

Theory and Applications of Natural Language Processing

Edited volumes

Imed Zitouni *Editor*

# Natural Language Processing of Semitic Languages



Springer

# Theory and Applications of Natural Language Processing

Series Editors:

Graeme Hirst (Textbooks)

Eduard Hovy (Edited volumes)

Mark Johnson (Monographs)

## Aims and Scope

The field of Natural Language Processing (NLP) has expanded explosively over the past decade: growing bodies of available data, novel fields of applications, emerging areas and new connections to neighboring fields have all led to increasing output and to diversification of research.

“Theory and Applications of Natural Language Processing” is a series of volumes dedicated to selected topics in NLP and Language Technology. It focuses on the most recent advances in all areas of the computational modeling and processing of speech and text across languages and domains. Due to the rapid pace of development, the diversity of approaches and application scenarios are scattered in an ever-growing mass of conference proceedings, making entry into the field difficult for both students and potential users. Volumes in the series facilitate this first step and can be used as a teaching aid, advanced-level information resource or a point of reference.

The series encourages the submission of research monographs, contributed volumes and surveys, lecture notes and textbooks covering research frontiers on all relevant topics, offering a platform for the rapid publication of cutting-edge research as well as for comprehensive monographs that cover the full range of research on specific problem areas.

The topics include applications of NLP techniques to gain insights into the use and functioning of language, as well as the use of language technology in applications that enable communication, knowledge management and discovery such as natural language generation, information retrieval, question-answering, machine translation, localization and related fields.

The books are available in printed and electronic (e-book) form:

- \* Downloadable on your PC, e-reader or iPad
- \* Enhanced by Electronic Supplementary Material, such as algorithms, demonstrations, software, images and videos
- \* Available online within an extensive network of academic and corporate R&D libraries worldwide
- \* Never out of print thanks to innovative print-on-demand services
- \* Competitively priced print editions for eBook customers thanks to MyCopy service <http://www.springer.com/librarians/e-content/mycopy>

For other titles published in this series, go to  
[www.springer.com/series/8899](http://www.springer.com/series/8899)

Imed Zitouni

Editor

# Natural Language Processing of Semitic Languages



Springer

*Editor*

Imed Zitouni  
Microsoft  
Redmond, WA  
USA

ISSN 2192-032X  
ISBN 978-3-642-45357-1  
DOI 10.1007/978-3-642-45358-8  
Springer Heidelberg New York Dordrecht London

ISSN 2192-0338 (electronic)  
ISBN 978-3-642-45358-8 (eBook)

Library of Congress Control Number: 2014936994

© Springer-Verlag Berlin Heidelberg 2014

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

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

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media ([www.springer.com](http://www.springer.com))

# Preface

Modern communication technologies, such as the television and the Internet, have made readily available massive amounts of information in many languages. More such data is being generated in real time, 24 h a day and 7 days a week, aided by social networking sites such as Facebook and Twitter. This information explosion is in the form of multilingual audio, video, and Web content. The task of processing this large amount of information demands effective, scalable, multilingual media processing, monitoring, indexing, and search solutions. Natural Language Processing (NLP) technologies have long been used to address this task, and several researchers have developed several technical solutions for it. In the last two decades, NLP researchers have developed exciting algorithms for processing large amounts of text in many different languages. Nowadays the English language has obtained the lion's share in terms of available resources as well as developed NLP technical solutions. In this book, we address another group of interesting and challenging languages for NLP research, that is, the Semitic languages. The Semitic languages have existed in written form since a very early date, with texts written in a script adapted from Sumerian cuneiform. Most scripts used to write Semitic languages are *abjads*, a type of alphabetic script that omits some or all of the vowels. This is feasible for these languages because the consonants in the Semitic languages are the primary carriers of meaning. Semitic languages have interesting morphology, where word roots are not themselves syllables or words, but isolated sets of consonants (usually three characters). Words are composed out of roots by adding vowels to the root consonants (although prefixes and suffixes are often added as well). For example, in Arabic, the root meaning “write” has the form k - t - b. From this root, words are formed by filling in the vowels, e.g., kitAb “book,” kutub “books,” kAtib “writer,” kuttAb “writers,” kataba “he wrote,” yaktubu “he writes,” etc. Semitic languages, as stated in Wikipedia, are spoken by more than 270 million people. The most widely spoken Semitic languages today are Arabic (206 million native speakers), Amharic (27 million), Hebrew (7 million), Tigrinya (6.7 million), Syriac (1 million), and Maltese (419 thousand). NLP research applied to Semitic languages has been the focus of attention of many researchers for more than a decade, and several technical solutions have been proposed, especially

