution
-2.27512	qtopic_ner=PERSON|out_rel=/tv/regular_tv_appearance/character
-2.19146	in|incoming_relation_text_overlap=some_overlap
-2.13185	qword=where|out_rel=/common/topic/notable_types
-2.1318	children|qtopic_ner=PERSON|incoming_relation_text_overlap=no_overlap
-2.12925	qtopic_ner=PERSON|/type/object/type=User
-2.12267	qword=who|out_relation_rank=beyond_top_100
-2.11863	qword=who|out_rel=/tv/regular_tv_appearance/series
-2.11708	qverb_tag=VBZ|qtopic_ner=ORG|parent_node_text_overlap=some_overlap
-2.07752	qword=what|/type/object/type=abh-city
-2.07493	qword=who|nsubj|qtopic_ner=PERSON|/type/object/type=Person
-2.07106	qtopic_ner=LOCATION|/type/object/type=Government_Office_or_Title
-2.04276	currenc|prep_of|qtopic_ner=LOCATION|incoming_relation_rank=top_1
-2.03287	qtopic_ner=ORG|nn|qtopic_ner=ORG|/type/object/type=Sports_team_extra
-2.00522	qword=what|nsubj|qtopic_ner=LOCATION|/type/object/type=Topic
-1.99172	qtopic_ner=PERSON|out_rel=/base/popstra/sww_base/source
-1.98979	origin|/type/object/type=sww_base
-1.97703	qverb_tag=VB|qword=what|/type/object/type=BV=_Medical_Condition
-1.94967	qtopic_ner=LOCATION|/type/object/type=Intézmények
-1.93891	qtopic_ner=MISC|/type/object/type=Film_director
-1.90722	new|/type/object/type=Film_actor
-1.90116	qtopic_ner=LOCATION|/type/object/type=Reporting_Unit
-1.89828	qverb=go|/type/object/type=Bibs_Location
-1.88412	most|/type/object/type=Topic

Table 5.11.: Full list of top 40 negative features.

populate a database with factoids of a form matching a given schema.[14] Given the ease with which reasonably accurate, deep syntactic structure can be automatically derived over (English) text, it is not surprising that IE researchers would start including such "features" in their models.

In this section we compare the output from the two systems in a detailed manner and try to answer a central question: what is the difference between an information extraction system with access to syntax, as compared to a semantic parser, when both are targeting a factoid-extraction style task?

We find that these two systems are on par with each other, with no significant differences in terms of accuracy between them. A major distinction between the work of Berant et al. (2013) and ours (Yao and Van Durme, 2014) is the ability of the former to represent, and compose, aggregation operators (such as `argmax`, or `count`), as well as integrate disparate pieces of information. This representational capability was important in previous, closed-domain tasks such as GeoQuery. The move to Freebase by the SP community was meant to provide richer, open-domain challenges. While the vocabulary increased, our analysis suggests that compositionality and complexity decreased. We therefore conclude that the semantic parsing community should target more challenging open-domain datasets, ones that "standard IE" methods are less capable of attacking.

5.7.1. Evaluation Metrics

Both Berant et al. (2013) and we tested their systems on the WebQuestions dataset. Berant et al. (2013) reported a score of 31.4% in terms of accuracy (with partial credit if inexact match) on the test set and later in Berant and Liang (2014) revised it to 35.7%. Berant et al. focused on "accuracy" (or more precisely, average F_1 over all questions) – how many questions were correctly answered by the system. Our system on the other hand only answered 80% of all test questions. For the purpose of comparing among *all* test questions, we lowered the logistic regression prediction threshold (usually 0.5) on jacana-freebase for the other 20% of questions where jacana-freebase had not proposed

[14]So-called Open Information Extraction (OIE) is simply a further blurring of the distinction between IE and SP, where the schema is allowed to grow with the number of verbs, and other predicative elements of the language.

		jacana ($F_1 = 1$)		**jacana** ($F_1 \geq 0.5$)	
		$\surd$	$\times$	$\surd$	$\times$
Sempre	$\surd$	153 (0.08)	383 (0.19)	429 (0.21)	321 (0.16)
	$\times$	136 (0.07)	1360 (0.67)	366 (0.18)	916 (0.45)

Table 5.12.: The absolute and proportion of questions Sempre and jacana-freebase answered correctly ($\surd$) and incorrectly ($\times$) jointly and separately, running a threshold F_1 of 1 and 0.5.

an answer to, and selected the best-possible prediction with the highest prediction score as the answer. In this way jacana-freebase was able to answer all questions with a lower accuracy of 35.4%. In the following we present analysis results based on the test questions where the two systems had very similar performance (35.7% vs. 35.4%). The difference is not significant according to the paired permutation test (Smucker et al., 2007).

5.7.2. Accuracy vs. Coverage

First, we were interested to see the proportions of questions Sempre and jacana-freebase jointly and separately answered correctly. The answer to many questions in the dataset is a set of answers, for example what to see near sedona arizona?. Since turkers did not exhaustively pick out all possible answers, evaluation is performed by computing the F_1 between the set of answers given by the system and the answers provided by turkers. With a strict threshold of $F_1 = 1$ and a permissive threshold of $F_1 \geq 0.5$ to judge the correctness, we list the pair-wise correctness matrix in Table 5.12. Not surprisingly, both systems had most questions wrong given that the averaged F_1's were only around 35%. With the threshold $F_1 = 1$, Sempre answered more questions exactly correctly compared to jacana-freebase, while when $F_1 \geq 0.5$, it was the other way around. This shows that Sempre is more accurate in certain questions. The reason behind this is that Sempre always fires queries that return exactly one set of answers from Freebase, while jacana-freebase could potentially tag multiple nodes as the answer, which may lower the accuracy.

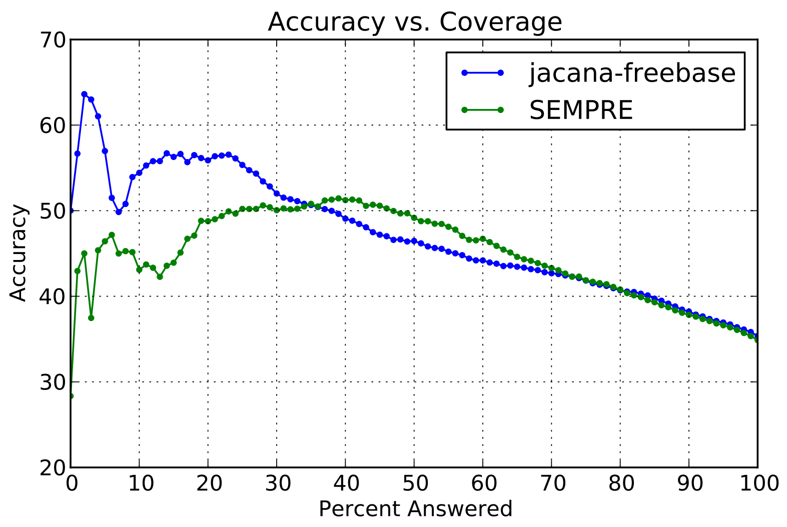


Figure 5.3.: Accuracy with respect to proportion of questions answered

We have shown that both systems can be more accurate in certain questions, but when? Is there a correlation between the system confidence and accuracy? Thus we took the logistic decoding score (between 0 and 1) from jacana-freebase and the probability from the log-linear model used by SEMPRE as confidence, and plotted an "accuracy vs. coverage" curve, which shows the accuracy of a QA engine with respect to its coverage of all questions. The curve basically answers one question: at a fixed accuracy, what is the proportion of questions that can be answered? A better system should be able to answer more questions correctly with the same accuracy.

The curve was drawn in the following way. For each question, we select the best answer candidate with the highest confidence score. Then for the whole test set, we have a list of (question, highest ranked answer, confidence score) tuples. Running a threshold from 1 to 0, we select those questions with an answer confidence score above the threshold and compute accuracy at this point. The X-axis indicates the percentage of questions above the threshold and the Y-axis the accuracy, shown in Figure 5.3.

The two curves generally follow a similar trend, but while jacana-freebase has higher accuracy when coverage is low, SEMPRE obtains slightly better accuracy when more questions are answered.

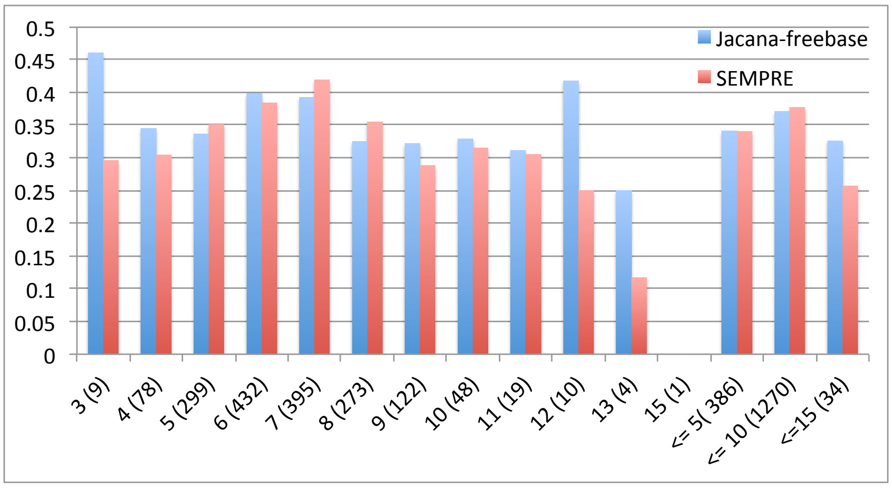


Figure 5.4.: Accuracy (Y-axis) by question length. The X-axis specifies the question length in words and the total number of questions in parenthesis.

5.7.3. Accuracy by Question Length and Type

Do accuracies of the two systems differ with respect to the complexity of questions? Since there is no clear way to measure question complexity, we use question length as a surrogate and report accuracies by question length in Figure 5.4. Most of the questions were 5 to 8 words long and there was no substantial difference in terms of accuracies. The major difference lies in questions of length 3, 12 and 13. However, the number of such questions was not high enough to show any statistical significance.

Figure 5.5 further shows the accuracies with respect to the question types (as reflected by the WH-word). Again, there is no significant difference between the two systems.

5.7.4. Learned Features

What did the systems learn during training? We compare them by presenting the top features by weight, as listed in Table 5.13. Clearly, the type of knowledge learned by the systems in these features is similar: both systems learn to associate certain phrases with predicates from the KB.

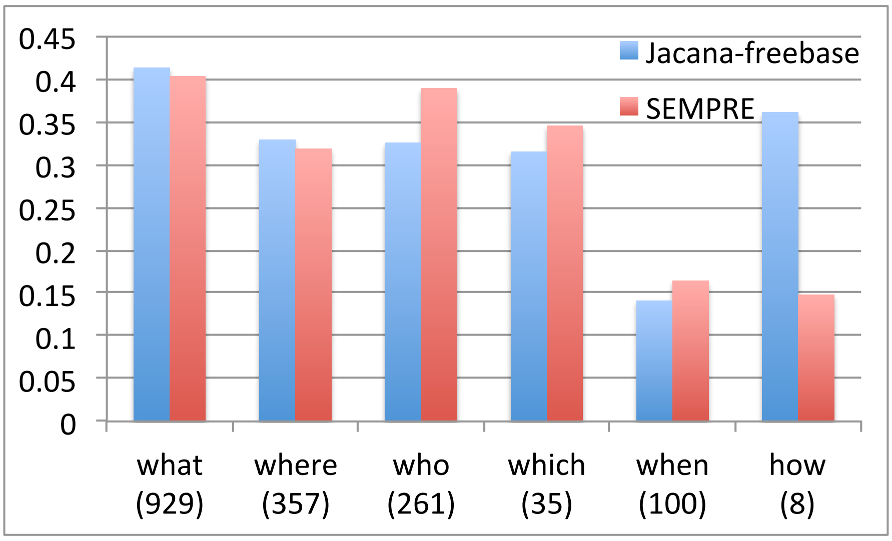


Figure 5.5.: Accuracy by question type (and the number of questions).

We note, however, that SEMPRE also obtains information from the fully constructed logical form. For instance, SEMPRE learns that logical forms that return an empty set when executed against the KB are usually incorrect (the weight for this feature is -8.88). In this respect the SP approach "understands" more than the IE approach.

We did not further compare on other datasets such as GeoQuery (Tang and Mooney, 2001) and FREE917 (Cai and Yates, 2013a). The first one involves geographic inference and multiple constraints in queries, directly fitting the compositional nature of semantic parsing. The second one was manually generated by looking at Freebase topics. Both datasets were less realistic than the WEBQUESTIONS dataset. Both datasets were also less challenging (accuracy/F_1 were between 80% and 90%) compared to WEBQUESTIONS (around 40%).

5.7.5. Summary

Our analysis of two QA approaches, semantic parsing and information extraction, has shown no significant difference between them. Note the similarity between features used in both systems shown in Table 5.13: the systems learned the same "knowledge" from data, with the distinction that the IE approach acquired this through a direct association

feature	weight
qfocus=religion|type=Religion	8.60
qfocus=money|type=Currency	5.56
qverb=die|type=CauseOfDeath	5.35
qword=when|type=datetime	5.11
qverb=border|rel=location.adjoins	4.56

(a) jacana-freebase

feature	weight
die from=CauseOfDeath	10.23
die of=CauseOfDeath	7.55
accept=Currency	7.30
bear=PlaceOfBirth	7.11
in switzerland=Switzerland	6.86

(b) SEMPRE

Table 5.13.: Learned top features and their weights for jacana-freebase and SEMPRE.

between dependency parses and answer properties, while the SP approach acquired this through optimizing on intermediate logic forms.

With a direct information extraction technology easily getting on par with the more sophisticated semantic parsing method, it suggests that SP-based approaches for QA with Freebase has not yet shown its power from a "deeper" understanding of the questions, among questions of various lengths. We suggest that more compositional open-domain datasets should be created, and that SP researchers should focus on utterances in existing datasets that are beyond the reach of direct IE methods.

5.8. Conclusion

This chapter is based on the following two published papers:

Xuchen Yao and Benjamin Van Durme. *Information Extraction over Structured Data: Question Answering with Freebase.* ACL. Baltimore, MD, USA. 2014.

Xuchen Yao, Benjamin Van Durme and Jonathan Berant. *Freebase QA:*

Information Extraction or Semantic Parsing?. ACL Workshop on Semantic Parsing. Baltimore, MD, USA. 2014.

The main ideas and scientific contributions are:

A novel method to use information extraction over structured data for question answering. Modern open-domain knowledge base presents an enormous search space for any question answering systems. Semantic parsing methods focus on generating a precise database query from an appropriate logic form, usually converted from the syntactic parses of the question. This process can be succinctly represented as:

question → syntactic parse → logic form → database query → answer

We instead showed that a simpler but novel method achieved the same or better result. We followed the intuition of how a human would search the answer over a knowledge base: first go to the page of relevant topic (coarse search), then extract the exact answer node (fine-grained pinpointing). This process can be represented as:

question → information retrieval → information extraction → answer

Note that it is exactly the traditional pipeline of question answering over text. In this process useful linguistic signals for answer patterns are represented by graph relations and properties of candidate nodes, rather than traditionally linguistic annotations of text fragments. We applied exactly the same *feature-driven* idea to automatically generate features and learn from training data. Results are promising: we outperformed the semantic parsing approach in terms of F_1 with less complex linguistic processing. A fair comparison between jacana-freebase and SEMPRE also shows that semantic parsing on the web-style factoid structured data roughly equals information extraction: both approaches learned very similar features and achieved comparable results. I hope that this result evokes thoughts in two directions: first, semantic parsing researchers should target on more compositional questions to show its power; second, methods for question answering over structured data do not limit to semantic parsing – information extraction is less complicated but even more effective.

Web-scale data mining for textual schema matching. Textual schema matching is the task of mapping natural language words with database relations. It is a key component in any KB-backed question answering systems. Previous approaches have used templates (Unger et al., 2012), Wikipedia (Cai and Yates, 2013b), Wiktionary (Kwiatkowski et al., 2013), and curated ReVerb knowledge base (Berant et al., 2013). We pushed this data challenge to its limit by mining the ClueWeb corpus, a 5TB crawl of the web. The final mapping between about 2 thousand Freebase relations and 10 thousand natural language words is represented as a log-probability table and a table of raw co-occurrences. By sharing this data with the community, we lifted the barrier of very demanding computational resources to process almost the entire ClueWeb. Both intrinsic evaluation of mapping quality and coverage and extrinsic evaluation of end QA performance proved this resource's usefulness.

Conclusion and Future Work We proposed an automatic method for Question Answering from structured data source (Freebase). Our approach associates question features with answer patterns described by Freebase and has achieved state-of-the-art result on a balanced and realistic QA corpus. To compensate for the problem of domain mismatch or overfitting, we exploited ClueWeb, mined mappings between KB relations and natural language text, and showed that it helped both relation prediction and answer extraction. Our method employs relatively lightweight machinery but has good performance. We hope that this result establishes a new baseline against which semantic parsing researchers can measure their progress towards deeper language understanding and answering of human questions.

Future work can be explored in a few directions:

- More complicated questions. The median length of the WebQuestions dataset is 7 words (c.f. Figure 5.4 on page 230). A lot of the questions can be reduced to binary slot filing tasks, such as: who does joakim noah play for?. A very simple question analysis on the question type, focus and topic would capture the gist of the query. In the future I would like to explore whether our information extraction based method solves more complicated questions, such as those require n-nary relations. One direction distinct from semantic parsing could be directly converting dependency trees to logic forms (Rudinger and Van Durme, 2014).

- Mapping of words for more implicit or higher-order relations. Natural language words can express more relations that a KB encompasses. For instance, grandparent is a concatenation of two /people/person/parent relations, and brother-in-law is a concatenation of /people/person/spouse and /people/person/sibling_s relations. So far we have only mined words for direct relation. Can we go further to learning the mapping for high-order relations?

- Exploit the graph nature of Freebase. All research except ours so far has only treated the graph-based Freebase as triple stores based on the RDF (Resource Description Framework) standard. Triple stores are not the natural way to explore a graph database. For instance, if we want to find out what is in common between two nodes in a graph, a Dijkstra's algorithm would be a good fit for finding their nearest path. However, it is computationally very expensive to find the path with triple stores (think of triple stores as tables).

- Generating questions from Freebase. To look at things in an inverse way, the WebQuestions dataset is a good example of how questions are generated from Freebase. Fader et al. (2013) also collected 18 million natural language questions from WikiAnswers. Can we learn a question grammar from these 18 million questions and generate fact-based questions using Freebase? Suppose for every fact in Freebase, we could generate one or several questions for this fact: i.e., enumerating all possible questions that can be asked on Freebase, then can we use the aligner described in Chapter 4 on page 134 to align a new question with the automatically generated questions with known answers? This is certainly another way to perform question answering on Freebase. The automatically generated questions from Freebase facts can also be used as "trivia" questions, in, for instance, intelligent educational tutoring systems.

- Integrating answers from both structured and unstructured sources. KB-powered systems are usually of high precision; text-powered systems have better recalls. A natural next step is a hybrid system that utilizes the power of both sources.

6. Conclusion and Future Directions

6.1. Summary

This dissertation is about two topics: **feature-driven question answering** and **discriminative methods for monolingual alignment**.

6.1.1. Feature-driven Question Answering

The first topic is extremely difficult to be novel about: there has already been decades of research extensively studying the QA problem. Yet it is still a well-defined, fundamental, and unsolved NLP application. Some of the early and most useful QA techniques (c.f. § 2.3 on page 46) include template matching, answer typing, tree/graph matching and web redundancy, with various degrees of machine learning. The last one, web redundancy, is a "gift" from the booming Internet. But it has double edges: it helps increasing scores but blindfolds us from realizing how much we believe the improvement is due to true technology advancement vs. more abundant data.[1] Thus I have only focused on the first three techniques: template matching, answer typing, tree/graph matching. Note that each one of them could be and has been individually studied. But any modern high-performance QA systems would inevitably need all three components (and others as well). Is there a general framework to incorporate them? Can we kick-start a com-

[1]That is one reason why later QA@CLEF switched the focus to machine reading a single document: the information retrieval front end was deliberately removed and answer redundancy did not help any more, see § 2.2.2 on page 35 for more information.

petitive system as a platform for future QA research? Can we do everything in a very timely manner so extensive feature engineering does not exhaust the passion of QA researchers? I proposed the feature-driven idea with discriminative Conditional Random Field (Lafferty et al., 2001) training in Chapter 3 to tackle exactly these problems.

The central idea of feature-driven is large-scale machine learning answer patterns from basic linguistic annotations. Learning from data does not only have the merit of freeing researchers from writing labor-intensive rules, but also capturing useful answer patterns that would have otherwise not been captured. For instance, we found out from the optimized features that, when answering where questions, there is a very high feature weight for words with the named entity label PERSON. We further discovered that our named entity recognizer often made the "mistake" of recognizing person names from location names, especially when the text was not wellformed (such as all lowercased, in which case the NER tool would lose an important cue for this: capitalization of first character). If it had been a system with manual answer type mapping, these kind of errors would have given the linguist a hard time and lower the performance. But the feature-driven idea is immune to this: errors in the NLP pipeline are captured and taken advantage of. In other words, the feature-driven idea cancels out the notorious phenomenon of *error propagation* in pipelined systems.[2] If we think very openly about this problem: NER labels such as PERSON and LOCATION are both *meaningful but misleading* symbols for a person. A person would naturally assume that LOCATION answers where questions and PERSON answers who questions but the truth is that it is not always the case.[3] Perhaps it is better if we replaced the output labels of NER tools with some meaningless symbols, then a person would have to resort to other means to draw the mapping for answer typing – very likely this other means would be the feature-driven approach.

The feature-driven approach also sheds light on how the information retrieval front end should work for the answer extraction back end. Previously keyword based information

[2]For instance, Peñas et al. (2013) observed that the upper bound for a QA system is about 60% F_1 due to error propagation in pipelined system through running several years of QA@CLEF.

[3]Maybe this explanation could rationalize what Fred Jelinek meant by his famous quote: "every time I fire a phonetician/linguist, the performance of the speech recognizer goes up " in the revolutionary work of modern statistical speech recognition he led at IBM in the 1980s: phoneticians are misled by the rules they *assumed* the data followed but not the rules the data *actually* followed.

retrieval was not coupled tightly with answer retrieval: information retrieval maximizes the hit rate between *what is known* in the query and the snippet (e.g., if you search cat, a search engine returns pages on cats), but answer retrieval instead asks for *what is unknown* in the query (e.g., if you search how many toes a cat has,[4] a search engine does not know to return a number for you). We proposed to use the learned features from the answer extraction back end to guide the search in the information retrieval front end. For instance, we have already learned from answer extraction that both the NER label of NUMBER and the POS tag of CD (for cardinal digit) help answer how many questions, then we could use this insight directly in information retrieval to search specifically for a number. What about the notorious error propagation in NLP pipeline affecting on the IR front end? Again, feature-driven information retrieval cancels it out. The section § 3.4 on page 110 (Structured Information Retrieval for Question Answering) illustrates this idea. Modern search engines are not the most direct and effective way for human-computer interaction: they just present possible web snippets for the answer seeker but do not solve the query directly. The feature-driven question answering technology points out one promising direction.

In recent years researchers diverted their interest from QA from text to QA from databases, partly due to the availability of more open-domain and practical knowledge bases such as Freebase and DBpedia. Even though the current volume of knowledge bases can only help solving about 5% of the problem,[5] they provide a reliable knowledge source for certain types of question in a structured and organized way. For instance, both Google and Microsoft use their own knowledge bases to answer some frequent queries on movies and famous people. Researchers combined the traditional semantic parsing approach with modern machine learning and developed the task into solving more open-domain problems on a much larger scale. The most notable dataset so far has been WebQuestions (Berant et al., 2013), which includes more than five thousand

[4]18: 5 on each front paw and 4 on each hind paw for most cats.

[5]This is based on estimation from two sources: 1. only 5% of the 1 million questions crawled from Google Suggest by Berant et al. (2013) have been identified answerable from Freebase by workers in Amazon Mechanical Turk. 2. Structured data sources (both automatically extracted from parses/rules and manually crafted such as Yago and DBpedia) contributed 2% $\sim$ 5% accuracy overall to the IBM Watson system (Fan et al., 2012, Kalyanpur et al., 2012a).

frequently asked questions crawled off the Google Suggest service that are answerable by Freebase. Most of these questions are fairly simple: they usually contain no or just one time or location constraint (e.g., who did x play for in 2011?). When answer source is changed from unstructured data (text) to structured (knowledge base), the feature-driven idea still applies well (Chapter 5). We turned the task of QA from knowledge base into a binary classification task on each relevant graph node. Each node carries features combining simple question analysis result and that node's graph relations and properties. In this way we automatically learn the most useful feature combinations from data. For instance, we learned a high feature weight for qfocus=money|type=Currency, meaning when the question is about money (e.g., what money to bring to Cuba?), nodes carrying the Currency property are likely to be the answer. We have shown that this approach is essentially learning the same thing as the much more complex semantic parsing method and the final results are comparable. The feature-drive idea again has shown its power on a different answer source.

To conclude, the feature-driven approach is both an overarching principle and a general framework. Under this framework I have demonstrated that:

- NLP QA (specifically, answer extraction from text) easily compares to the v0.1 version of IBM Watson based on the DeepQA technologies (Figure 1.1 on page 6 and § 4.6 on page 180).
- KB QA (specifically, answer extraction from Freebase) learns the same knowledge from data as the semantic parsing approaches, but is much less complicated (Chapter 5 on page 198).
- IR4QA (specifically, structured information retrieval guided by answer extraction) bridges the gap between information retrieval and question answering and serves information retrieval precisely for the purpose of question answering (§ 3.4 on page 110).
- Large-scale learning from the million-type feature space is possible: 35 million features were applied in NLP QA (§ 4.6.4.2 on page 186) and 7 million features in KB QA (§ 5.6.4 on page 221). The pure decoding time (excluding feature

annotation and extraction) for a Jeopardy question, for instance, can be reduced to less than 0.1 second when executed in parallel.

Adding more features to improve the end system is an engineering problem, but proposing the feature-driven framework to easily incorporate these features and demonstrating that it has worked effectively on all three important types of question answering is the research problem this dissertation has aimed to tackle.

6.1.2. Discriminative Methods for Monolingual Alignment

The second topic is aimed at providing a useful tool for monolingual alignment, with the potential for outreaching to other NLP tasks besides question answering.

I started with using a rule-based tree matching aligner based on Tree Edit Distance (§ 3.2 on page 90), inspired by the work of Heilman and Smith (2010) and Wang and Manning (2010). The aligner provided a mechanism to effectively incorporate tree matching information in the general framework of feature-driven methods. It also went beyond the sole tree matching methods for QA pioneered by Lin and Pantel (2001) and Cui et al. (2005). Ablation test (Figure 3.5 on page 108) showed that alignment-based features increased QA performance in F_1 by 10% on top of a strong answer typing model. But the TED aligner, similar to other rule-based aligners, has the limitation of extensibility. Meanwhile, there are many lexical resources for at least the English language that can be used for better alignment. A discriminative model would be a natural fit for this task.

We adapted the CRF-based model by Blunsom and Cohn (2006) to monolingual alignment for 1x1 token alignment (§ 4.3.1) and improved it with the semi-Markov CRF model (Sarawagi and Cohen, 2004) for many-to-many phrase alignment (§ 4.3.2 on page 154). We did extensive experiments on three datasets: MSR06 (Brockett, 2007), Edinburgh++ (Cohn et al., 2008, Thadani et al., 2012) and MTReference (newly created) while comparing against various aligners: GIZA++ (Och and Ney, 2003), TED (Yao et al., 2013d), Meteor (Denkowski and Lavie, 2011) and the MANLI family of aligners (MacCartney et al., 2008, Thadani and McKeown, 2011, Thadani et al., 2012). Our system outperformed these aligners on these three datasets. Moreover, detailed evaluation divided down by alignment types (identical vs. nonidentical, token vs. phrase) depicted a more clear picture of the current status for this task. The current state-of-the-art monolingual aligners have excellent performance on identical alignment (96% F_1), moderate performance on non-identical alignment (60% $\sim$ 70% F_1), good performance on token alignment (90% F_1) and very poor performance on phrasal alignment (40% F_1 or less). This will shed light on future research directions. The implementation system is jacana-align. It is open-source, platform independent, and ships with a bundle of lexical resources including PPDB (Ganitkevitch et al., 2013), Wiktionary, WordNet (Fellbaum, 1998, Snow et al., 2006) and word2vec models (Mikolov et al., 2013).

We then applied the aligner back to question answering, with the aim to justify the task of monolingual alignment in an NLP application. We did a set of large-scale experiments on the Jeopardy data in § 4.6 on page 180. First of all, with alignment features, the QA performance was improved 25% relatively in terms of average precision (39.45% vs. 49.09% in Figure 4.4 on page 187). Also the 9% difference in alignment F_1 between discriminatively trained jacana-align and rule-based TED (88.6% vs. 79.8% in Table 4.7 on page 170) translated into 1% difference in average end QA precision on the Jeopardy data (49.09% vs. 47.92% in Figure 4.4 on page 187), but the difference is statistically significant ($p = 0.006$).

The implications brought by jacana-align are more than just a 1% difference though: it is an aligner that automatically optimizes various features, has the potential to be extended with more lexical resources and re-trained on more data, and has the ability to recognize more paraphrases and entailment relations that otherwise rule-based systems are very limited of. I hope that this line of work could affect how researchers outside the QA community might tackle their problem differently than what they would have otherwise without the aligner.

6.2. Take-home Messages

Question answering is a very complicated task. Here are a few messages I have been trying to deliver in this dissertation, ranging from high-level to more specific:

- Machine learning works its magic. Log-linear models are in general good fit for a lot of subtasks in question answering, such as answer ranking and answer extraction.
  - Log-linear models support various feature regularization and have a nice adjustable prediction threshold in $[0, 1]$. Learned feature weights usually can be easily interpreted by its value.
  - Throughout my work, I have used CRFs for answer extraction from text, logistic regression for answer sentence ranking and answer extraction from Freebase. I have tried other algorithms and models such as perceptron and SVM but they did not work better.
  - Answer ranking in Watson also used logistic regression, after trying with other techniques such as SVMs, boosting, neural nets and decision trees.
- The KISS principle works its magic. Always start with a small and simple system that is very fast to be tested. In my early QA work (§ 3.3.4.1 on page 105) an entire round of feature extraction, training on TRAIN and testing on DEV took less than one minute. One hour later I would have tried 30 different things to find the "sweet spot" of the system on the data I was testing on.[6] Tuning features is somehow even more important than tuning the model.
- Acknowledge the imperfections of NLP pipeline and do not waste time to correct them. Error propagation in pipelined systems is a nasty thing to deal with. We can never make each component in the pipeline perfect. But we can use the fact that NLP tools always make *consistent* errors. The feature-driven idea makes use of this fact: imperfect feature annotations are encoded in learning and decoded during test so the errors in these features would cancel out.
- The feature-driven framework can be the statistical backbone for QA systems. It is a very simple idea but has been shown successful in three very important components in the QA system: information retrieval, answer extraction from text, and answer extraction from knowledge base.

[6]In retrospect, this is quite similar to the AdaptWatson methodology used by the Watson team.

- Current information retrieval techniques do not satisfy the need for question answering. Structured retrieval as early as the document retrieval stage tackles this problem.
- Semantic parsing is not necessarily the best solution for *simple* web queries on Freebase. Feature-driven information extraction technique so far has worked well.
- The major answer source for question answering is still text, see footnote 5 on page 238.
- Monolingual alignment for the English language has matured in both theory and practice, with off-the-shelf open-source tools. It works the best in applications that require precise alignment. I highly recommend jacana-align for this task.
- Human annotations on monolingual alignment tasks can be heavily influenced by automatic pre-alignment. The current datasets also lack proper annotation of phrase boundaries. In the future should more annotated alignment dataset be created, it is suggested to make a focus on annotating more phrase alignment.
- Linguistic features drawn from alignment between QA pairs help boost QA end performance significantly.

6.3. Future Directions

I have listed future directions at the end of each chapter: § 3.5 on page 132, § 4.8 on page 195 and § 5.8 on page 234. In terms of big ideas and important research problems I think the following directions are worth exploring. Some of the goals are quite ambitious (but not as ambitious as, for instance, "solve AI"). But I imagine seeing them accomplished in one's lifetime.

- Web-scale structured information retrieval over text. Current computing infrastructure already has the ability to parse the whole web and add various layers of linguistic annotations on top of the raw text. It would be helpful for QA systems to be able to search, for instance, a country *class* instead of specific *instances* of country. I imagine queries such as "Toronto [COUNTRY]" retrieve the names of all countries that might have a place called Toronto. I also imagine density based IR techniques can be improved from density closeness with word orders to density closeness with syntactic paths, so that "long range" dependencies are not treated as long-range any more in density estimation.

- Information retrieval over structured data. On the WebQuestions dataset simpler information extraction methods performed as well as semantic parsing methods. Thus we called for more complicated questions to show the power of semantic parsing. However, note that the best performing system achieved less than 50% F_1. Obviously introducing more complicated questions would not improve this number. On the other hand, semantic parsing methods really nailed more compositional questions on other much smaller domains: the current state of the art in domains like Geo880 and Jobs640 have more than 90% accuracy. Thus I suspect that the problem of low accuracy in WebQuestions is due to very big search space from Freebase: it contains more than 40 million topics and 2.5 billion facts. In my experiment each Freebase topic contains 1000 nodes as potential answer candidates. While text-based QA can use the help from search engines, in KB-based QA there has not been a satisfying solution for effectively and efficiently narrowing down this search space.[7] This calls for information retrieval over structured data sources.

- Deeper understanding of text. So far we have put a lot of focus on question understanding: the structure of questions is analyzed; topic/focus words are extracted; deeper semantics and logic forms are obtained. How about the other half of the QA equation, the answer source? Our understanding of text snippets is not to the level of what we have understood about the question.

- Going contextual and conversational. A QA system should have at least short term memory: it should remember what questions were asked one or several turns

[7]The Freebase Search API only returns relevant topic nodes (similar to "document" retrieval) but does not rank nodes within the topic (analogous to no "passage" retrieval).

ago and it should be able to track topic changes. This is a key factor in creating conversational QA systems.

- Answering questions in ill-formed text. Automatically transcribed speech can be difficult to parse and understand, so is typed input from a smaller keyboard only a few inches in size. I imagine a QA system answer questions based on the n-best output from a speech recognition lattice, or several possible typing corrections based on the n-best proposals of a language correction model.
- Recognize how questions can be asked differently. Humans contribute answers (community-based QA) to a lot of questions on the web (such as WikiAnswers, Yahoo Answers, Quora.com). For questions already coupled with an answer, we need to recognize (maybe through monolingual alignment) how a new question is essentially asking the same thing.
- Finally: replace search. Search engines are not the best solution for human-computer interaction. They are even worse solutions for interaction on mobile devices. QA should replace search in the future. It is as simple and as hard as that.

7. Curriculum Vita

Xuchen Yao was born in Feixian County of Linyi City, Shandong Province of China on April 7th, 1984. He attended Nanjing University and graduated with Bachelor of Science in Acoustics in 2006. He studied and worked on Speech Recognition for a year at Institute of Acoustics, Chinese Academy of Sciences and then Speech Synthesis at Nokia Research Center Beijing for another year. From 2008 to 2010, he joined the European Masters Program in Language and Communication Technologies and graduated with Master of Arts in Linguistics from University of Groningen, the Netherlands (advisor: Gosse Bouma), and with Master of Science in Language Science and Technology from Saarland University, Germany (advisors: Hans Uszkoreit and Yi Zhang). His Master thesis was on *Question Generation with Minimal Recursion Semantics*. In 2010 he started pursuing a Computer Science PhD degree at the Johns Hopkins University under the supervision of Benjamin Van Durme and Chris Callison-Burch.

Xuchen Yao was a visiting student/intern at the following academic institutions:

- Institute for Creative Technologies, University of Southern California, with Anton Leuski, Kenji Sagae, Kallirroi Georgila and David Traum
- City University of Hong Kong, with Chunyu KIT
- Information Science Institute , University of Southern California, with David Chiang

He also interned at EnReach, Motorola, Nokia, Vulcan (now the Allen Institute for Artificial Intelligence, with Peter Clark) and Google Research (with Dekang Lin).

A. Examples of GIZA++ vs. jacana-align on 3 Alignment Datasets

This appendix chapter shows some concrete examples of the following three aligners described in Chapter 4:

1. GIZA++
2. the token version of jacana-align
3. the phrase version of jacana-align

on the following three datasets of different characteristics:

	length	**%1x1 align**	**GIZA++**	**GIZA++**	**jacana-align**
			pre-aligned	$\mathbf{F_1}$	$\mathbf{F_1}$
MSR06	29/11	96%	No	78.3	88.3/87.3
Edinburgh++	22/22	95%	Yes	85.5	86.7/85.9
MTReference	22/17	88%	Yes	77.4	77.1/77.4

- MSR06: not pre-aligned by GIZA++, with a majority of 1x1 alignment.
- Edinburgh++: pre-aligned by GIZA++, with a majority of 1x1 alignment.

- MTReference: pre-aligned by GIZA++, with more one-to-many and many-to-many alignment.

The purpose of this comparison is to show by examples the alignment difference between a bilingual aligner (GIZA++) and a monolingual aligner (jacana-align). Among the family of open-source monolingual aligners, there is a close competitor: Meteor. We show the alignment difference between Meteor and jacana-align in the next appendix chapter.

In the following we select two to three examples from each dataset and show two pictures per example:

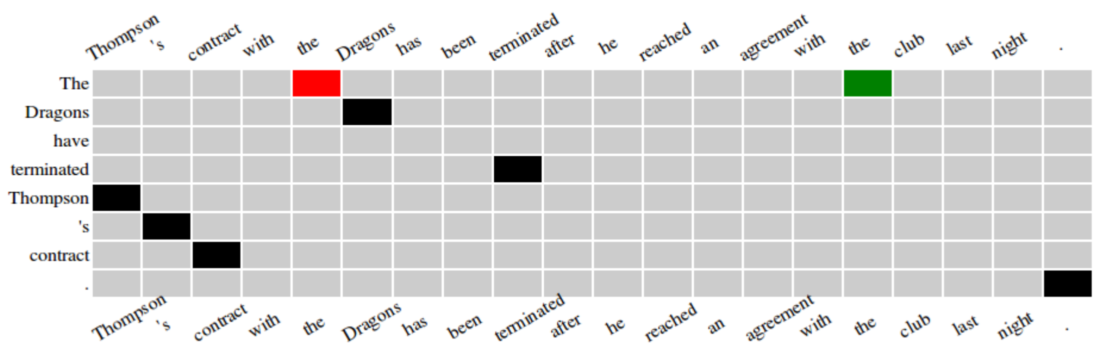
1. The first picture compares alignment between the gold standard and GIZA++, with color codes:
    - black marks alignment from both aligners;
    - red marks alignment from only the gold standard;
    - green marks alignment from only GIZA++.
2. The second picture compares alignment between the token and phrase version of jacana-align, with color codes:
    - black marks alignment from both aligners;
    - red marks alignment from only the token aligner;
    - green marks alignment from only the phrase version.

A.1. MSR06

Example 1 Alignment between the following two sentences:

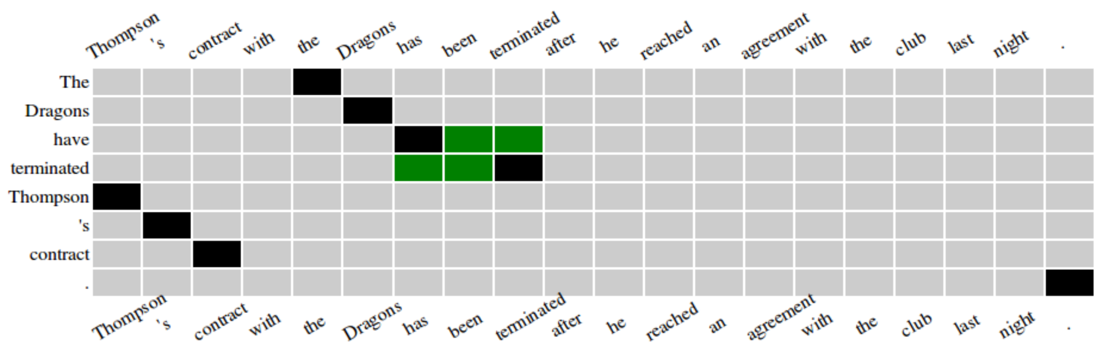
- The Dragons have terminated Thompson 's contract .
- Thompson 's contract with the Dragons has been terminated after he reached an agreement with the club last night .

Gold standard vs. GIZA++ (black: both; red: gold standard only; green: GIZA++ only):



In the above example GIZA++ misaligned the stop word the.

jacana-token vs. jacana-phrase (black: both; red: jacana-token only; green: jacana-phrase only):

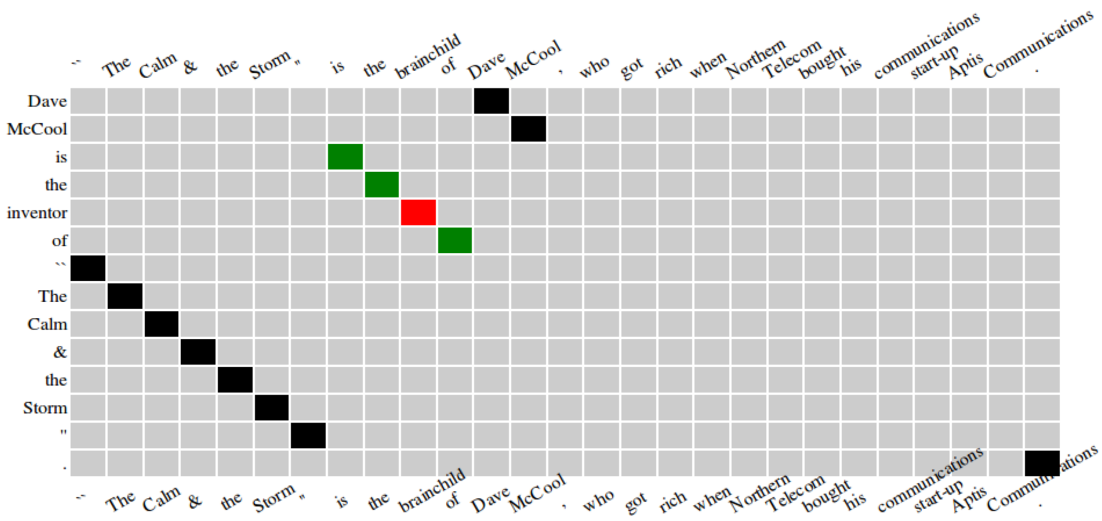


In the above example jacana-phrase misaligned the phrase have terminated↔has been terminated. On the other hand jacana-token did a fairly good job.