Arabic NLP where we find a very large amount of accomplished research. This will be reflected in this book, where Arabic will take the lion’s share. Hebrew also has been the center of attention of several NLP research works, but to a smaller degree when compared to Arabic. Most of the key published research works in Hebrew NLP will be discussed in this book. For Amharic, Maltese, and Syriac, because of the very limited amount of NLP research publicly available, we didn’t limit ourselves to present key techniques, but we also proposed solutions inspired from Arabic and Hebrew. Our aim for this book is to provide a “one-stop shop” to all the requisite background and practical advice when building NLP applications for Semitic languages. While this is quite a tall order, we hope that, at a minimum, you find this book a useful resource.

Similar to English, the dominant approach in NLP for Semitic languages has been to build a statistical model that can learn from examples. In this way, a model can be robust to changes in the type of text and even the language of text on which it operates. With the right design choices, the same model can be trained to work in a new domain simply by providing new examples in that domain. This approach also obviates the need for researchers to lay out, in a painstaking fashion, all the rules that govern the problem at hand and the manner in which those rules must be combined. A statistical system typically allows for researchers to provide an abstract expression of possible *features* of the input, where the relative importance of those features can be learned during the *training* phase and can be applied to new text during the *decoding*, or *inference*, phase. While this book will devote some attention to cutting-edge algorithms and techniques, the primary purpose will be a thorough explication of best practices in the field. Furthermore, every chapter describes how the techniques discussed apply to Semitic languages.

This book is divided into two parts. Part I, includes the first five chapters and lays out several core NLP problems and algorithms to attack those problems. The first chapter introduces some basic linguistic facts about Semitic languages, covering orthography, morphology, and syntax. It also shows a contrastive analysis of some of these linguistic phenomena across the various languages. The second chapter introduces the important concept of *morphology*, the study of the structure of words, and ways to process the diverse array of morphologies present in Semitic languages. Chapter 3 investigates the various methods of uncovering a sentence’s internal structure, or *syntax*. Syntax has long been a dominant area of research in NLP. This dominance is explained in part by the fact that the structure of a sentence is related to the sentence’s meaning, and so uncovering syntactic structure can serve as a first step toward “understanding” a sentence. One step beyond syntactic parsing toward understanding a sentence is to perform semantic parsing that consists in finding a structured meaning representation for a sentence or a snippet of text. This is the focus of Chap. 4 that also covers a related subproblem known as *semantic role labeling*, which attempts to find the syntactic phrases that constitute the *arguments* to some verb or *predicate*. By identifying and classifying a verb’s arguments, we come closer to producing a *logical form* for a sentence, which is one way to represent a sentence’s meaning in such a way as to be readily processed by machine. In several NLP applications, one simply wants to accurately estimate the likelihood

of a word (or word sequence) in a phrase or sentence, without the need to analyze syntactic or semantic structure. The history, or context, that is used to make that estimation might be long or short, knowledge rich, or knowledge poor. The problem of producing a likelihood or probability estimate for a word is known as *language modeling*, and is the subject of Chap. 5.

Part II, takes the various core areas of NLP described in Part I and explains how they are applied to real-world NLP applications available nowadays for several Semitic languages. Chapters in this part of the book explore several tradeoffs in making various algorithmic and design choices when building a robust NLP application for Semitic languages, mainly Arabic and Hebrew. Chapter 6 describes one of the oldest problems in the field, namely, *machine translation*. Automatically translating from one language to another has long been a holy grail of NLP research, and in recent years the community has developed techniques that make machine translation a practical reality, reaping rewards after decades of effort. This chapter discusses recent efforts and techniques in translating Semitic languages such as Arabic and Hebrew to and from English.

The following three chapters focus on the core parts of a larger application area known as *information extraction*. Chapter 7, describes ways to identify and classify *named entities* in text. Chapter 8, discusses the linguistic relation between two textual entities which is determined when a textual entity (the anaphor) refers to another entity of the text which usually occurs before it (the antecedent). The last chapter of this trilogy, Chap. 9, continues the information extraction discussion, exploring techniques for finding out how two entities are related to each other, known as *relation extraction*.

The subject of finding few documents or subparts of documents that are relevant based on a search query is clearly an important NLP problem, as it shows the popularity of search engines such as Bing or Google. This problem is known as *information retrieval* and is the subject of Chap. 10. Another way in which we might tackle the sheer quantity of text is by automatically summarizing it. This is the content of Chap. 11. This problem either involves finding the sentences, or bits of sentences, that contribute toward providing a relevant summary, or else ingesting the text, summarizing its meaning in some internal representation, and then *generating* the text that constitutes a summary. Often, humans would like machines to process text automatically because they have questions they seek to answer. These questions can range from simple, factoid-like questions, such as “what is the family of languages to which Hebrew belongs?” to more complex questions such as “what political events succeeded the Tunisian revolution?” Chapter 12 discusses recent techniques to build systems to answer these types of questions automatically.