Example 2 Alignment between the following two sentences:

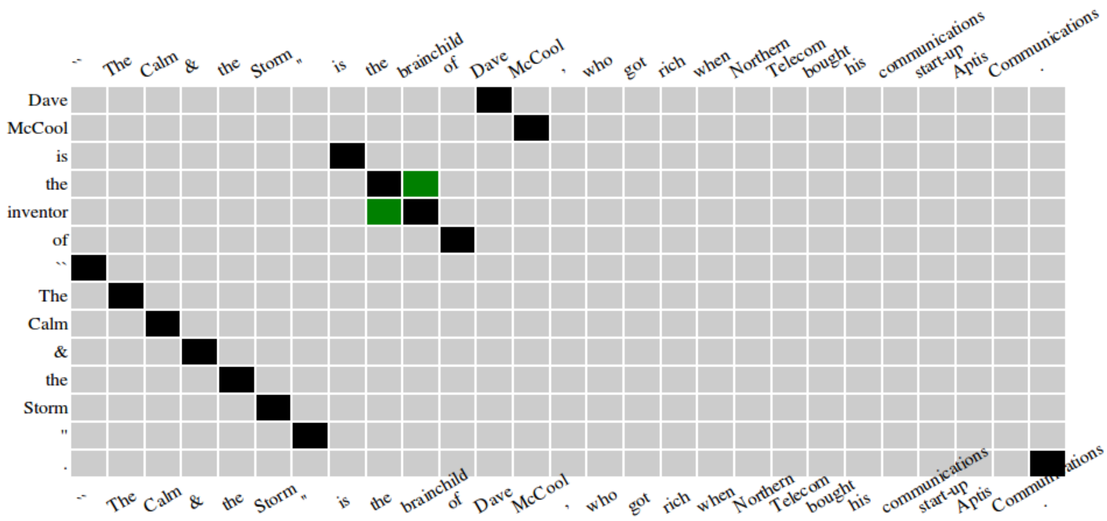
- Dave McCool is the inventor of " The Calm & the Storm " .
- " The Calm & the Storm " is the brainchild of Dave McCool , who got rich when Northern Telecom bought his communications start-up Aptis Communications .

Gold standard vs. GIZA++ (black: both; red: gold standard only; green: GIZA++ only):



In the above example for the red and green colored alignment, human annotators only aligned inventor↔brainchild, which is strictly not paraphrase of each other but still somehow related to the context. GIZA++ aligned the surrounding contextual stop words but unfortunately did not align inventor↔brainchild.

jacana-token vs. jacana-phrase (black: both; red: jacana-token only; green: jacana-phrase only):

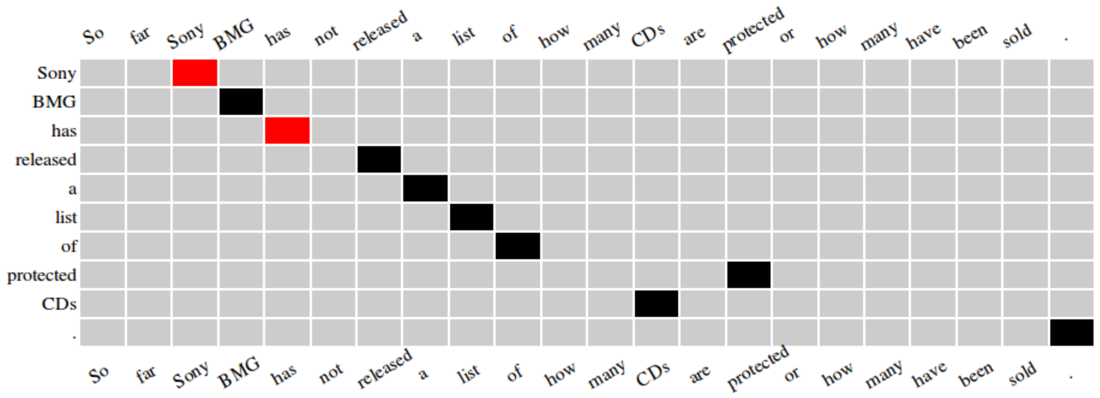


In the above example jacana-token aligned the whole token sequence: is the inventor of↔is the brainchild of. The words inventor and brainchild did not appear in its paraphrase table. However, the contextual and Markovian features gave jacana-token enough confidence to align them anyways since all their surrounding words were aligned. jacana-phrase further aligned the phrase block the inventor↔the brainchild.

Example 3 Alignment between the following two sentences:

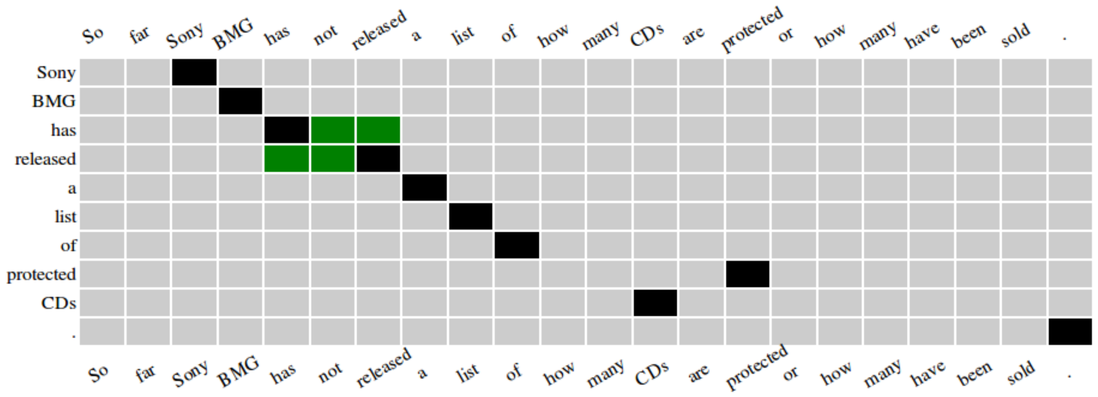
- Sony BMG has released a list of protected CDs .
- So far Sony BMG has not released a list of how many CDs are protected or how many have been sold .

Gold standard vs. GIZA++ (black: both; red: gold standard only; green: GIZA++ only):



It was not clear to me why GIZA++ did not align Sony↔Sony and has↔has. It could likely be that the GIZA++ result shown here was after applying the INTERSECTION heuristics in both directions and these words were not aligned in both directions.

jacana-token vs. jacana-phrase (black: both; red: jacana-token only; green: jacana-phrase only):



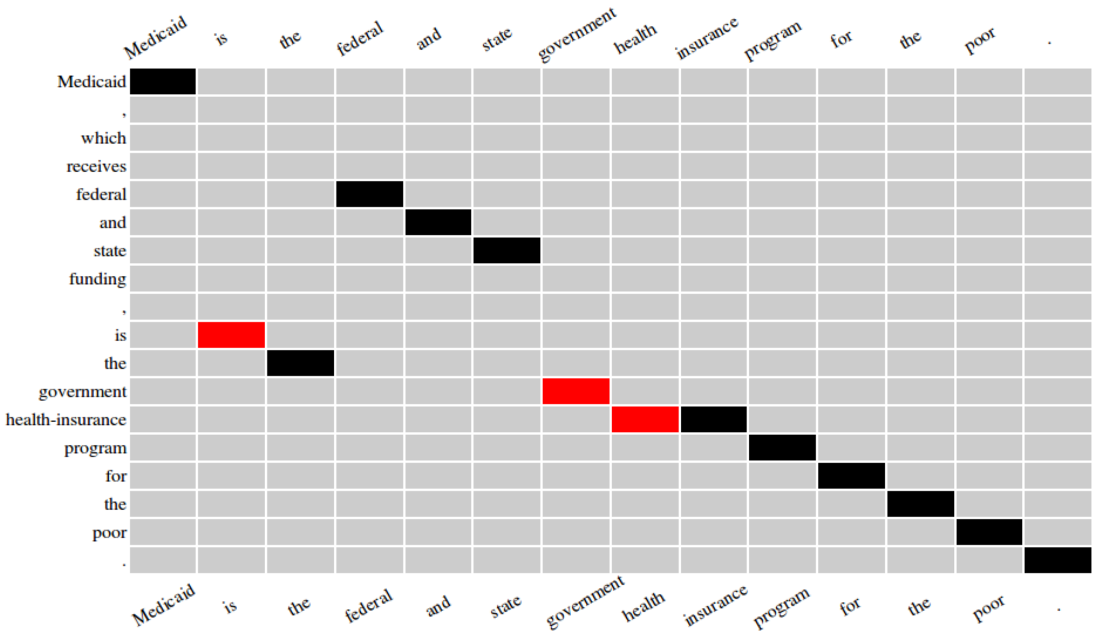
jacana-phrase here made a bad alignment: has released↔has not released. The reason was that both of them were verb phrases recognized by the chunker and they were long enough to have a high string similarity score.

A.2. Edinburgh++

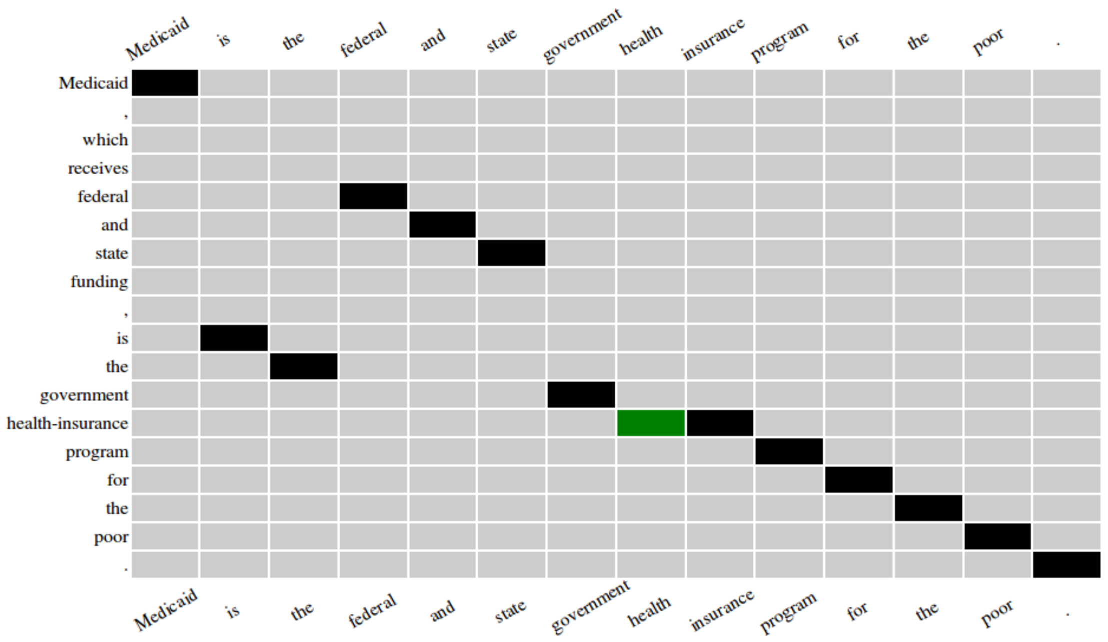
Example 1 Alignment between the following two sentences:

- Medicaid , which receives federal and state funding , is the government health-insurance program for the poor .
- Medicaid is the federal and state government health insurance program for the poor .

Gold standard vs. GIZA++ (black: both; red: gold standard only; green: GIZA++ only):



jacana-token vs. jacana-phrase (black: both; red: jacana-token only; green: jacana-phrase only):

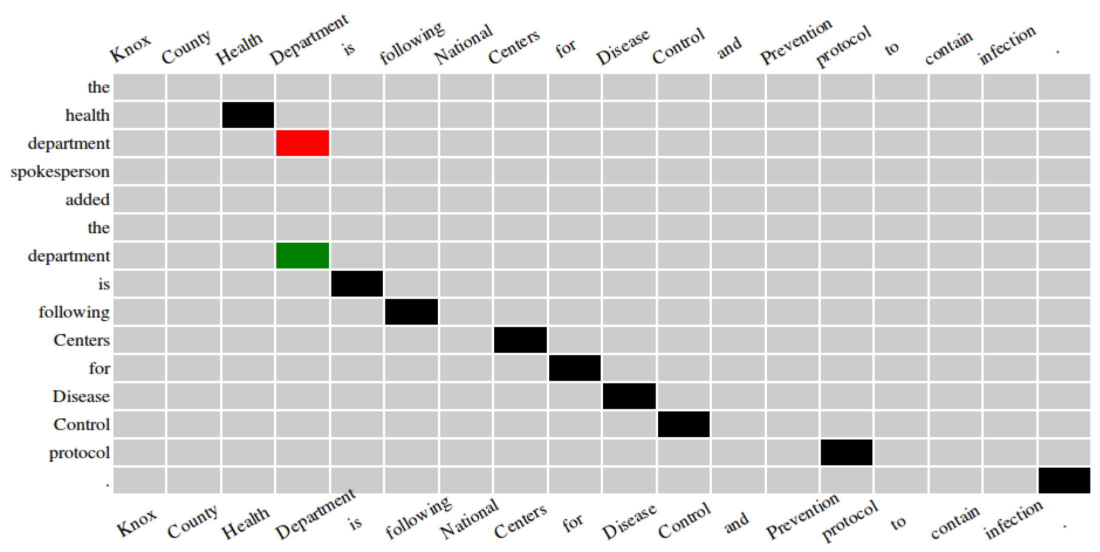


jacana-phrase made a perfect alignment in this example. It successfully recognized the hyphened compound alignment health-insurance↔health insurance.

Example 2 Alignment between the following two sentences:

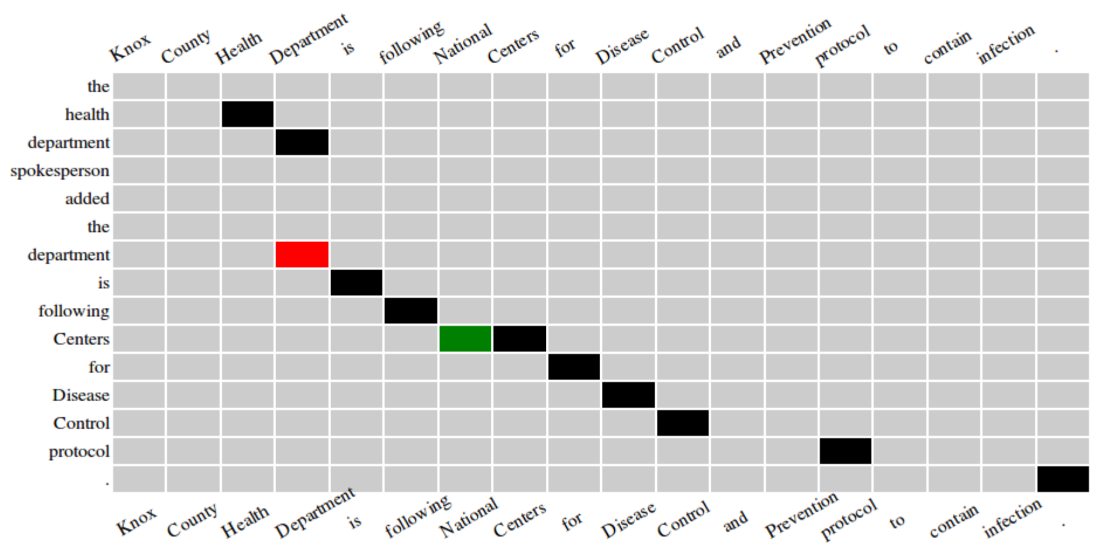
- the health department spokesperson added the department is following Centers for Disease Control protocol .
- Knox County Health Department is following National Centers for Disease Control and Prevention protocol to contain infection .

Gold standard vs. GIZA++ (black: both; red: gold standard only; green: GIZA++ only):



GIZA++ placed a wrong alignment in department↔Department. It might have been confused by the distortion feature since both tokens of department have their neighbors aligned.

jacana-token vs. jacana-phrase (black: both; red: jacana-token only; green: jacana-phrase only):



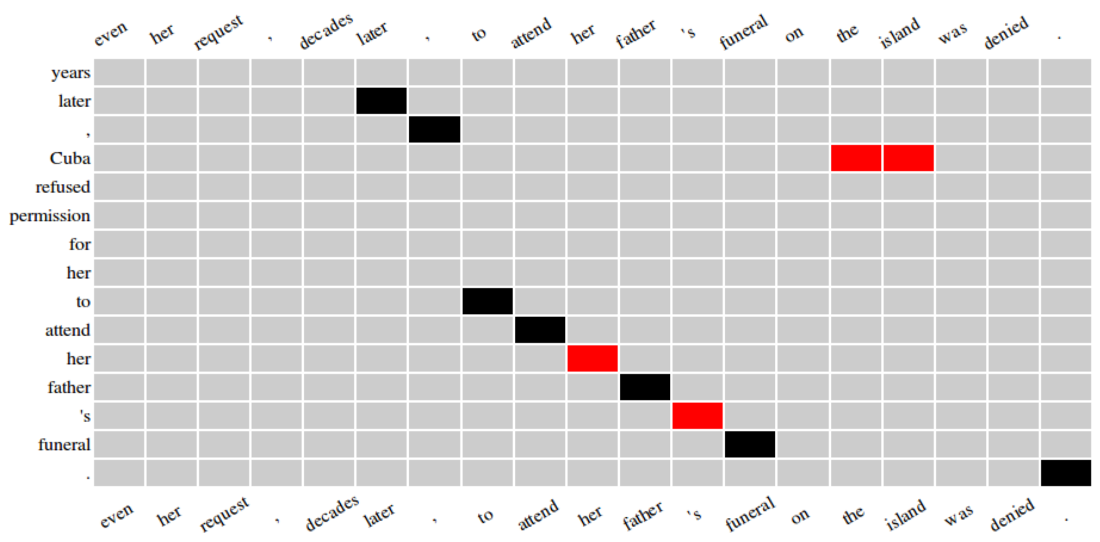
jacana-token made two alignments between department↔Department. This is allowed since the model supports one-to-many alignment. The alignment made by jacana-phrase was reasonable.

Example 3 Alignment between the following two sentences:

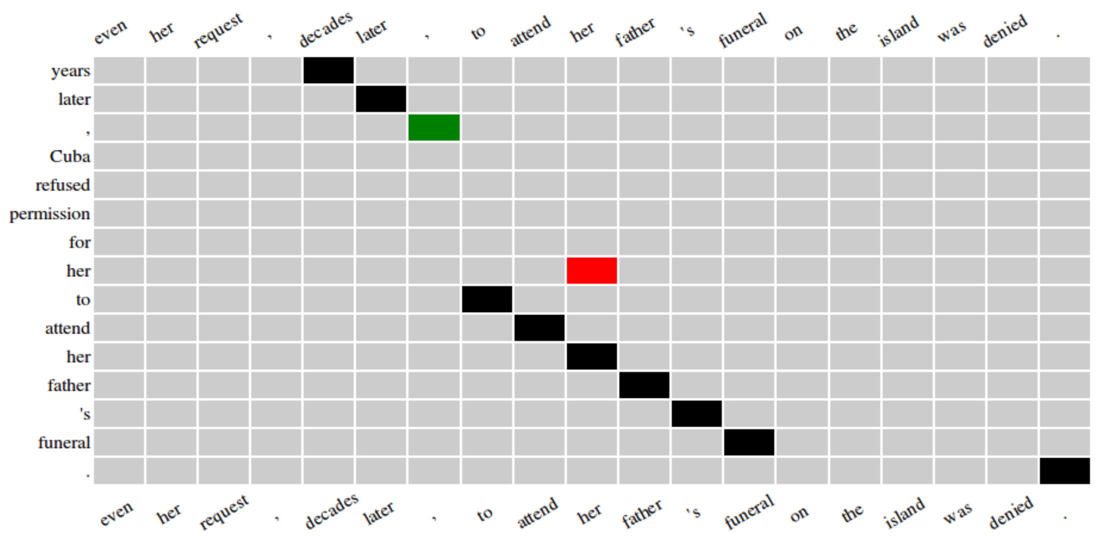
- years later , Cuba refused permission for her to attend her father 's funeral .

- even her request , decades later , to attend her father 's funeral on the island was denied .

Gold standard vs. GIZA++ (black: both; red: gold standard only; green: GIZA++ only):



jacana-token vs. jacana-phrase (black: both; red: jacana-token only; green: jacana-phrase only):



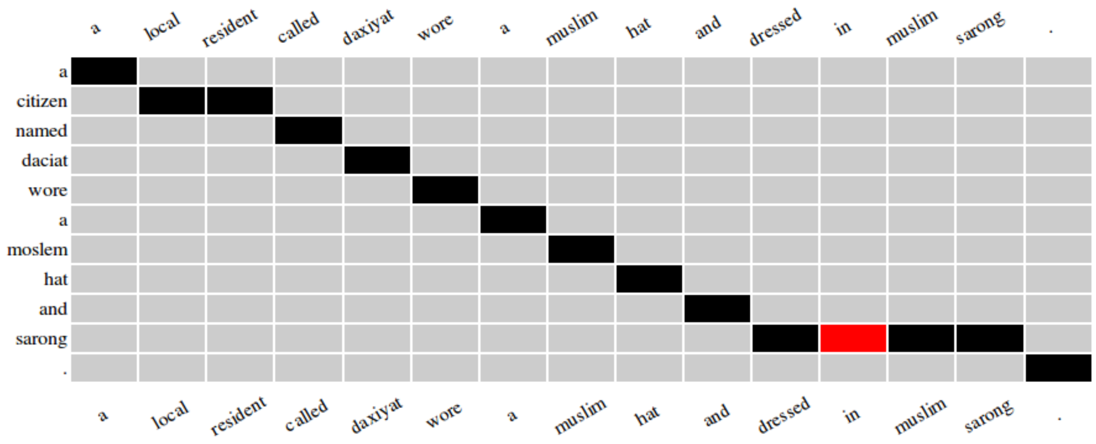
jacana-token made two alignments between her↔her, since it is inherently a one-to-many aligner. All three aligners did not recognize Cuba↔the island, which was a difficult case.

A.3. MTReference

Example 1 Alignment between the following two sentences:

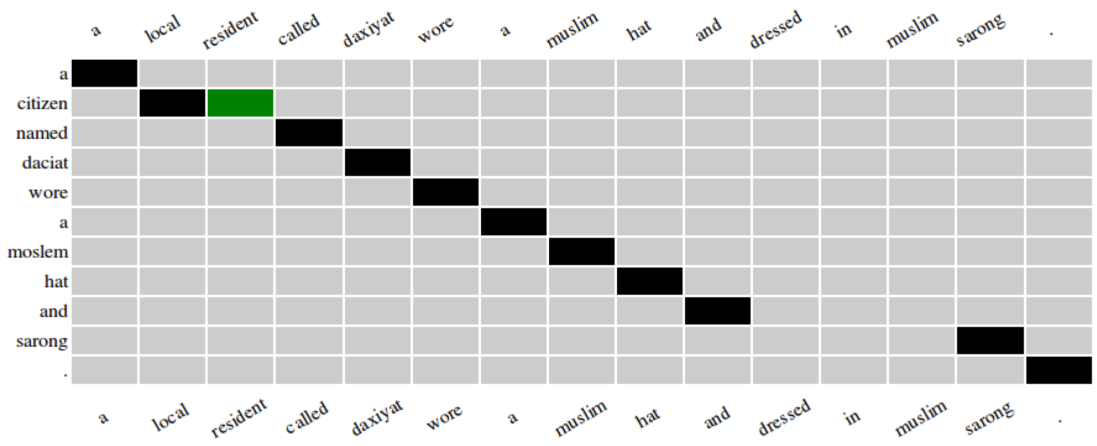
- a citizen named daciat wore a moslem hat and sarong .
- a local resident called daxiyat wore a muslim hat and dressed in muslim sarong .

Gold standard vs. GIZA++ (black: both; red: gold standard only; green: GIZA++ only):



This example showed how human annotators were influenced by the prealignment from GIZA++. Originally GIZA++ aligned sarong↔dressed ~~in~~ muslim sarong but left out the stop word in. Human annotators had two choices: either only align sarong↔sarong, or align the whole sequence sarong↔dressed in muslim sarong. They chose the latter option.

jacana-token vs. jacana-phrase (black: both; red: jacana-token only; green: jacana-phrase only):

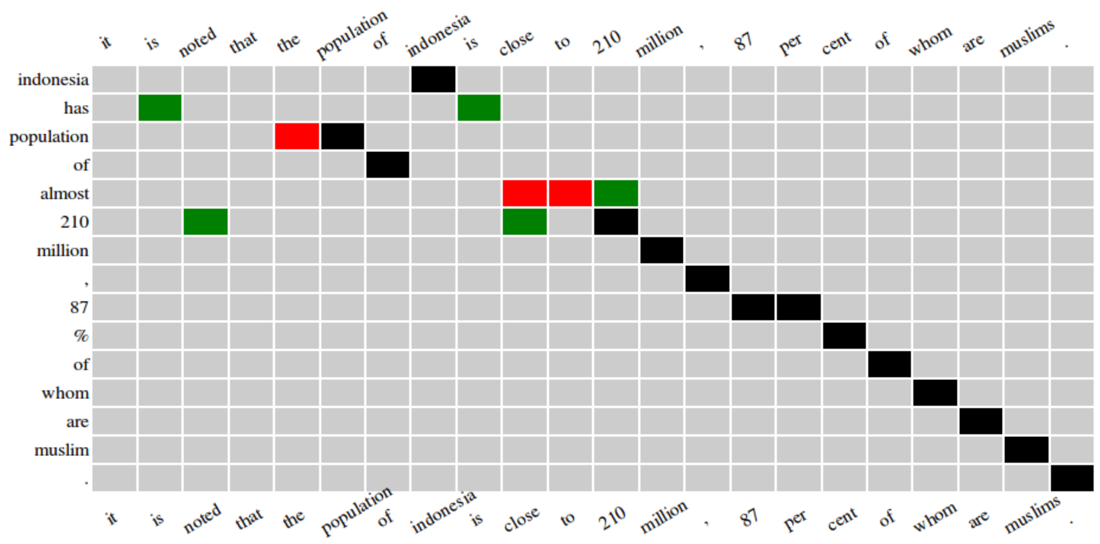


Both jacana-token vs. jacana-phrase did not recognize sarong↔dressed in muslim sarong but only aligned sarong↔sarong. jacana-phrase additionally aligned the 1x2 phrase: citizen↔local resident. But why did it not align the whole phrase a citizen↔a local resident? I think it could be that both of the determiners a appeared at the very beginning of the sentences. There was a strong feature that encourages 1x1 alignment at the beginning of the sentence.

Example 2 Alignment between the following two sentences:

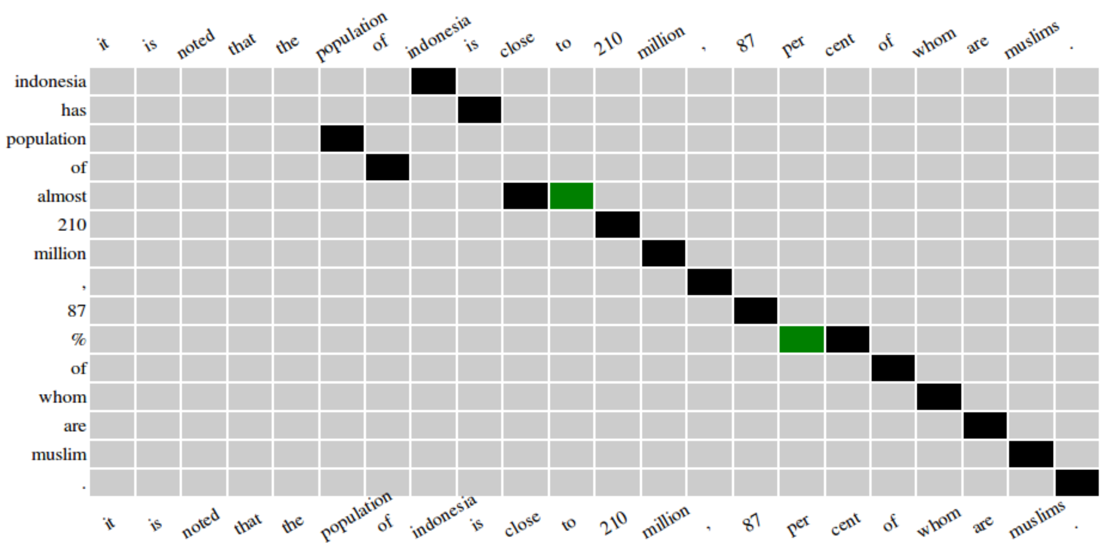
- indonesia has population of almost 210 million , 87 % of whom are muslim .
- it is noted that the population of indonesia is close to 210 million , 87 per cent of whom are muslims .

Gold standard vs. GIZA++ (black: both; red: gold standard only; green: GIZA++ only):



Turkers corrected quite some errors made by GIZA++: has↔is, 210↔noted, 210↔close, and also improved from almost 210↔210 to almost 210↔close to 210. Some either "better disguised" errors were not recovered, such as that 87 %↔87 per cent was annotated as 87↔87 per and %↔cent.

jacana-token vs. jacana-phrase (black: both; red: jacana-token only; green: jacana-phrase only):

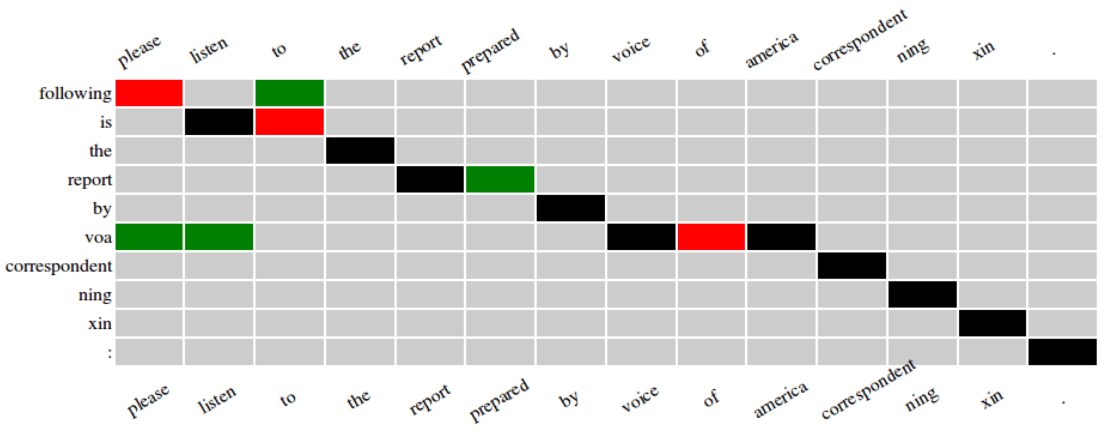


jacana-phrase successfully recognized almost↔close to and corrected the alignment between %↔per cent, even though it would be judged wrong according to the erroneous gold standard.

Example 3 Alignment between the following two sentences:

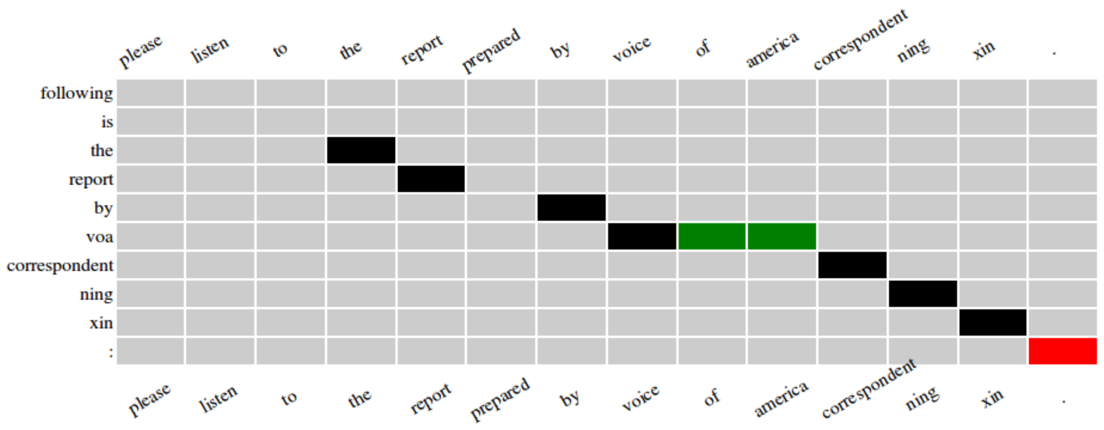
- following is the report by voa correspondent ning xin :
- please listen to the report prepared by voice of america correspondent ning xin .

Gold standard vs. GIZA++ (black: both; red: gold standard only; green: GIZA++ only):



It was not clear why Turkers would align following↔please and is↔listen to. But they did improve the alignment between voa↔voice of america.

jacana-token vs. jacana-phrase (black: both; red: jacana-token only; green: jacana-phrase only):



jacana-token only aligned voa↔voice because they looked similar in the first two characters. jacana-phrase on the other hand successfully recognized this acronym.

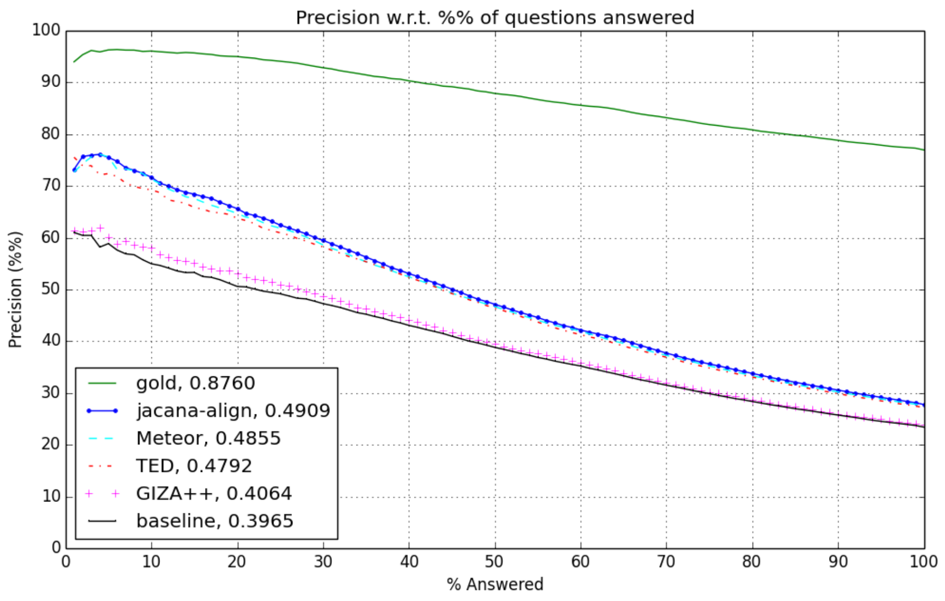
A.4. Summary

This appendix chapter shows some concrete examples of GIZA++ and jacana-align (token/phrase version) on three datasets. I have shown that:

- how human annotators were influenced by the pre-alignment of GIZA++;
- GIZA++ sometimes made alignment that cannot be clearly explained why, but probably due to the way it computed lexical co-occurrence matrices and distortion probabilities;
- the token version of jacana-align was precise most of the time;
- the phrase version of jacana-align made reasonable phrasal alignment *sometimes*, due to that the datasets were still mostly filled with token alignment and thus the phrase aligner was conservative in making phrase alignment.

B. Examples of jacana-align vs. Meteor on Jeopardy!

In Figure 4.4 on page 187 I presented the performance of jacana-qa with alignment features drawn from different aligners and the significance test. For ease of reading, the figure is redrawn here:

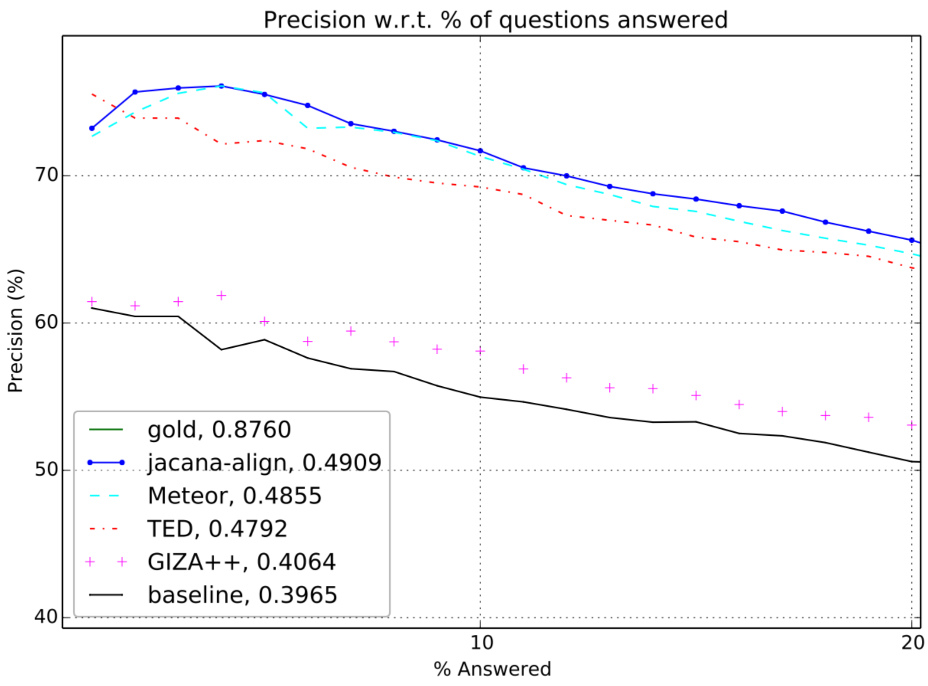


The significance test is:

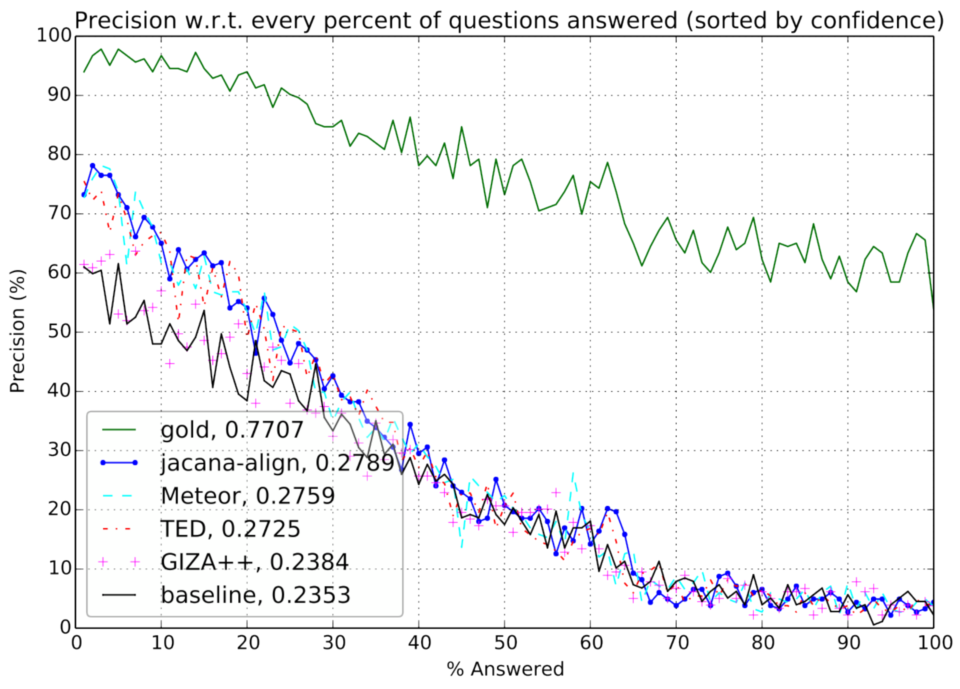
null hypothesis	p
jacana-align vs. Meteor	0.039
jacana-align vs. TED	0.006
jacana-align vs. baseline	0.000
Meteor vs. TED	0.230
GIZA++ vs. baseline	0.000

Note that the difference between jacana-align and Meteor is very close, and the p-value (0.039) is barely significant. Thus it is interesting to compare jacana-align and Meteor in detail.

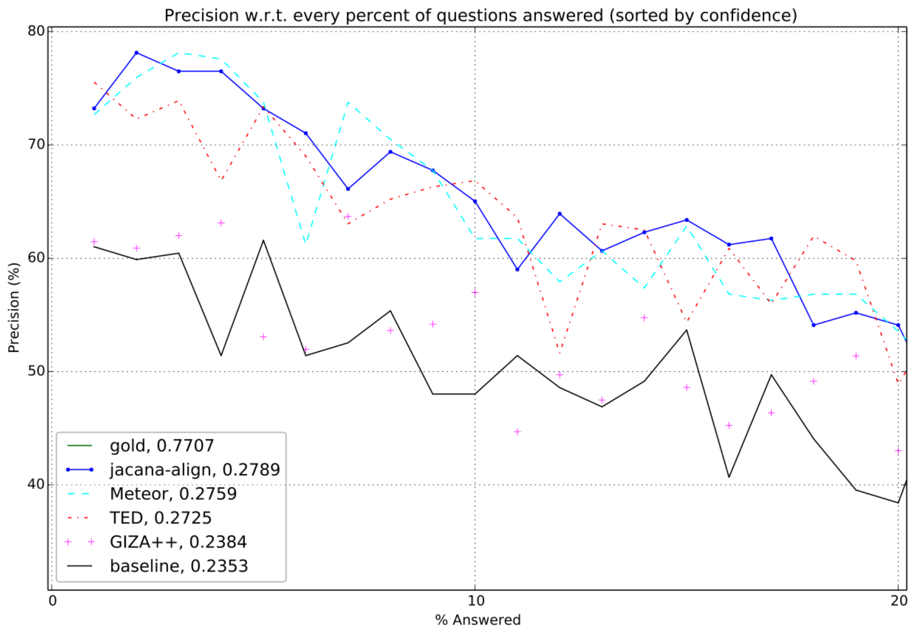
One cue of statistical significance is that the curve of jacana-align always lies above the curve of Meteor: even though the difference is close, but it is constant. The above curve zoomed into the top 20% answered questions is:



The curve of jacana-align still constantly lies above or overlaps with the curve of Meteor. However, this is a "cumulative" precision-recall curve: for instance, precision calculated at 20% recall also includes correctly answered questions at 10% recall. Thus it depicts the precision values *up to* a recall value. The following is a picture of precision values *at* each recall value, where each point stands for precision values for 1% of all test questions (18, 725 in all):

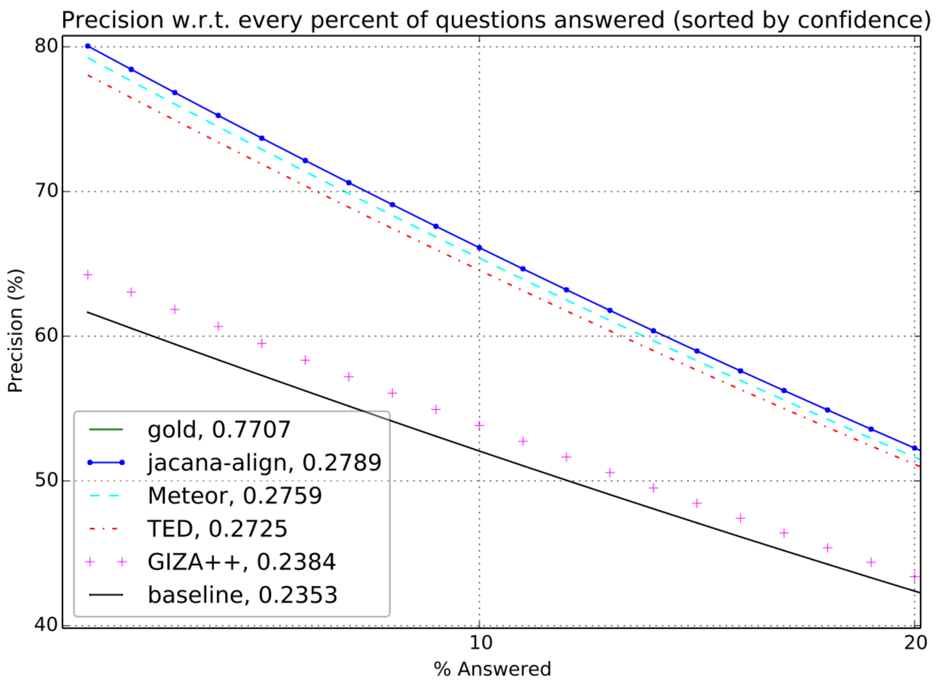


Again, zoomed-in to the top 20% of questions:



The curve is bumpy because precision at each percent of questions is not smoothed.