In several cases, we would like our speech to be automatically transcribed so that we can interact more naturally with machines. The process of converting speech into text, known as *Automatic Speech Recognition*, is the subject of Chap. 13. Plenty of advances have been made in the recent years in Arabic speech recognition. Current systems for Arabic and Hebrew achieve very reasonable performance and can be used in real NLP applications.

As much as we hope this book is self-contained and covers most research work in NLP for Semitic languages, we also hope that for you, the reader, it serves as the beginning and not an end. Each chapter has a long list of relevant work upon which it is based, allowing you to explore any subtopic in great detail. The large community of NLP researchers is growing throughout the world, and we hope you join us in our exciting efforts to process language automatically and that you interact with us at universities, at research labs, at conferences, on social networks, and elsewhere. NLP systems of the future and their application to Semitic languages are going to be even more exciting than the ones we have now, and we look forward to all your contributions!

Bellevue WA, USA  
March 2014

Imed Zitouni

# Acknowledgments

This book is the result of a highly collaborative effort. I am immensely grateful for the encouraging support obtained from Springer, especially from Olga Chiarcos who helped this project to get off the ground and from Federica Corradi dell Acqua for her support during the different stages of this project until completion. I am also grateful to Lyn Imeson for her efficiency in copyediting this book.

A book of this kind would also not have been possible without the effort and technical acumen of my fellow chapter authors, so I owe huge thanks to Ray Fabri, Michael Gasser, Nizar Habash, George Kiraz, Shuly Wintner, Mouna Diab, Yuval Marton, Ilana Heintz, Hany Hassan, Kareem Darwish, Behrang Mohit, Khadiga Mahmoud Seddik, Ali Farghaly, Vittorio Castelli, Yassine Benajiba, Paolo Rosso, Lahsen Abouenour, Omar Trigui, Karim Bouzoubaa, Lamia Belguith, Marem Ellouze, Mohamed H. Maaloul, Maher Jaoua, Fatma K. Jaoua, Philippe Blache, Hagen Soltau, George Saon, Lidia Mangu, Hong-Kwang Kuo, Brian Kingsbury, Stephen Chu, and Fadi Biadsy.

Finally, I am very grateful to my fellow technical review committee for their generous time, effort, and valuable feedback that contributed significantly in improving the technical quality of the chapters. Huge thanks to Chafik Aloulou, Mohammed Afify, Yassine Benajiba, Joseph Dichy, Yi Fang, Kavita Ganesan, Spence Green, Dilek Hakkani-Tur, Xiaoqiang Luo, Siddharth Patwardhan, Eric Ringger, Murat Saraclar, Ruhi Sarikaya, Lucy Vanderwende, and Bing Zhao.

Imed Zitouni



# **Technical Review Committee**

- Chafik Aloulou (University of Sfax, Sfax, Tunisia)
- Mohammed Afify (Orange Lab, Cairo, Egypt)
- Yassine Benajiba (Thomson Reuters, NY)
- Joseph Dichy (University of Lyon 2, Lyon, France)
- Yi Fang (Santa Clara University, CA)
- Kavita Ganesan (UIUC, IL)
- Spence Green (Stanford, CA)
- Dilek Hakkani-Tur (Microsoft, CA)
- Xiaoqiang Luo (Google, NY)
- Siddharth Patwardhan (IBM, NY)
- Eric Ringger (Brigham Young University, UT)
- Murat Saraclar (Bogazici University, Istanbul Turkey)
- Ruhi Sarikaya (Microsoft, WA)
- Lucy Vanderwende (Microsoft, WA)
- Bing Zhao (SRI, CA)