For smoothing, I fit the curve with a 2-degree polynomial function and zoomed in to the top 20% again:



The fitted curve of jacana-align still constantly lies above with the fitted curve of Meteor. Next I give some concrete examples.

B.1. jacana-align Helped Answering

Given the following clue and answer:

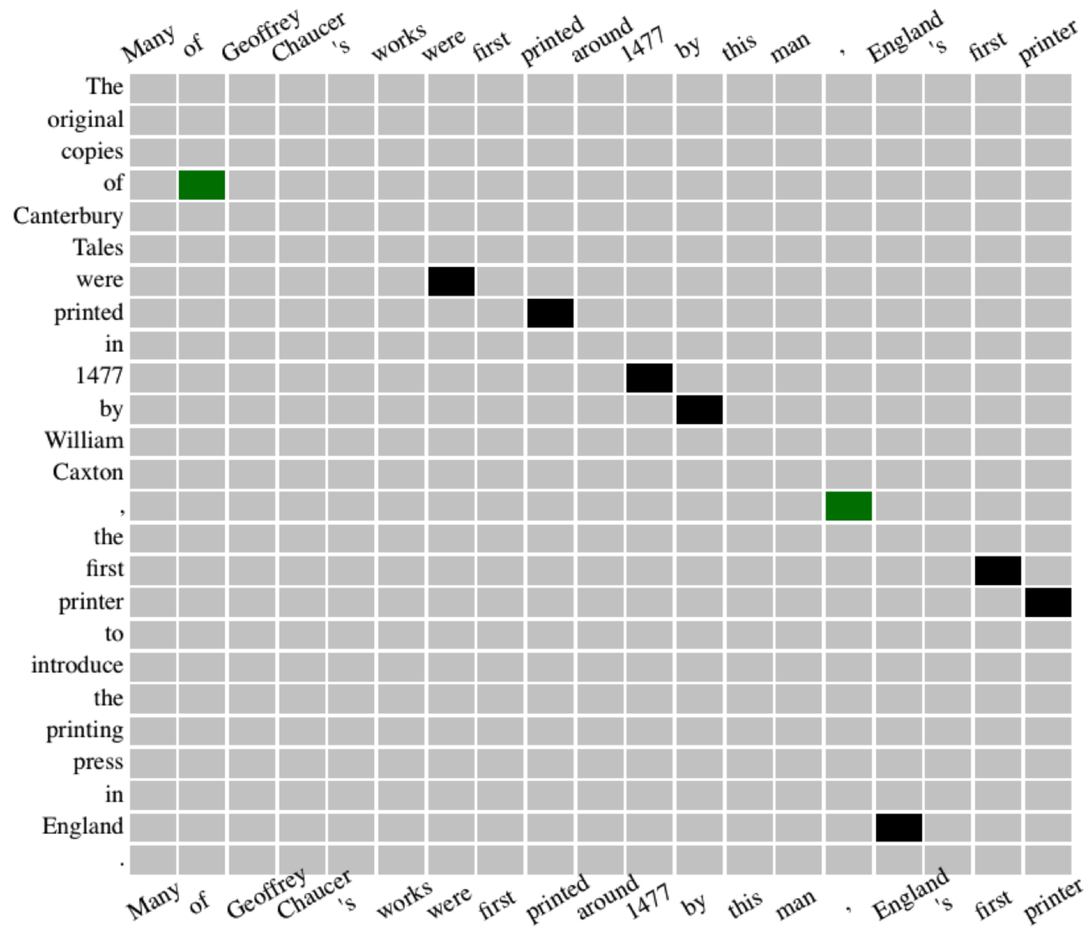
- Clue: Many of Geoffrey Chaucer 's works were first printed around 1477 by **this man** , England 's first printer .
- Answer: William Caxton

jacana-qa paired with jacana-align was able to propose the correct answer, but not with Meteor. In the following I show some alignment matrices with overlapping output from jacana-align and Meteor. The color codes are:

- black marks alignment from both aligners;
- red marks alignment from only jacana-align;

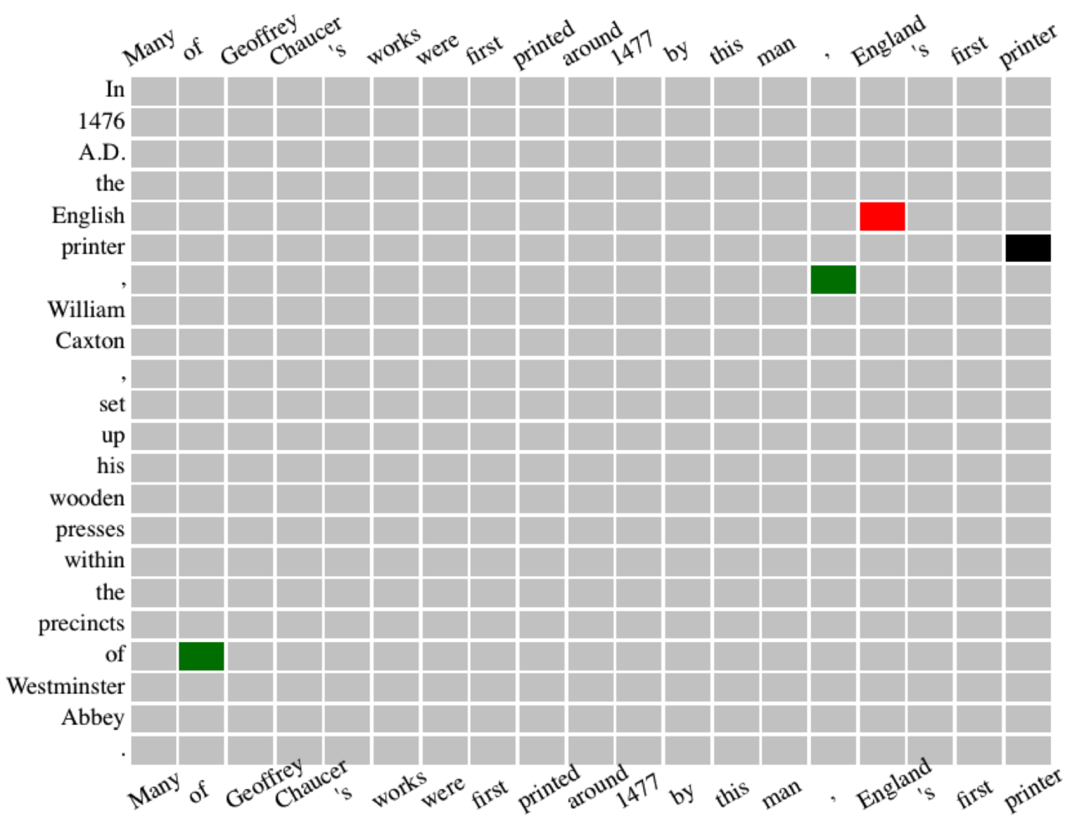
- green marks alignment from only Meteor.

The following is an alignment pair between the clue and a candidate sentence:

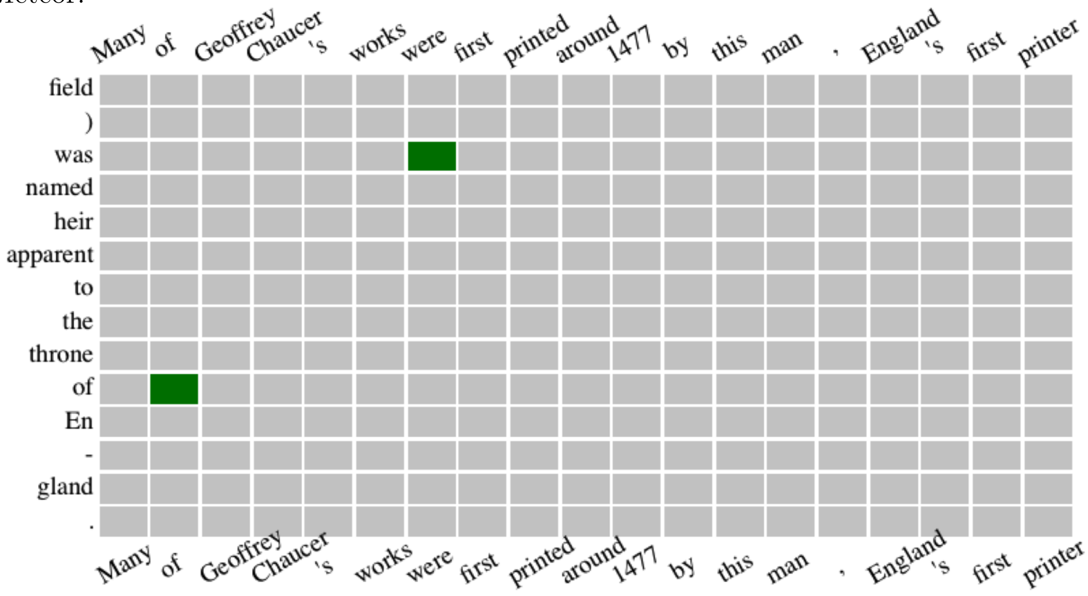


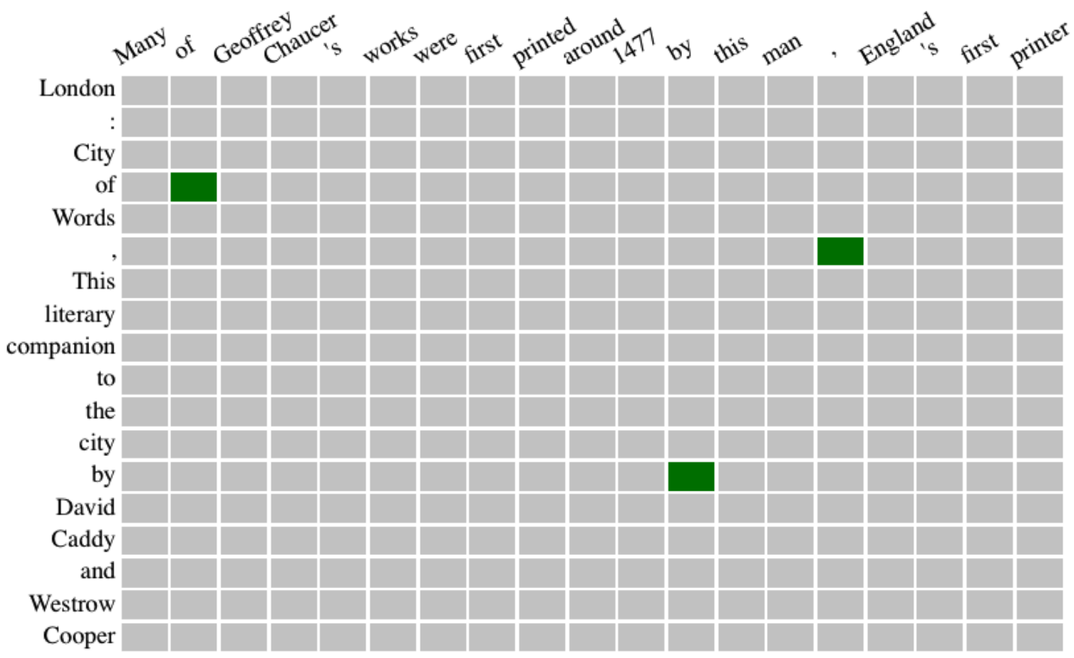
Both aligners successfully aligned the "important" content words in the pair, such as England, first printer. More importantly, the preposition by in the clue (by this man) was also aligned to the preposition by in the sentence (by William Caxton). This is an important cue for extracting the correct answer William Caxton.

In the above matrix, Meteor also aligned two extra light words: of and a comma ,. The following examples confirmed Meteor's tendency to over-align light words:



Again, Meteor aligned of and a comma. jacana-align additionally recognized England↔English. The following two pictures show an extreme case of over-alignment from Meteor:



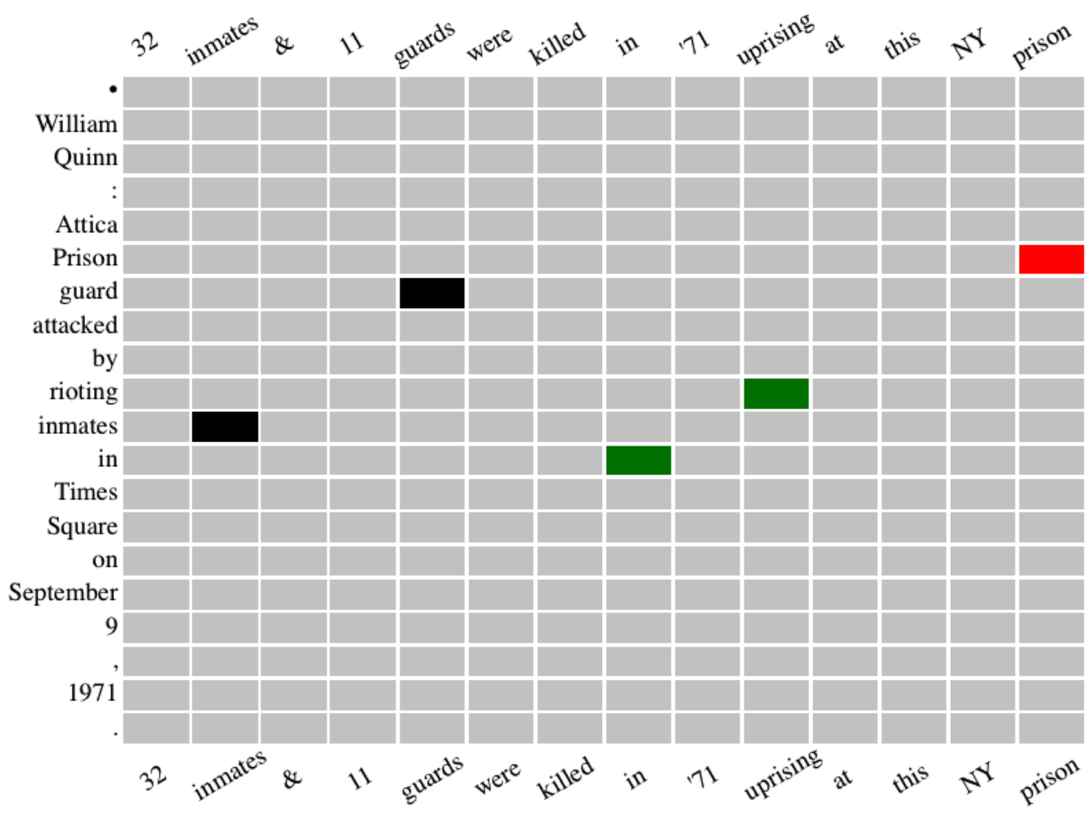


In these two examples, the search sentences are barely relevant to the clue. By aligning some of the common light words, Meteor gave jacana-qa a false signal that those sentences are relevant to a degree. jacana-align, on the other hand, simply chose not to align anything, due to its global decoding inference.

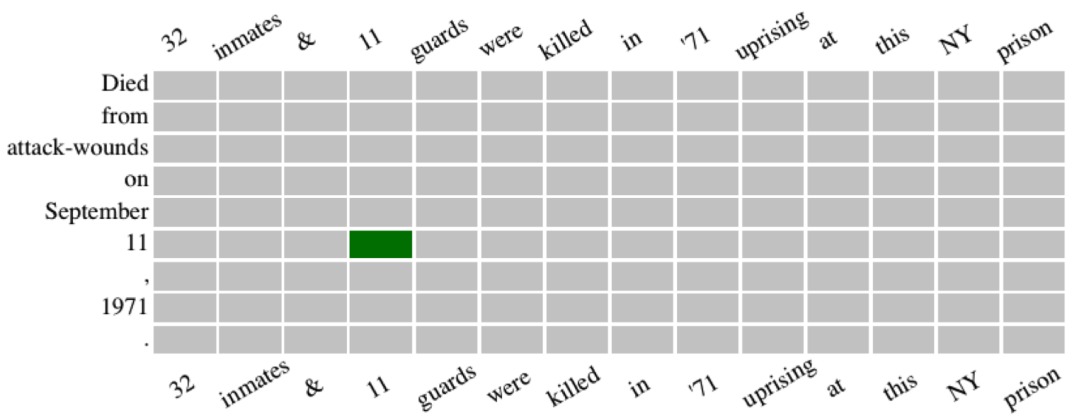
Meteor optimizes alignment by trying to minimize cross-alignment between words. Sometimes it can be problematic. For instance, given the following clue:

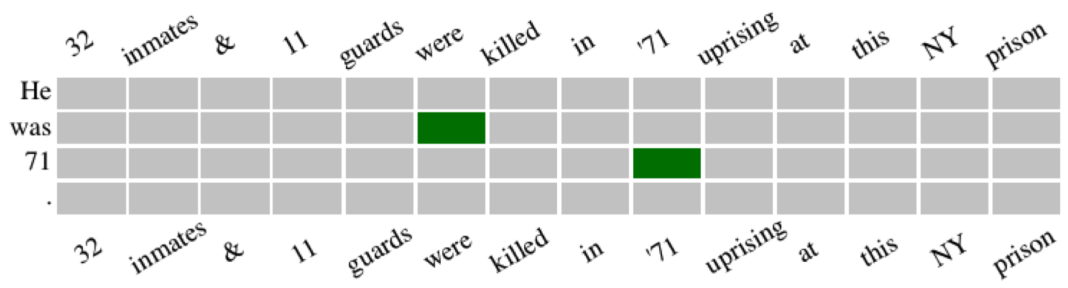
- Clue: 32 inmates & 11 guards were killed in '71 uprising at **this NY prison** .
- Answer: **Attica**

in the following picture, Meteor failed to align the crucial keyword prison, due to that prison in the clue is in the end position while prison in the sentence is in the front position, and aligning them would introduce cross-alignment.



Again, in the following irrelevant sentences, Meteor still made unnecessary alignments:





B.2. Meteor Helped Answering

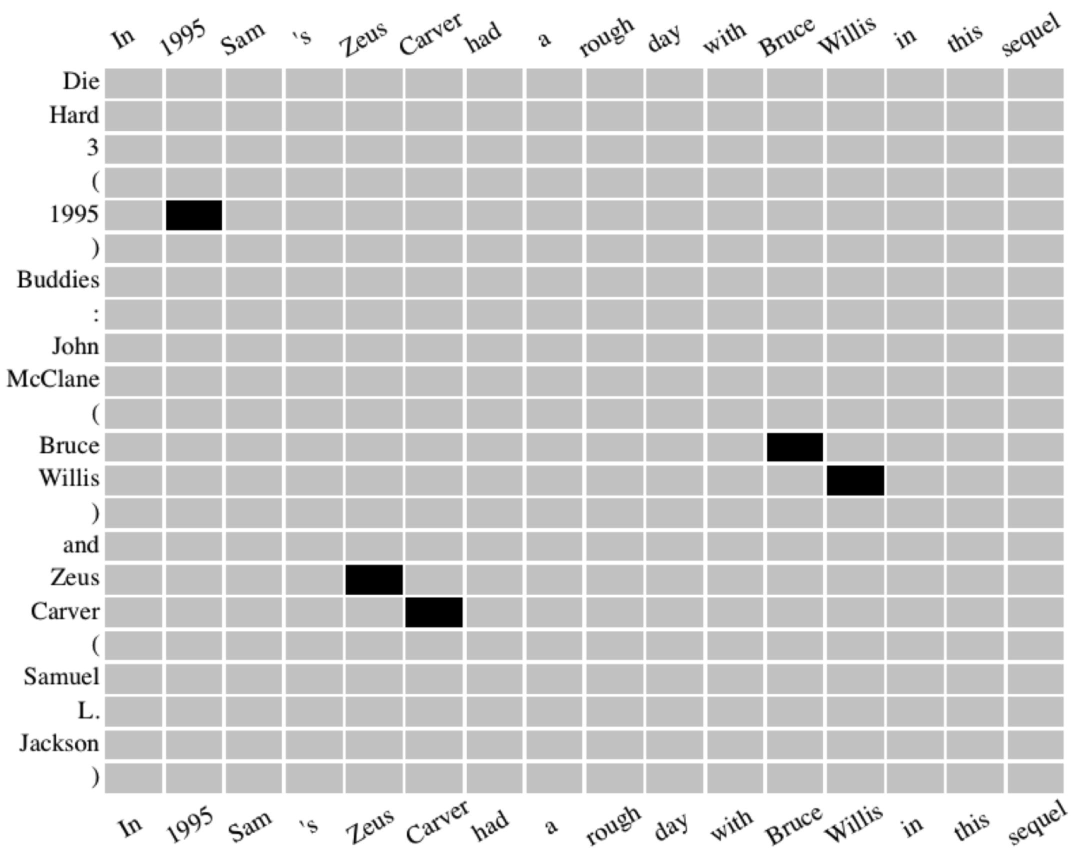
In the previous section I showed the high-precision nature of jacana-align, which helped answer extraction, a task that requires the same level of high precision. However, the recall might suffer for precision gain. Here I show one example:

- Clue: In 1995 Sam 's Zeus Carver had a rough day with Bruce Willis in **this sequel** .
- Answer: Die Hard 3

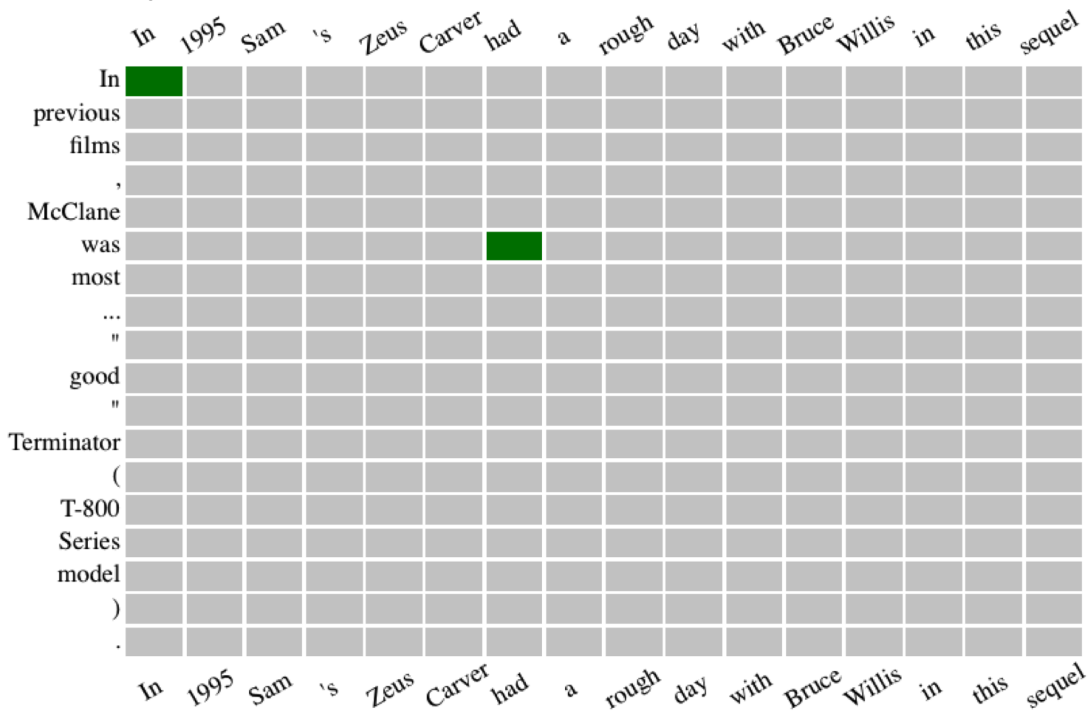
jacana-qa with Meteor answered this question correctly. But when paired with jacana-align, it proposed the following answers with low confidence scores:

terminator (t-800 series model)	0.44
simon	0.40
die hard 2	0.38
mora	0.37
lethal weapon 3	0.34
die hard 3	0.24
the die hard series	0.20

First of all, both jacana-align and Meteor reasonably aligned the answer-bearing sentence with the clue:

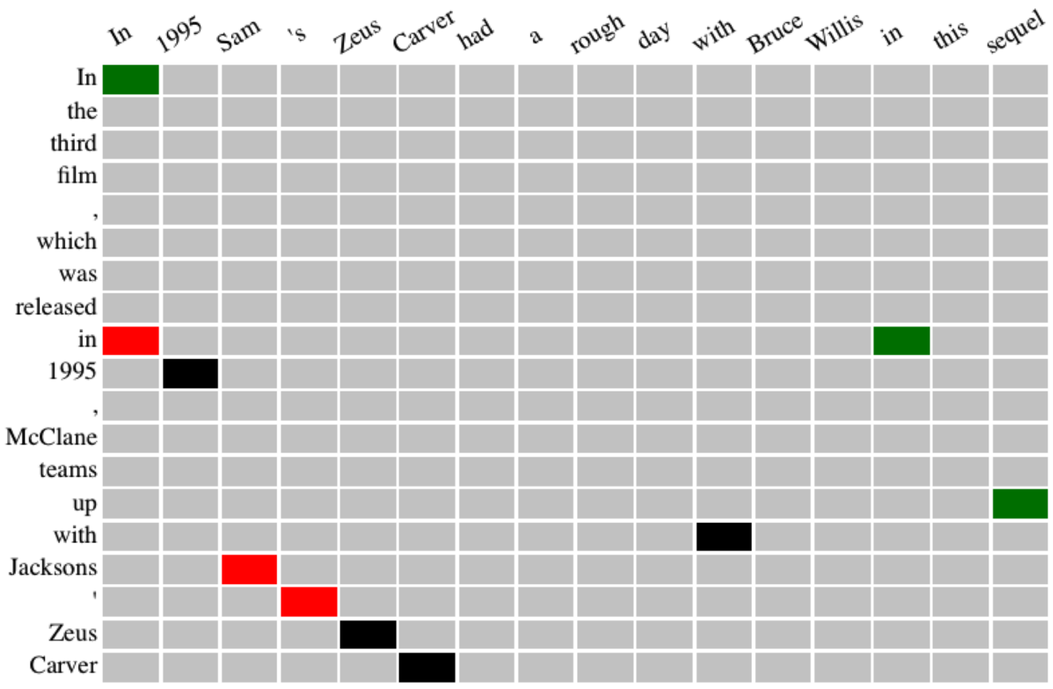


However, this seems to not have made jacana-qa give enough confidence to Die Hard 3. jacana-qa with jacana-align ranked terminator (t-800 series model) as the best from the following sentence:



Interestingly, jacana-align even did not align anything in the pair. My guess is that when not enough alignment-based features were triggered, other types of features instead took more important roles in deciding the best answer during the loglinear model decoding.

Finally, the following picture exemplifies a few distinctions between jacana-align and Meteor:



There are a few interesting alignments in the figure:

- There are two in's in both sentences, Meteor tried to align both of them without introducing cross-alignment. But both alignments were wrong. jacana-align successfully aligned the words pairs in 1995↔In 1995 due to its contextual feature and CRF-based global decoding nature.

- Beyond Zeus Carver↔Zeus Carver, jacana-align additionally aligned the possessive nouns: Jacksons ' Zeus Carver↔Sam 's Zeus Carver. This is a good alignment because Jacksons and Sam are the same person: Samuel Jackson played as McClane's reluctant partner Zeus Carver in Die Hard 3. This can be treated as a form of "shallow inference" jacana-align learned from its training data.

B.3. Summary

Analysis shows that jacana-align is a highly precise aligner for the task of answer extraction. Usually for a pair of English sentences, there are some light word overlap between the two, regardless of whether the pair is semantically relevant or not. A few examples shown previously have made it clear that in this situation jacana-align is able to decide whether to align these light words. Or in the case of multiple choices of light word alignment, jacana-align is able to figure out the best alignment based on context. Meteor, on the other hand, has failed to perform well in these examples. However, despite the high-precision nature of jacana-align, it sometimes can be over conservative in terms of making new alignment, a typical precision-recall tradeoff.

C. Examples of jacana-freebase vs. SEMPRE on WebQuestions

This section is a continuation of § 5.7 on page 224: "Comparison: Information Extraction vs. Semantic Parsing". I sample a few examples of the actual output form both jacana-freebase and SEMPRE. Note that on the WEBQUESTIONS dataset, there is not a mutually agreed DEV set but only a TEST set. Thus I only show a few examples without disclosing too much of the TEST set.

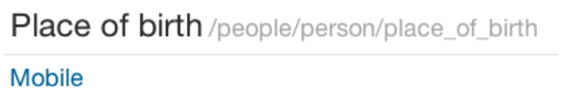
Also note that the WEBQUESTIONS dataset was crawled from Google Suggest. All questions are lowercased and are not necessarily well-formed English questions.

C.1. Both Correct

Question: where is jamarcus russell from?

Answer: Mobile

Entry from Freebase:



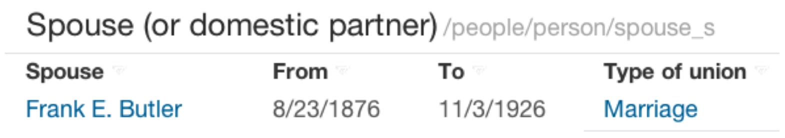
Place of birth /people/person/place_of_birth

Mobile

Question: who did annie oakley married?

Answer: Frank E. Butler

Entry from Freebase:



Spouse (or domestic partner) /people/person/spouse_s

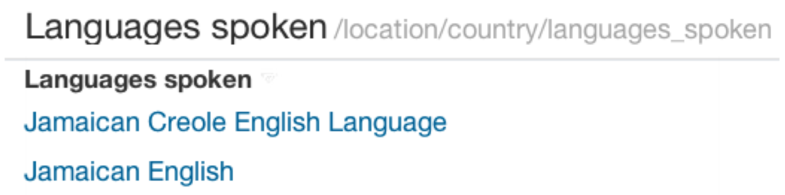
Spouse	From	To	Type of union
Frank E. Butler	8/23/1876	11/3/1926	Marriage

C.2. Only jacana-freebase Was Correct

Question: what does jamaican people speak?

Answer: [Jamaican Creole English Language, Jamaican English]

Entry from Freebase:



Languages spoken /location/country/languages_spoken

Languages spoken

Jamaican Creole English Language

Jamaican English

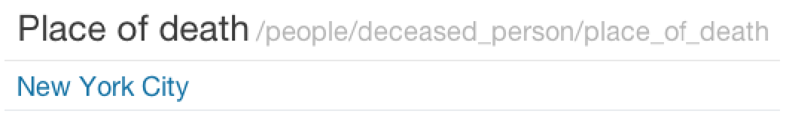
SEMPRE's answer:

[Chinese Jamaicans, Jamaicans of African ancestry, Jamaican American, Indo-Caribbean, British Jamaican, Jamaican Australian, Jamaican Canadian, Lebanese immigration to Jamaica, Igbo people in Jamaica, Chinese Caribbean]

Question: where did richard nixon die?

Answer: New York City

Entry from Freebase:


Place of death /people/deceased_person/place_of_death

New York City

SEMPRE's answer:

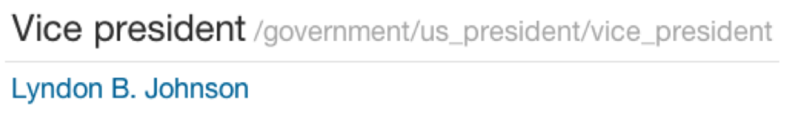
[Stroke, Cerebral edema]

C.3. Only SEMPRE Was Correct

Question: who was vice president after kennedy died?

Answer: Lyndon B. Johnson

Entry from Freebase:


Vice president /government/us_president/vice_president

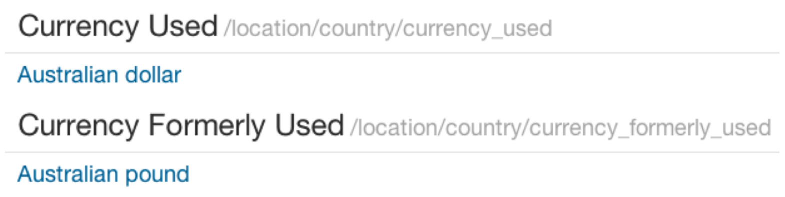
Lyndon B. Johnson

jacana-freebase proposed three answers (with unnormalized confidence): Dick Cheney (0.90), Aaron Burr (0.83), Al Gore (0.54). The reason is that it extracted answers from the Freebase vice_president page, instead of the page of John_F_Kennedy. This was an error partially due to the ranking of Freebase Search API.

Question: what is the australian dollar called?

Answer: Australian dollar

Entry from Freebase:



Currency Used /location/country/currency_used

Australian dollar

Currency Formerly Used /location/country/currency_formerly_used

Australian pound

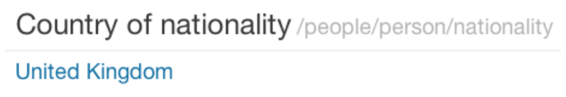
jacana-freebase classified both Australian dollar and Australian pound as the answer. SEMPRE did not make this mistake because that it only fired one (correct) query to Freebase and thus the only entry Australian dollar was retrieved.

C.4. Both Wrong

Question: where did andy murray started playing tennis?

Answer: United Kingdom

Entry from Freebase:

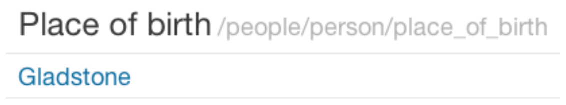

Country of nationality /people/person/nationality

United Kingdom

There are two “Andy Murray”’s on Freebase:

1. Andrew Barron "Andy" Murray, a Scottish tennis player, with Freebase page /en/andrew_murray;
2. Andy Murray, a US ice hockey coach, with Freebase page /en/andy_murray.

The Freebase Search API returned with the wrong /en/andy_murray, thus jacana-freebase extracted:


Place of birth /people/person/place_of_birth

Gladstone

This error is due to that the word tennis in the question was not part of the query. Thus the Freebase Search API could not disambiguate the two “Andy Murray”’s.

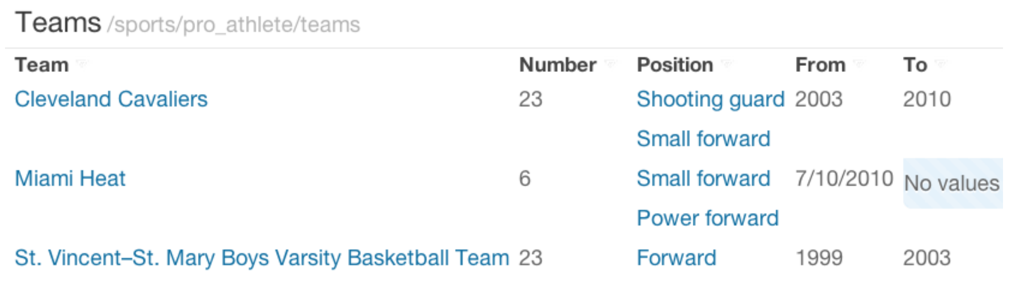
SEMPRE’s answer:

[London, Dunblane]

Question: what was lebron james first team?

Answer: Cleveland Cavaliers

Entry from Freebase:



Teams /sports/pro_athlete/teams

Team	Number	Position	From	To
Cleveland Cavaliers	23	Shooting guard Small forward	2003	2010
Miami Heat	6	Small forward Power forward	7/10/2010	No values
St. Vincent–St. Mary Boys Varsity Basketball Team	23	Forward	1999	2003

jacana-freebase used all teams as the answers.

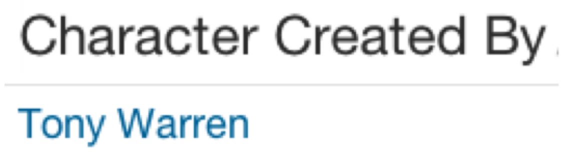
SEMPRE's answer:

[Miami Heat]

C.5. Errors Due to MTurk Annotation Error

Question: who plays ken barlow in coronation street?

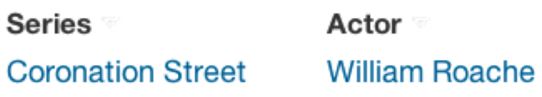
MTurk wrongly annotated the writer of Coronation Street:



Character Created By

Tony Warren

jacana-freebase proposed the correct answer William Roache:



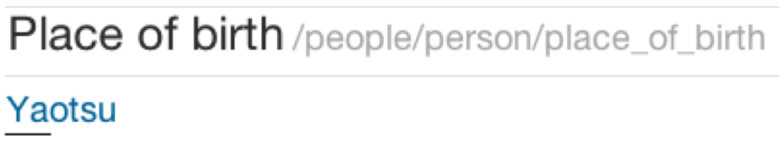
Series	Actor
Coronation Street	William Roache

SEMPRE's answer:

[Power forward, Small forward]

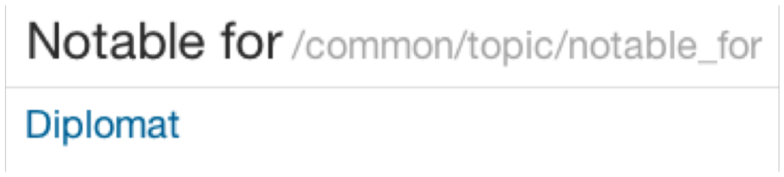
Question: what happened after mr. sugihara died?

MTurk wrongly annotated place of birth:


Place of birth /people/person/place_of_birth

Yaotsu

jacana-freebase proposed another wrong answer:


Notable for /common/topic/notable_for

Diplomat

The correct answer should be: a monument was dedicated to him in Japan (internet search). But this answer does not exist in Freebase.

SEMPRE's answer:

[Fujisawa]

C.6. Summary

Berant et al. (2013) described that all questions in the WebQuestions dataset were marked with identical answers by two Turkers. Even though, when performing error analysis on the dataset, it was surprising to see how frequent that Turkers gave wrong answers. Thus I did a random check on about 50 questions in the training set, and found that the annotation error rate was between 20% and 25%, depending on how strictly the answers had to be correct. A few of the questions with wrongly annotated answers include:

what state does selena gomez?	New York City
how old is sacha baron cohen?	a URL
what two countries invaded poland in the beginning of ww2?	Germany
which countries border the us?	Canada
where is rome italy located on a map?	Rome
how much did adriana lima gain during pregnancy?	Spike Guys' Choice Awards
what does thai mean?	Language
which wife did king henry behead?	Anne of the Thousand Days
what are the major cities in france?	Paris
what season did tony soprano get shot?	The Sopranos

Also, about 15% $\sim$ 20% errors came from more "complicated" questions: questions that have constraints based on time, location, comparison, etc. For instance:

1. what did james k polk do before he was president?
2. what is the oregon ducks 2012 football schedule?
3. what country did germany invade first in ww1?
4. who is governor of ohio 2011?
5. when did charles goodyear invented rubber?
6. who did france surrender to in ww2?
7. who did george w. bush run against for the second term?
8. who was the leader of soviet union during wwii?

Another 5% $\sim$ 10% of errors came from answer type matching failure:

1. what things did martin luther king do?
2. what town was martin luther king assassinated in?
3. what electorate does anna bligh represent?

4. what channel is the usa pageant on?

5. what are some of the traditions of islam?

6. what is the state flower of arizona?

7. what did the islamic people believe in?

8. what did the scientist chadwick discovered?

Other error types include Freebase search error (10%), ill-formed web text ($2\% \sim 3\%$), etc. Overall, in order for jacana-freebase to perform better, improvement is desired in the following directions:

1. better question analysis against the constrains in questions;

2. tighter answer typing for what X questions;

3. better text indexing and retrieval ranking.

Bibliography

Steven Abney, Michael Collins, and Amit Singhal. Answer extraction. In *Proceedings of the sixth conference on Applied natural language processing*, pages 296–301, 2000.

Arvind Agarwal, Hema Raghavan, Karthik Subbian, Prem Melville, Richard D. Lawrence, David C. Gondek, and James Fan. Learning to rank for robust question answering. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, pages 833–842, New York, NY, USA, 2012. ACM.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. Semeval-2012 task 6: A pilot on semantic textual similarity. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393, Montréal, Canada, 7-8 June 2012.

Ali Mohamed Nabil Allam and Mohamed Hassan Haggag. The question answering systems: A survey. *International Journal of Research and Reviews in Information Sciences (IJRRIS)*, 2(3), 2012.

A. Andrenucci and E. Sneiders. Automated question answering: Review of the main approaches. In *Third International Conference on Information Technology and Applications, 2005.*, volume 1, pages 514–519. IEEE, 2005.

Jesús Andrés-Ferrer and Alfons Juan. A phrase-based hidden semi-Markov approach to machine translation. In *Procedings of European Association for Machine Translation (EAMT)*, Barcelona, Spain, May 2009.

Nicholas Andrews, Jason Eisner, and Mark Dredze. Name phylogeny: a generative model of string variation. In *Proceedings of EMNLP 2012*, 2012.

Ioannis Androutsopoulos, Graeme D Ritchie, and Peter Thanisch. Natural Language Interfaces to Databases - An Introduction. *Natural Language Engineering*, 1995.

Ron Artstein and Massimo Poesio. Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics*, 34(4):555–596, 2008.

S.J. Athenikos and H. Han. Biomedical Question Answering: A Survey. *Computer methods and programs in biomedicine*, 99(1):1–24, 2010.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBPedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

Nikolaus Augsten, Denilson Barbosa, Michael Böhlen, and Themis Palpanas. TASM: Top-k Approximate Subtree Matching. In *Proceedings of the International Conference on Data Engineering (ICDE-10)*, pages 353–364, Long Beach, California, USA, March 2010. IEEE Computer Society.

Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

Colin Bannard and Chris Callison-Burch. Paraphrasing with bilingual parallel corpora. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 597–604, Morristown, NJ, USA, 2005.

Mohit Bansal, Chris Quirk, and Robert Moore. Gappy phrasal alignment by agreement. In *Proceedings of ACL*, Portland, Oregon, June 2011.

Jonathan Berant and Percy Liang. Semantic parsing via paraphrasing. In *Proceedings of ACL*, 2014.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of EMNLP*, 2013.

A. Berger and J. Lafferty. Information retrieval as statistical translation. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 222–229. ACM, 1999.

D.M. Bikel, R. Schwartz, and R.M. Weischedel. An algorithm that learns what's in a name. *Machine learning*, 34(1):211–231, 1999.

P. Bille. A survey on tree edit distance and related problems. *Theoretical computer science*, 337(1):217–239, 2005.

Matthew W. Bilotti, Jonathan L. Elsas, Jaime Carbonell, and Eric Nyberg. Rank Learning for Factoid Question Answering with Linguistic and Semantic Constraints. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM 2010)*, 2010a.

M.W. Bilotti. *Linguistic and semantic passage retrieval strategies for question answering.* PhD thesis, Carnegie Mellon University, 2009.

M.W. Bilotti and E. Nyberg. Improving text retrieval precision and answer accuracy in question answering systems. In *Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering*, pages 1–8, 2008.

M.W. Bilotti, P. Ogilvie, J. Callan, and E. Nyberg. Structured retrieval for question answering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 351–358. ACM, 2007.

M.W. Bilotti, J. Elsas, J. Carbonell, and E. Nyberg. Rank learning for factoid question answering with linguistic and semantic constraints. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 459–468. ACM, 2010b.

Steven Bird and Edward Loper. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, 2004.