# Contents

## Part I Natural Language Processing Core-Technologies

<b>1 Linguistic Introduction: The Orthography, Morphology and Syntax of Semitic Languages .....</b>	<b>3</b>
Ray Fabri, Michael Gasser, Nizar Habash, George Kiraz, and Shuly Wintner	
1.1 Introduction .....	3
1.2 Amharic .....	5
1.2.1 Orthography .....	6
1.2.2 Derivational Morphology .....	7
1.2.3 Inflectional Morphology .....	9
1.2.4 Basic Syntactic Structure .....	11
1.3 Arabic .....	13
1.3.1 Orthography .....	14
1.3.2 Morphology .....	15
1.3.3 Basic Syntactic Structure .....	18
1.4 Hebrew .....	19
1.4.1 Orthography .....	20
1.4.2 Derivational Morphology .....	22
1.4.3 Inflectional Morphology .....	23
1.4.4 Morphological Ambiguity .....	25
1.4.5 Basic Syntactic Structure .....	25
1.5 Maltese .....	26
1.5.1 Orthography .....	26
1.5.2 Derivational Morphology .....	27
1.5.3 Inflectional Morphology .....	29
1.5.4 Basic Syntactic Structure .....	30
1.6 Syriac .....	32
1.6.1 Orthography .....	32
1.6.2 Derivational Morphology .....	33

1.6.3	Inflectional Morphology .....	33
1.6.4	Syntax .....	34
1.7	Contrastive Analysis .....	34
1.7.1	Orthography .....	34
1.7.2	Phonology .....	35
1.7.3	Morphology .....	36
1.7.4	Syntax .....	37
1.7.5	Lexicon .....	37
1.8	Conclusion .....	38
	References .....	38
<b>2</b>	<b>Morphological Processing of Semitic Languages .....</b>	<b>43</b>
	Shuly Wintner	
2.1	Introduction .....	43
2.2	Basic Notions .....	44
2.3	The Challenges of Morphological Processing .....	45
2.4	Computational Approaches to Morphology .....	47
2.4.1	Two-Level Morphology .....	48
2.4.2	Multi-tape Automata .....	48
2.4.3	The Xerox Approach .....	49
2.4.4	Registered Automata .....	50
2.4.5	Analysis by Generation .....	50
2.4.6	Functional Morphology .....	51
2.5	Morphological Analysis and Generation of Semitic Languages .....	51
2.5.1	Amharic .....	52
2.5.2	Arabic .....	52
2.5.3	Hebrew .....	54
2.5.4	Other Languages .....	55
2.5.5	Related Applications .....	55
2.6	Morphological Disambiguation of Semitic Languages .....	56
2.7	Future Directions .....	58
	References .....	58
<b>3</b>	<b>Syntax and Parsing of Semitic Languages .....</b>	<b>67</b>
	Reut Tsarfaty	
3.1	Introduction .....	67
3.1.1	Parsing Systems .....	69
3.1.2	Semitic Languages .....	74
3.1.3	The Main Challenges .....	80
3.1.4	Summary and Conclusion .....	84
3.2	Case Study: Generative Probabilistic Parsing .....	84
3.2.1	Formal Preliminaries .....	85
3.2.2	An Architecture for Parsing Semitic Languages .....	91
3.2.3	The Syntactic Model .....	99
3.2.4	The Lexical Model .....	113

3.3	Empirical Results.....	117
3.3.1	Parsing Modern Standard Arabic .....	117
3.3.2	Parsing Modern Hebrew.....	120
3.4	Conclusion and Future Work .....	123
	References.....	124
<b>4</b>	<b>Semantic Processing of Semitic Languages .....</b>	<b>129</b>
	Mona Diab and Yuval Marton	
4.1	Introduction.....	129
4.2	Fundamentals of Semitic Language Meaning Units .....	130
4.2.1	Morpho-Semantics: A Primer .....	130
4.3	Meaning, Semantic Distance, Paraphrasing and Lexicon Generation .....	135
4.3.1	Semantic Distance .....	136
4.3.2	Textual Entailment.....	138
4.3.3	Lexicon Creation .....	138
4.4	Word Sense Disambiguation and Meaning Induction .....	139
4.4.1	WSD Approaches in Semitic Languages .....	140
4.4.2	WSI in Semitic Languages .....	141
4.5	Multiword Expression Detection and Classification.....	142
4.5.1	Approaches to Semitic MWE Processing and Resources .....	143
4.6	Predicate–Argument Analysis .....	145
4.6.1	Arabic Annotated Resources .....	146
4.6.2	Systems for Semantic Role Labeling .....	148
4.7	Conclusion .....	152
	References.....	152
<b>5</b>	<b>Language Modeling .....</b>	<b>161</b>
	Ilana Heintz	
5.1	Introduction.....	161
5.2	Evaluating Language Models with Perplexity .....	162
5.3	N-Gram Language Modeling .....	164
5.4	Smoothing: Discounting, Backoff, and Interpolation.....	166
5.4.1	Discounting .....	166
5.4.2	Combining Discounting with Backoff .....	168
5.4.3	Interpolation .....	168
5.5	Extensions to N-Gram Language Modeling .....	170
5.5.1	Skip N-Grams and FlexGrams .....	170
5.5.2	Variable-Length Language Models .....	171
5.5.3	Class-Based Language Models .....	173
5.5.4	Factored Language Models .....	174
5.5.5	Neural Network Language Models .....	175
5.5.6	Syntactic or Structured Language Models.....	177
5.5.7	Tree-Based Language Models .....	178
5.5.8	Maximum-Entropy Language Models .....	178