P. Blunsom, K. Kocik, and J.R. Curran. Question classification with log-linear models. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 615–616. ACM, 2006.

Phil Blunsom and Trevor Cohn. Discriminative word alignment with conditional random fields. In *Proceedings of ACL2006*, pages 65–72, 2006.

Robert J Bobrow, Philip Resnik, and Ralph M Weischedel. Multiple underlying systems: Translating user requests into programs to produce answers. In *Proceedings of ACL*, pages 227–234. Association for Computational Linguistics, 1990.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM, 2008.

Gosse Bouma, Jori Mur, and Gertjan van Noord. Question Answering for Dutch using Dependency Relations. In *Proceedings CLEF 2005*, 2005.

Eric Brill, Jimmy J Lin, Michele Banko, Susan T Dumais, and Andrew Y Ng. Data-intensive question answering. In *TREC*, 2001.

Chris Brockett. Aligning the RTE 2006 corpus. Technical report, Microsoft Research, 2007.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993.

Bertram Bruce. A model for temporal references and its application in a question answering program. *Artificial intelligence*, 3:1–25, 1972.

Chris Buckley. Implementation of the SMART information retrieval system. Technical report, Cornell University, 1985.

Razvan Bunescu and Yunfeng Huang. Towards a general model of answer typing: Question focus identification. In *Proceedings of The 11th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2010), RCS Volume*, pages 231–242, 2010.

Robin D Burke, Kristian J Hammond, Vladimir Kulyukin, Steven L Lytinen, Noriko Tomuro, and Scott Schoenberg. Question answering from frequently asked question files: Experiences with the faq finder system. *AI magazine*, 18(2):57, 1997.

Davide Buscaldi and Paolo Rosso. Mining knowledge from wikipedia for the question answering task. In *Proceedings of the International Conference on Language Resources and Evaluation*, pages 727–730, 2006.

Qingqing Cai and Alexander Yates. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of ACL*, 2013a.

Qingqing Cai and Alexander Yates. Semantic parsing freebase: Towards open-domain semantic parsing. *Atlanta, Georgia, USA*, 30:328, 2013b.

James Callan, W. Bruce Croft, and Stephen M. Harding. The INQUERY Retrieval System. In *In Proceedings of the Third International Conference on Database and Expert Systems Applications*, pages 78–83. Springer-Verlag, 1992.

Nathanael Chambers, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon, Bill MacCartney, Marie-Catherine de Marneffe, Daniel Ramage, Eric Yeh, and Christopher D Manning. Learning alignments and leveraging natural logic. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 165–170, 2007.

David L Chen and Raymond J Mooney. Learning to Interpret Natural Language Navigation Instructions from Observations. In *AAAI*, volume 2, pages 1–2, 2011.

J. Chu-Carroll, J. Fan, B. K. Boguraev, D. Carmel, D. Sheinwald, and C. Welty. Finding needles in the haystack: Search and candidate generation. *IBM Journal of Research and Development*, 2012.

Jennifer Chu-Carroll, John Prager, Christopher Welty, Krzysztof Czuba, and David Ferrucci. A multi-strategy and multi-source approach to question answering. In *In Proceedings of TREC*, 2003.

Kenneth Ward Church. On memory limitations in natural language processing. Master's thesis, Massachusetts Institute of Technology, 1980.

Peter Clark, Vinay Chaudhri, Sunil Mishra, Jérôme Thoméré, Ken Barker, and Bruce Porter. Enabling domain experts to convey questions to a machine: a modified, template-based approach. In *Proceedings of the 2nd international conference on Knowledge capture*, K-CAP '03, pages 13–19, New York, NY, USA, 2003. ACM.

Charles LA Clarke, Gordon V Cormack, and Thomas R Lynam. Exploiting redundancy in question answering. In *Proceedings of SIGIR*, pages 358–365. ACM, 2001a.

Charles LA Clarke, Gordon V Cormack, Thomas R Lynam, CM Li, and GL McLearn. Web reinforced question answering (multitest experiments for trec 2001). In *TREC*, 2001b.

C.L.A. Clarke and E.L. Terra. Passage retrieval vs. document retrieval for factoid question answering. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 427–428. ACM, 2003.

C.L.A. Clarke, G.V. Cormack, and E.A. Tudhope. Relevance ranking for one to three term queries. *Information processing & management*, 36(2):291–311, 2000.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. Driving semantic parsing from the world's response. In *Proceedings of CoNLL*, 2010.

Trevor Cohn, Chris Callison-Burch, and Mirella Lapata. Constructing corpora for the development and evaluation of paraphrase systems. *Computational Linguistics*, 34(4): 597–614, December 2008.

W Bruce Croft, Donald Metzler, and Trevor Strohman. *Search engines: Information retrieval in practice*. Addison-Wesley Reading, 2010.

Silviu Cucerzan and Eugene Agichtein. Factoid question answering over unstructured and structured web content. In *TREC*, 2005.

Hang Cui, Keya Li, Renxu Sun, Tat-Seng Chua, and Min-Yen Kan. National University of Singapore at the TREC 13 Question Answering Main Task. In *TREC 2004 QA Track*, 2004.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 400–407, New York, NY, USA, 2005. ACM.

Deborah A Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proceedings of the workshop on Human Language Technology*, pages 43–48, 1994.

Marie-Catherine De Marneffe and Christopher D Manning. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8, 2008.

John DeNero and Dan Klein. Tailoring word alignments to syntactic machine translation. In *Proceedings of ACL2007*, 2007.

Y. Deng and W. Byrne. Hmm word and phrase alignment for statistical machine translation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(3):494–507, 2008.

Michael Denkowski and Alon Lavie. Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In *Proceedings of the EMNLP 2011 Workshop on Statistical Machine Translation*, 2011.

Shilin Ding, Gao Cong, Chin yew Lin, and Xiaoyan Zhu. Using conditional random fields to extract contexts and answers of questions from online forums. In *In Proceedings of ACL-08: HLT*, 2008.

Bill Dolan, Chris Quirk, and Chris Brockett. Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In *Proceedings of COLING*, Stroudsburg, PA, USA, 2004.

Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. Web question answering: Is more always better? In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 291–298. ACM, 2002.

Abdessamad Echihabi and Daniel Marcu. A noisy-channel approach to question answering. In *Proceedings of ACL*, pages 16–23, 2003.

Katrin Erk and Sebastian Pado. Shalmaneser–a toolchain for shallow semantic parsing. In *Proceedings of LREC*, volume 6. Citeseer, 2006.

Jérôme Euzenat and Pavel Shvaiko. *Ontology matching.* Springer, 2007.

Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of EMNLP*, 2011.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Paraphrase-Driven Learning for Open Question Answering. In *Proceedings of ACL*, 2013.

J. Fan, A. Kalyanpur, D. C. Gondek, and D. A. Ferrucci. Automatic knowledge extraction from documents. *IBM Journal of Research and Development*, 2012.

Christiane Fellbaum. *WordNet: An Electronic Lexical Database.* 1998.

D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A.A. Kalyanpur, A. Lally, J.W. Murdock, E. Nyberg, J. Prager, et al. Building Watson: An overview of the DeepQA project. *AI Magazine*, 31(3):59–79, 2010.

D. A. Ferrucci. Introduction to this is watson. *IBM Journal of Research and Development*, 2012.

Charles J Fillmore, Christopher R Johnson, and Miriam RL Petruck. Background to FrameNet. *International journal of lexicography*, 16(3):235–250, 2003.

Michael Fleischman, Eduard Hovy, and Abdessamad Echihabi. Offline strategies for online question answering: Answering questions before they are asked. In *Proceedings of ACL*, pages 1–7, 2003.

Anette Frank, Hans-Ulrich Krieger, Feiyu Xu, Hans Uszkoreit, Berthold Crysmann, Brigitte Jörg, and Ulrich Schäfer. Question answering from structured knowledge sources. *Journal of Applied Logic*, 5(1):20–48, 2007.

Evgeniy Gabrilovich, Michael Ringgaard, , and Amarnag Subramanya. FACC1: Freebase annotation of ClueWeb corpora, Version 1 (Release date 2013-06-26, Format version 1, Correction level 0). http://lemurproject.org/clueweb09/FACC1/, June 2013.

Robert Gaizauskas, Mark Hepple, and Mark Greenwood. Information retrieval for question answering a SIGIR 2004 workshop. In *ACM SIGIR Forum*, volume 38, pages 41–44. ACM, 2004.

M. Galley. A skip-chain conditional random field for ranking meeting utterances by importance. In *Proceedings of EMNLP 2006*, pages 364–372. Association for Computational Linguistics, 2006.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB: The Paraphrase Database. In *Proceedings of NAACL-HLT*, 2013.

Ruifang Ge and Raymond J Mooney. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 9–16, 2005.

Kevin Gimpel and Noah A. Smith. Softmax-margin CRFs: training log-linear models with cost functions. In *NAACL 2010*, pages 733–736, 2010.

Roxana Girju. Automatic detection of causal relations for question answering. In *Proceedings of the ACL 2003 workshop on Multilingual summarization and question answering*, pages 76–83, 2003.

D. C. Gondek, A. Lally, A. Kalyanpur, J. W. Murdock, P. A. Duboue, L. Zhang, Y. Pan, Z. M. Qiu, and C. Welty. A framework for merging and ranking of answers in deepqa. *IBM Journal of Research and Development*, 2012.

Bert F Green, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 219–224. ACM, 1961.

Cordell Green. Theorem-proving by resolution as a basis for question-answering systems, in. *Machine Intelligence, B. Meltzer and D. Michie, eds*, pages 183–205, 1969.

Mark A. Greenwood, editor. *Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering.* Coling 2008 Organizing Committee, Manchester, UK, August 2008.

Poonam Gupta and Vishal Gupta. A Survey of Text Question Answering Techniques. *International Journal of Computer Applications*, 53(4):1–8, September 2012. Published by Foundation of Computer Science, New York, USA.

K. Hacioglu and W. Ward. Question classification with support vector machines and error correcting codes. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2*, pages 28–30. Association for Computational Linguistics, 2003.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11 (1):10–18, 2009.

K. Hammond, R. Burke, C. Martin, and S. Lytinen. FAQ finder: a case-based approach to knowledge navigation. In *Conference on Artificial Intelligence Applications*, 1995.

Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. UMBC-EBIQUITY-CORE: Semantic Textual Similarity Systems. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, 2013.

S. Harabagiu, D. Moldovan, M. Pasca, M. Surdeanu, R. Mihalcea, R. Girju, V. Rus, F. Lactusu, P. Morarescu, and R. Bunescu. Answering complex, list and context questions with lcc's question-answering server. In *Proceedings of TREC 2001*, 2001.

Sanda Harabagiu and Andrew Hickl. Methods for using textual entailment in open-domain question answering. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 905–912, 2006.

Sanda M Harabagiu, Dan I Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan C Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. FALCON: Boosting Knowledge for Answer Engines. In *TREC*, volume 9, pages 479–488, 2000.

Donna Harman and Gerald Candela. Retrieving records from a gigabyte of text on a mini-computer using statistical ranking. *JASIS*, 41(8):581–589, 1990.

Donna K Harman and Ellen M Voorhees. Trec: An overview. *Annual review of information science and technology*, 40(1):113–155, 2006.

Sven Hartrumpf. Semantic Decomposition for Question Answering. In *ECAI*, pages 313–317, 2008.

Michael Heilman and Noah A. Smith. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of NAACL 2010*, pages 1011–1019, Los Angeles, California, June 2010.

Gary G Hendrix, Earl D Sacerdoti, Daniel Sagalowicz, and Jonathan Slocum. Developing a natural language interface to complex data. *ACM Transactions on Database Systems (TODS)*, 3(2):105–147, 1978.

U. Hermjakob. Parsing and question classification for question answering. In *Proceedings of the workshop on Open-domain question answering-Volume 12*, pages 1–6, 2001.

Ulf Hermjakob, Abdessamad Echihabi, and Daniel Marcu. Natural language based reformulation resource and wide exploitation for question answering. In *TREC*, 2002.

Lynette Hirschman and Robert Gaizauskas. Natural language question answering: The view from here. *Natural Language Engineering*, 7(4):275–300, 2001.

Jerry R Hobbs. Ontological promiscuity. In *Proceedings of ACL*, 1985.

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard De Melo, and Gerhard Weikum. Yago2: exploring and querying world knowledge in time, space, context, and many languages. In *Proceedings of the 20th international conference companion on World Wide Web*, pages 229–232. ACM, 2011.

Eduard Hovy, Ulf Hermjakob, and Deepak Ravichandran. A question/answer typology with surface text patterns. In *Proceedings of the second international conference on Human Language Technology Research*, pages 247–251. Morgan Kaufmann Publishers Inc., 2002.

Eduard H Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin. Question answering in webclopedia. In *TREC*, 2000.

Zhiheng Huang, Marcus Thint, and Zengchang Qin. Question classification using head words and their hypernyms. In *Proceedings of EMNLP*, pages 927–936, 2008.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

A. Ittycheriah, M. Franz, and S. Roukos. Ibm's statistical question answering system—trec-10. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, 2001a.

Abraham Ittycheriah, Martin Franz, Wei-Jing Zhu, Adwait Ratnaparkhi, and Richard J Mammone. Question answering using maximum entropy components. In *Proceedings of NAACL*, 2001b.

Peter Jackson. *Introduction To Expert Systems.* Addison-Wesley Longman Publishing Co., Inc., 1990.

Valentin Jijkoun, Maarten De Rijke, and Jori Mur. Information extraction for question answering: Improving recall through syntactic patterns. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1284, 2004.

Bevan Keeley Jones, Mark Johnson, and Sharon Goldwater. Semantic parsing with bayesian tree transducers. In *Proceedings of ACL*, 2012.

John Judge, Aoife Cahill, and Josef Van Genabith. QuestionBank: Creating a Corpus of Parse-Annotated Questions. In *In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 497–504, 2006.

Michael Kaisser. Answer Sentence Retrieval by Matching Dependency Paths acquired from Question/Answer Sentence Pairs. In *EACL*, pages 88–98, 2012.

A. Kalyanpur, B. K. Boguraev, S. Patwardhan, J. W. Murdock, A. Lally, C. Welty, J. M. Prager, B. Coppola, A. Fokoue-Nkoutche, L. Zhang, Y. Pan, and Z. M. Qiu. Structured data and inference in deepqa. *IBM Journal of Research and Development*, 2012a.

A. Kalyanpur, S. Patwardhan, B. K. Boguraev, A. Lally, and J. Chu-Carroll. Fact-based question decomposition in deepqa. *IBM Journal of Research and Development*, 2012b.

Rohit J Kate and Raymond J Mooney. Using string-kernels for learning semantic parsers. In *Proceedings of ACL*, 2006.

Boris Katz and Jimmy Lin. Selectively using relations to improve precision in question answering. In *Proceedings of the workshop on Natural Language Processing for Question Answering (EACL 2003)*, pages 43–50, 2003.

Boris Katz, Sue Felshin, Deniz Yuret, Ali Ibrahim, Jimmy Lin, Gregory Marton, Alton Jerome McFarland, and Baris Temelkuran. Omnibase: Uniform access to heterogeneous data for question answering. In *Natural Language Processing and Information Systems*, pages 230–234. Springer, 2002.

Boris Katz, Gary C Borchardt, and Sue Felshin. Natural language annotations for question answering. In *FLAIRS Conference*, pages 303–306, 2006.

Tibor Kiss and Jan Strunk. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525, 2006.

Dan Klein and Christopher D. Manning. Accurate Unlexicalized Parsing. In *In Proc. the 41st Annual Meeting of the Association for Computational Linguistics*, 2003.

Philipp Koehn. *Statistical Machine Translation.* Cambridge University Press, New York, NY, USA, 2010.

Oleksandr Kolomiyets and Marie-Francine Moens. A survey on question answering technology from an information retrieval perspective. *Information Sciences*, 181(24):5412 – 5434, 2011.

Lingpeng Kong and Noah A Smith. An empirical comparison of parsing methods for stanford dependencies. *arXiv preprint arXiv:1404.4314*, 2014.

Leila Kosseim and Jamileh Yousefi. Improving the performance of question answering with semantically equivalent answer patterns. *Data & Knowledge Engineering*, 66(1): 53–67, 2008.

Milen Kouylekov and Bernardo Magnini. Recognizing textual entailment with tree edit distance algorithms. In *PASCAL Challenges on RTE*, pages 17–20, 2005.

Klaus H. Krippendorff. *Content Analysis: An Introduction to Its Methodology.* Sage Publications, Inc, 2nd edition, 2004.

Jayant Krishnamurthy and Tom M Mitchell. Weakly supervised training of semantic parsers. In *Proceedings of EMNLP-CoNLL*, 2012.

Julian Kupiec. Murax: A robust linguistic approach for question answering using an online encyclopedia. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 181–190. ACM, 1993.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of EMNLP*, pages 1223–1233, 2010.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of EMNLP*, 2011.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. Scaling Semantic Parsers with On-the-fly Ontology Matching. In *Proceedings of EMNLP*, 2013.

Cody Kwok, Oren Etzioni, and Daniel S Weld. Scaling question answering to the web. *ACM Transactions on Information Systems (TOIS)*, 19(3):242–262, 2001.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

A. Lally, J. M. Prager, M. C. McCord, B. K. Boguraev, S. Patwardhan, J. Fan, P. Fodor, and J. Chu-Carroll. Question analysis: How watson reads a clue. *IBM Journal of Research and Development*, 2012.

Wendy G Lehnert. A conceptual theory of question answering. In *Proceedings of the 5th international joint conference on Artificial intelligence-Volume 1*, pages 158–164. Morgan Kaufmann Publishers Inc., 1977.

Jochen L Leidner, Gail Sinclair, and Bonnie Webber. Grounding spatial named entities for information extraction and question answering. In *Proceedings of the HLT-NAACL 2003 workshop on Analysis of geographic references-Volume 1*, pages 31–38, 2003.

Fangtao Li, Xian Zhang, Jinhui Yuan, and Xiaoyan Zhu. Classifying what-type questions by head noun tagging. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 481–488, 2008.

X. Li and D. Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics*, 2002.

Xiaoyan Li. Syntactic features in question answering. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 383–384. ACM, 2003.

Xin Li and Dan Roth. Learning question classifiers: the role of semantic information. *Natural Language Engineering*, 12(3):229–249, 2006.

Percy Liang. Lambda dependency-based compositional semantics. *arXiv preprint arXiv:1309.4408*, 2013.

Percy Liang and Christopher Potts. Bringing machine learning and compositional semantics together. *the Annual Review of Linguistics*, 2014.

Percy Liang, Ben Taskar, and Dan Klein. Alignment by agreement. In *Proceedings of NAACL*, pages 104–111, 2006.

Percy Liang, Michael I. Jordan, and Dan Klein. Learning Dependency-Based Compositional Semantics. In *Proceedings of ACL*, 2011.

M. Light, G.S. Mann, E. Riloff, and E. Breck. Analyses for elucidating current question answering technology. *Natural Language Engineering*, 7(04):325–342, 2001.

Dekang Lin. Principle-based parsing without overgeneration. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 112–120. Association for Computational Linguistics, 1993.

Dekang Lin and Patrick Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, (4), 2001.

J. Lin and B. Katz. Building a reusable test collection for question answering. *Journal of the American Society for Information Science and Technology*, 57(7):851–861, 2006.

Jimmy Lin. The web as a resource for question answering: Perspectives and challenges. In *Proceedings of TREC*, 2002.

Jimmy Lin. An exploration of the principles underlying redundancy-based factoid question answering. *ACM Transactions on Information Systems*, 25(2), April 2007. ISSN 1046-8188.

Thomas Lin, Oren Etzioni, et al. Entity Linking at Web Scale. In *Proceedings of Knowledge Extraction Workshop (AKBC-WEKEX)*, pages 84–88, 2012.

Babak Loni. A survey of state-of-the-art methods on question classification. *Literature Survey, Published on TU Delft Repository*, 2011.

Vanessa Lopez, Michele Pasin, and Enrico Motta. Aqualog: An ontology-portable question answering system for the semantic web. In *The Semantic Web: Research and Applications*, pages 546–562. Springer, 2005.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S Zettlemoyer. A generative model for parsing natural language to meaning representations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 783–792, 2008.

Bill MacCartney. *Natural language inference.* PhD thesis, Stanford University, 2009.

Bill MacCartney, Michel Galley, and Christopher D Manning. A phrase-based alignment model for natural language inference. In *Proceedings of EMNLP*, pages 802–811, 2008.

B. Magnini, M. Negri, R. Prevete, and H. Tanev. Is it the right answer?: exploiting web redundancy for answer validation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 425–432, 2002.

Bernardo Magnini, Simone Romagnoli, Alessandro Vallin, Jesus Herrera, Anselmo Penas, Victor Peinado, Felisa Verdejo, Maarten de Rijke, and Ro Vallin. The multiple language question answering track at clef 2003. In *CLEF 2003*. Springer-Verlag, 2003.

Bernardo Magnini, Alessandro Vallin, Christelle Ayache, Gregor Erbach, Anselmo Peñas, Maarten De Rijke, Paulo Rocha, Kiril Simov, and Richard Sutcliffe. Overview of the CLEF 2004 multilingual question answering track. In *Multilingual Information Access for Text, Speech and Images*, pages 371–391. Springer, 2005.

Gideon S Mann. A statistical method for short answer extraction. In *Proceedings of the workshop on Open-domain question answering*, pages 1–8, 2001.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

Daniel Marcu and William Wong. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP-2002*, pages 133–139, 2002.

Andrew McCallum, Kedar Bellare, and Fernando Pereira. A Conditional Random Field for Discriminatively-trained Finite-state String Edit Distance. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI 2005)*, July 2005.

M. C. McCord, J. W. Murdock, and B. K. Boguraev. Deep parsing in watson. *IBM Journal of Research and Development*, 2012.

Michael C McCord. Slot grammars. *Computational Linguistics*, 6(1):31–43, 1980.

R. McDonald, K. Crammer, and F. Pereira. Online large-margin training of dependency parsers. In *Proceedings of ACL 2005*, pages 91–98, 2005.

Yashar Mehdad. Automatic cost estimation for tree edit distance using particle swarm optimization. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 289–292, 2009.

D. Metzler and W.B. Croft. Analysis of statistical question classification for fact-based questions. *Information Retrieval*, 8(3):481–504, 2005.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL-HLT*, pages 746–751, 2013.

D. Moldovan, M. Paşca, S. Harabagiu, and M. Surdeanu. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems (TOIS)*, 21(2):133–154, 2003a.

Dan Moldovan, Christine Clark, Sanda Harabagiu, and Steve Maiorano. Cogex: A logic prover for question answering. In *Proceedings of NAACL*, pages 87–93, 2003b.

Dan I Moldovan and Vasile Rus. Logic form transformation of wordnet and its applicability to question answering. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 402–409, 2001.

Dan I Moldovan, Sanda M Harabagiu, Roxana Girju, Paul Morarescu, V Finley Lacatusu, Adrian Novischi, Adriana Badulescu, and Orest Bolohan. Lcc tools for question answering. In *TREC*, 2002.

Diego Mollá and Menno Van Zaanen. Learning of graph rules for question answering. In *Proceedings of the Australasian Language Technology Workshop*, 2005.

Diego Mollá and José Luis Vicedo. Question answering in restricted domains: An overview. *Computational Linguistics*, 33(1):41–61, 2007.

Diego Mollá, Rolf Schwitter, Fabio Rinaldi, James Dowdall, and Michael Hess. Nlp for answer extraction in technical domains. *Proceedings of EACL*, 2003.

A. Moschitti, S. Quarteroni, R. Basili, and S. Manandhar. Exploiting syntactic and shallow semantic kernels for question answer classification. In *ACL*, volume 45, page 776, 2007.

J. W. Murdock, J. Fan, A. Lally, H. Shima, and B. K. Boguraev. Textual evidence gathering and analysis. *IBM Journal of Research and Development*, 2012a.

J. W. Murdock, A. Kalyanpur, C. Welty, J. Fan, D. A. Ferrucci, D. C. Gondek, L. Zhang, and H. Kanayama. Typing candidate answers using type coercion. *IBM Journal of Research and Development*, 2012b.

V. Murdock and W.B. Croft. Simple translation models for sentence retrieval in factoid question answering. In *Proceedings of the SIGIR-2004 Workshop on Information Retrieval For Question Answering (IR4QA)*, pages 31–35, 2004.

S. Narayanan and S. Harabagiu. Question answering based on semantic structures. In *Proceedings of the 20th international conference on Computational Linguistics*, page 693. Association for Computational Linguistics, 2004.

E. Nyberg, T. Mitamura, J. Callan, J. Carbonell, R. Frederking, K. Collins-thompson, L. Hiyakumoto, Y. Huang, C. Huttenhower, S. Judy, J. Ko, L. V. Lita, V. Pedro, D. Svoboda, and B. Van Durme. The javelin question-answering system at trec 2002. In *Proceedings of TREC 12*, 2003.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51, 2003.

P. Ogilvie. *Retrieval using Document Structure and Annotations*. PhD thesis, Carnegie Mellon University, 2010.

Naoaki Okazaki. CRFsuite: a fast implementation of Conditional Random Fields (CRFs), 2007.

Naoaki Okazaki. Classias: a collection of machine-learning algorithms for classification, 2009. URL `http://www.chokkan.org/software/classias/`.

Dave Orr, Amar Subramanya, Evgeniy Gabrilovich, and Michael Ringgaard. 11 billion clues in 800 million documents: A web research corpus annotated with freebase concepts. http://googleresearch.blogspot.com/2013/07/11-billion-clues-in-800-million.html, July 2013.

Mari Ostendorf, Vassilios V Digalakis, and Owen A Kimball. From HMM's to segment models: a unified view of stochastic modeling for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 4(5):360–378, 1996.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, 2005.

Yan Pan, Yong Tang, Luxin Lin, and Yemin Luo. Question classification with semantic tree kernel. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 837–838. ACM, 2008.

Marius A Pasca and Sandra M Harabagiu. High performance question/answering. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 366–374. ACM, 2001.

A. Penas, A. Rodrigo, V. Sama, and F. Verdejo. Testing the reasoning for question answering validation. *Journal of Logic and Computation*, 18(3):459–474, 2008.

Anselmo Peñas, Pamela Forner, Richard Sutcliffe, Álvaro Rodrigo, Corina Forăscu, Iñaki Alegria, Danilo Giampiccolo, Nicolas Moreau, and Petya Osenova. Overview of ResPubliQA 2009: question answering evaluation over European legislation. In *Multilingual Information Access Evaluation I. Text Retrieval Experiments*, pages 174–196. Springer, 2010.

Anselmo Peñas, Eduard H Hovy, Pamela Forner, Álvaro Rodrigo, Richard FE Sutcliffe, Corina Forascu, and Caroline Sporleder. Overview of QA4MRE at CLEF 2011: Question Answering for Machine Reading Evaluation. In *CLEF 2011 Labs and Workshop Notebook Papers*. Citeseer, 2011.

Anselmo Peñas, Eduard Hovy, Pamela Forner, Álvaro Rodrigo, Richard Sutcliffe, and Roser Morante. Qa4mre 2011-2013: Overview of question answering for machine reading evaluation. In *Information Access Evaluation. Multilinguality, Multimodality, and Visualization*, pages 303–320. Springer, 2013.

Fuchun Peng, Ralph Weischedel, Ana Licuanan, and Jinxi Xu. Combining deep linguistics analysis and surface pattern learning: A hybrid approach to chinese definitional question answering. In *Proceedings of EMNLP*, pages 307–314, 2005.

Christopher Pinchak and Dekang Lin. A probabilistic answer type model. In *Proceedings of EACL*, 2006.

Christopher Pinchak, Dekang Lin, and Davood Rafiei. Flexible answer typing with discriminative preference ranking. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 666–674, Athens, Greece, March 2009a.

Christopher Pinchak, Davood Rafiei, and Dekang Lin. Answer typing for information retrieval. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 1955–1958, New York, NY, USA, 2009b. ACM.

Warren J. Plath. Request: a natural language question-answering system. *IBM Journal of Research and Development*, 20(4):326–335, 1976.

Martha Elizabeth Pollack. Inferring domain plans in question-answering. Technical report, SRI International, Philadelphia, PA, USA, 1986. UMI order no. GAX86-14850.

Jay M Ponte and W Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281. ACM, 1998.

Martin F Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980.

Sameer S Pradhan, Wayne Ward, Kadri Hacioglu, James H Martin, and Daniel Jurafsky. Shallow Semantic Parsing using Support Vector Machines. In *HLT-NAACL*, pages 233–240, 2004.

J. Prager, J. Chu-Carroll, E. Brown, and K. Czuba. Question answering by predictive annotation. *Advances in Open Domain Question Answering*, pages 307–347, 2006.

John Prager, Eric Brown, Anni Coden, and Dragomir Radev. Question-answering by predictive annotation. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, pages 184–191, New York, NY, USA, 2000. ACM.

John Prager, Jennifer Chu-Carroll, and Krzysztof Czuba. Use of wordnet hypernyms for answering what-is questions. In *Proceedings of TREC 2001*, 2001.

John M Prager. Open-domain question-answering. *Foundations and Trends in Information Retrieval*, 1(2):91–231, 2006.

Vasin Punyakanok, Dan Roth, and Wen T. Yih. Mapping Dependencies Trees: An Application to Question Answering. In *Proceedings of the 8th International Symposium on Artificial Intelligence and Mathematics*, Fort Lauderdale, Florida, 2004.

Silvia Quarteroni and Suresh Manandhar. Designing an interactive open-domain question answering system. *Natural Language Engineering*, 15(1):73–95, 2009.

Chris Quirk, Chris Brockett, and William B Dolan. Monolingual Machine Translation for Paraphrase Generation. In *EMNLP*, pages 142–149, 2004.

L Rabiner, AE Rosenberg, and SE Levinson. Considerations in dynamic time warping algorithms for discrete word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 26(6):575–582, 1978.

Dragomir Radev, Weiguo Fan, Hong Qi, Harris Wu, and Amardeep Grewal. Probabilistic question answering on the web. *Journal of the American Society for Information Science and Technology*, 56(6):571–583, 2005.

Erhard Rahm and Philip A Bernstein. A survey of approaches to automatic schema matching. *the VLDB Journal*, 10(4):334–350, 2001.

Ganesh Ramakrishnan, Apurva Jadhav, Ashutosh Joshi, Soumen Chakrabarti, and Pushpak Bhattacharyya. Question answering via bayesian inference on lexical relations. In *Proceedings of the ACL 2003 workshop on Multilingual summarization and question answering-Volume 12*, pages 1–10, 2003.

L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *CoNLL*, 6 2009.

A. Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP*, volume 1, pages 133–142, 1996.

Deepak Ravichandran and Eduard Hovy. Learning surface text patterns for a question answering system. In *Proceedings of ACL*, ACL '02, pages 41–47, Stroudsburg, PA, USA, 2002.

Deepak Ravichandran, Abraham Ittycheriah, and Salim Roukos. Automatic derivation of surface text patterns for a maximum entropy based question answering system. In *Proceedings of NAACL, short papers*, pages 85–87, 2003.

D. C. Reis, P. B. Golgher, A. S. Silva, and A. F. Laender. Automatic web news extraction using tree edit distance. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 502–511, New York, NY, USA, 2004. ACM.

S.E. Robertson and S. Walker. Okapi/keenbow at trec-8. In *Proc. of TREC*, volume 8, 1999.

Á. Rodrigo, A. Peñas, and F. Verdejo. Overview of the answer validation exercise 2008. *Evaluating Systems for Multilingual and Multimodal Information Access*, pages 296–313, 2009.

Michael Roth and Anette Frank. Aligning predicates across monolingual comparable texts using graph-based clustering. In *Proceedings of EMNLP-CoNLL*, pages 171–182, Jeju Island, Korea, July 2012.

Rachel Rudinger and Benjamin Van Durme. Is the Stanford Dependency Representation Semantic? In *Association for Computational Linguistics (ACL), Workshop on EVENTS*, 2014.

Tetsuya Sakai, Hideki Shima, Noriko Kando, Ruihua Song, Chuan-Jie Lin, Teruko Mitamura, Miho Sugimito, and Cheng-Wei Lee. Overview of the ntcir-7 aclia ir4qa task. In *Proceedings of NTCIR-8 Workshop Meeting*, Tokyo, Japan, 2010.

Gerard Salton. *The SMART retrieval system - experiments in automatic document processing.* Prentice-Hall, 1971.

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

Sarawagi Sarawagi and William Cohen. Semi-markov conditional random fields for information extraction. *Advances in Neural Information Processing Systems*, 17:1185–1192, 2004.

Nico Schlaefer, Petra Gieselman, and Guido Sautter. The ephyra qa system at trec 2006. In *In Proceedings TREC*, 2006.

Stefan Schlobach, David Ahn, Maarten De Rijke, and Valentin Jijkoun. Data-driven type checking in open domain question answering. *Journal of Applied Logic*, 5(1):121–143, 2007.

Aliaksei Severyn and Alessandro Moschitti. Automatic feature engineering for answer selection and extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 458–467, Seattle, Washington, USA, October 2013.

Saeedeh Shekarpour, Axel-Cyrille Ngonga Ngomo, and Sören Auer. Question answering on interlinked data. In *Proceedings of WWW*, 2013.

D. Shen and M. Lapata. Using semantic roles to improve question answering. In *Proceedings of EMNLP-CoNLL*, pages 12–21, 2007.

Dan Shen and Dietrich Klakow. Exploring correlation of dependency relation paths for answer extraction. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 889–896, 2006.

H. Shima, N. Lao, E. Nyberg, and T. Mitamura. Complex cross-lingual question answering as sequential classification and multi-document summarization task. In *Proceedings of NTICIR-7 Workshop, Japan*, 2008.

Joao Silva, Luísa Coheur, Ana Cristina Mendes, and Andreas Wichert. From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review*, 35(2):137–154, 2011.

R. F. Simmons. Answering english questions by computer: A survey. *Communications of the ACM*, 8(1):53–70, January 1965.

Robert F Simmons. Natural language question-answering systems: 1969. *Communications of the ACM*, 13(1):15–30, 1970.

Amit Singhal. Modern Information Retrieval: A Brief Overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24(4):35–43, 2001.

James R Slagle. Experiments with a deductive question-answering program. *Communications of the ACM*, 8(12):792–798, 1965.

David A. Smith and Jason Eisner. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the HLT-NAACL Workshop on Statistical Machine Translation*, pages 23–30, New York, June 2006.

Temple F Smith and Michael S Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

Mark D. Smucker, James Allan, and Ben Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 623–632, New York, NY, USA, 2007. ACM.

E. Sneiders. *Automated question answering: template-based approach.* PhD thesis, KTH, 2002a.

Eriks Sneiders. Automated question answering using question templates that cover the conceptual model of the database. In *Natural Language Processing and Information Systems*, pages 235–239. Springer, 2002b.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 801–808, 2006.

Radu Soricut and Eric Brill. Automatic question answering: Beyond the factoid. In *HLT-NAACL*, pages 57–64, 2004.

Martin M. Soubbotin. Patterns of potential answer expressions as clues to the right answers. In *Proceedings of TREC 2001*, 2001.

Martin M Soubbotin and Sergei M Soubbotin. Use of patterns for detection of likely answer strings: A systematic approach. In *TREC*, 2002.

Rohini Srihari and Wei Li. A question answering system supported by information extraction. In *Proceedings of the sixth conference on Applied natural language processing*, pages 166–172, 2000.

Mark Steedman. *The syntactic process.* MIT Press, Cambridge, MA, USA, 2000.

T. Strohman, D. Metzler, H. Turtle, and W.B. Croft. Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligent Analysis*, volume 2, pages 2–6. Citeseer, 2005.

Mingyu Sun and Joyce Y Chai. Discourse processing for context question answering based on linguistic knowledge. *Knowledge-Based Systems*, 20(6):511–526, 2007.

Renxu Sun, Jing Jiang, Yee Fan, Tan Hang, Cui Tat-seng, and Chua Min yen Kan. Using syntactic and semantic relation analysis in question answering. In *Proceedings of TREC*, 2005.

Charles Sutton and Andrew Mccallum. Collective segmentation and labeling of distant entities in information extraction. Technical Report TR # 04-49, University of Massachusetts, July 2004.

Lappoon R Tang and Raymond J Mooney. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Machine Learning: ECML 2001.* Springer, 2001.

S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th annual* *international ACM SIGIR conference on Research and development in informaion retrieval*, pages 41–47. ACM, 2003.

Alberto Tellez-Valero, Manuel Montes-y Gómez, Luis Villasenor Pineda, and Anselmo Penas. Towards multi-stream question answering using answer validation. *Informatica (Slovenia)*, 2010.

Kapil Thadani and Kathleen McKeown. Optimal and syntactically-informed decoding for monolingual phrase-based alignment. In *Proceedings of ACL short*, 2011.

Kapil Thadani, Scott Martin, and Michael White. A joint phrasal and dependency model for paraphrase alignment. In *Proceedings of COLING 2012: Posters*, pages 1229–1238, Mumbai, India, December 2012. The COLING 2012 Organizing Committee.

Cynthia A Thompson and Raymond J Mooney. Acquiring word-meaning mappings for natural language interfaces. *Journal of Artificial Intelligence Research*, 18(1):1–44, 2003.

H. Turtle and W.B. Croft. Inference networks for document retrieval. In *Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 1–24. ACM, 1989.

H. Turtle, Y. Hegde, and S.A. Rowe. Yet another comparison of lucene and indri performance. *Open Source Information Retrieval*, page 64, 2012.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-based question answering over RDF data. In *Proceedings of the 21st international conference on World Wide Web*, 2012.

Surya Ganesh Veeravalli and Vasudeva Varma. Passage retrieval using answer type profiles in question answering. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation*, pages 559–568, Hong Kong, December 2009. City University of Hong Kong.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. HMM-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics - Volume 2*, COLING '96, pages 836–841, 1996.

Ellen M Voorhees. The trec-8 question answering track report. In *TREC*, volume 99, pages 77–82, 1999.

Ellen M. Voorhees. Overview of the TREC 2001 Question Answering Track. In *TREC*, 2001.

Ellen M Voorhees and Donna Harman. Overview of the Sixth Text REtrieval Conference (TREC-6). *Information Processing & Management*, 36(1):3–35, 2000.

David L Waltz. An english language question answering system for a large relational database. *Communications of the ACM*, 21(7):526–539, 1978.

Mengqiu Wang. A Survey of Answer Extraction Techniques in Factoid Question Answering. CMU 11-762 Language and Statistics II literature review project, 2006.

Mengqiu Wang and Christopher D. Manning. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1164–1172, Stroudsburg, PA, USA, 2010.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 22–32, Prague, Czech Republic, June 2007.

Xing Wei, W. Bruce Croft, and Andrew Mccallum. Table extraction for answer retrieval. *Information Retrieval*, 9:589–611, 2006.

Ralph M Weischedel. Research and development in natural language understanding as part of the strategic computing program. Technical report, BBN Systems and Technologies Corporation, 1989.

Robert Wilensky, David N Chin, Marc Luria, James Martin, James Mayfield, and Dekai Wu. The berkeley unix consultant project. *Computational Linguistics*, 14(4):35–84, 1988.

Werner Winiwarter. An adaptive natural language interface architecture to access faq knowledge bases. In *Proc. of the 4th Int. Conf. on Applications of NL to Information Systems*, 1999.

Terry Winograd. Understanding natural language. *Cognitive psychology*, 3(1):1–191, 1972.

Travis Wolfe, Benjamin Van Durme, Mark Dredze, Nicholas Andrews, Charley Beller, Chris Callison-Burch, Jay DeYoung, Justin Snyder, Jonathan Weese, Tan Xu, and Xuchen Yao. PARMA: A Predicate Argument Aligner. In *Proceedings of ACL short*, 2013.

Yuk Wah Wong and Raymond J Mooney. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of ACL*, 2007.

William A Woods. Lunar rocks in natural english: Explorations in natural language question answering. *Linguistic structures processing*, 5:521–569, 1977.

William A. Woods. Semantics and quantification in natural language question answering. *Advances in computers*, 17(3), 1978.

Min Wu, Mingyuan Duan, Samira Shaikh, Sharon Small, and Tomek Strzalkowski. ILQUA–An IE-Driven Question Answering System. In Ellen M. Voorhees and Lori P. Buckland, editors, *TREC*, volume Special Publication 500-266. National Institute of Standards and Technology (NIST), 2005.

Jinxi Xu, Ana Licuanan, Jonathan May, Scott Miller, and Ralph M Weischedel. Trec 2002 qa at bbn: Answer selection and confidence estimation. In *TREC*, 2002.

Jinxi Xu, Ana Licuanan, and Ralph M Weischedel. TREC 2003 QA at BBN: Answering Definitional Questions. In *TREC*, pages 98–106, 2003.

Wei Xu, Alan Ritter, and Ralph Grishman. Gathering and Generating Paraphrases from Twitter with Application to Normalization. In *Proceedings of the 6th Workshop on Building and Using Comparable Corpora (BUCC)*, Sofia, Bulgaria, August 2013.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. Natural language questions for the web of data. In *Proceedings of EMNLP*, 2012.

Hui Yang, Hang Cui, Mstislav Maslennikov, Long Qiu, Min-Yen Kan, and Tat-Seng Chua. Qualifier in trec-12 qa main task. In *TREC*, pages 480–488, 2003.

Xuchen Yao and Benjamin Van Durme. Information Extraction over Structured Data: Question Answering with Freebase. In *Proceedings of ACL*, Baltimore, MD, USA, 2014.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. Semi-Markov Phrase-based Monolingual Alignment. In *Proceedings of EMNLP*, 2013a.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. A Lightweight and High Performance Monolingual Word Aligner. In *Proceedings of ACL short*, Sofia, Bulgaria, 2013b.

Xuchen Yao, Benjamin Van Durme, and Peter Clark. Automatic Coupling of Answer Extraction and Information Retrieval. In *Proceedings of ACL short*, Sofia, Bulgaria, 2013c.

Xuchen Yao, Benjamin Van Durme, Peter Clark, and Chris Callison-Burch. Answer Extraction as Sequence Tagging with Tree Edit Distance. In *Proceedings of NAACL*, 2013d.

Xuchen Yao, Jonathan Berant, and Benjamin Van Durme. Freebase QA: Information Extraction or Semantic Parsing? In *Proceedings of ACL Workshop on Semantic Parsing*, 2014.

Rémi Zajac. Towards ontological question answering. In *Proceedings of the workshop on Open-domain question answering-Volume 12*, pages 1–7, 2001.

John M Zelle and Raymond J Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1050–1055, 1996.

Luke S Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *Uncertainty in Artificial Intelligence (UAI)*, 2005.

Luke S. Zettlemoyer and Michael Collins. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of EMNLP-CoNLL*, 2007.

Luke S Zettlemoyer and Michael Collins. Learning context-dependent mappings from sentences to logical form. In *Proceedings of ACL-CoNLL*, 2009.

D. Zhang and W.S. Lee. Question classification using support vector machines. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 26–32. ACM, 2003.

K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262, December 1989.

Xian Zhang, Yu Hao, Xiaoyan Zhu, Ming Li, and David R. Cheriton. Information distance from a question to an answer. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 874–883, New York, NY, USA, 2007. ACM.

L. Zhao and J. Callan. A generative retrieval model for structured documents. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 1163–1172. ACM, 2008.

Zhiping Zheng. AnswerBus question answering system. In *Proceedings of the second international conference on Human Language Technology Research*, pages 399–404. Morgan Kaufmann Publishers Inc., 2002.

George Kingsley Zipf. *The psycho-biology of language: an introduction to dynamic philology.* Boston: Houghton Mifflin company, 1935.