5.5.9	Discriminative Language Models .....	180
5.5.10	LSA Language Models .....	183
5.5.11	Bayesian Language Models .....	184
5.6	Modeling Semitic Languages .....	187
5.6.1	Arabic .....	188
5.6.2	Amharic .....	189
5.6.3	Hebrew .....	191
5.6.4	Maltese .....	191
5.6.5	Syriac .....	192
5.6.6	Other Morphologically Rich Languages .....	192
5.7	Summary .....	193
	References .....	193

## Part II Natural Language Processing Applications

<b>6</b>	<b>Statistical Machine Translation .....</b>	199
	Hany Hassan and Kareem Darwish	
6.1	Introduction .....	199
6.2	Machine Translation Approaches .....	200
6.2.1	Machine Translation Paradigms .....	200
6.2.2	Rule-Based Machine Translation .....	202
6.2.3	Example-Based Machine Translation .....	202
6.2.4	Statistical Machine Translation .....	203
6.2.5	Machine Translation for Semitic Languages .....	203
6.3	Overview of Statistical Machine Translation .....	204
6.3.1	Word-Based Translation Models .....	204
6.3.2	Phrase-Based SMT .....	205
6.3.3	Phrase Extraction Techniques .....	206
6.3.4	SMT Reordering .....	207
6.3.5	Language Modeling .....	207
6.3.6	SMT Decoding .....	208
6.4	Machine Translation Evaluation Metrics .....	209
6.5	Machine Translation for Semitic Languages .....	210
6.5.1	Word Segmentation .....	210
6.5.2	Word Alignment and Reordering .....	211
6.5.3	Gender-Number Agreement .....	212
6.6	Building Phrase-Based SMT Systems .....	213
6.6.1	Data .....	213
6.6.2	Parallel Data .....	213
6.6.3	Monolingual Data .....	214
6.7	SMT Software Resources .....	214
6.7.1	SMT Moses Framework .....	214
6.7.2	Language Modeling Toolkits .....	214
6.7.3	Morphological Analysis .....	215

6.8	Building a Phrase-Based SMT System: Step-by-Step Guide .....	215
6.8.1	Machine Preparation.....	215
6.8.2	Data .....	216
6.8.3	Data Preprocessing .....	216
6.8.4	Words Segmentation.....	216
6.8.5	Language Model .....	217
6.8.6	Translation Model .....	217
6.8.7	Parameter Tuning .....	217
6.8.8	System Decoding .....	218
6.9	Summary .....	218
	References.....	218
7	<b>Named Entity Recognition .....</b>	221
	Behrang Mohit	
7.1	Introduction .....	221
7.2	The Named Entity Recognition Task .....	222
7.2.1	Definition .....	222
7.2.2	Challenges in Named Entity Recognition .....	223
7.2.3	Rule-Based Named Entity Recognition .....	224
7.2.4	Statistical Named Entity Recognition .....	225
7.2.5	Hybrid Systems .....	228
7.2.6	Evaluation and Shared Tasks .....	228
7.2.7	Evaluation Campaigns.....	229
7.2.8	Beyond Traditional Named Entity Recognition .....	230
7.3	Named Entity Recognition for Semitic Languages .....	230
7.3.1	Challenges in Semitic Named Entity Recognition .....	231
7.3.2	Approaches to Semitic Named Entity Recognition .....	232
7.4	Case Studies .....	233
7.4.1	Learning Algorithms .....	234
7.4.2	Features .....	234
7.4.3	Experiments .....	235
7.5	Relevant Problems .....	236
7.5.1	Named Entity Translation and Transliteration .....	236
7.5.2	Entity Detection and Tracking .....	238
7.5.3	Projection .....	238
7.6	Labeled Named Entity Recognition Corpora .....	239
7.7	Future Challenges and Opportunities.....	240
7.8	Summary .....	241
	References.....	241
8	<b>Anaphora Resolution .....</b>	247
	Khadiga Mahmoud Seddik and Ali Farghaly	
8.1	Introduction: Anaphora and Anaphora Resolution .....	247
8.2	Types of Anaphora .....	248
8.2.1	Pronominal Anaphora .....	248
8.2.2	Lexical Anaphora .....	249
8.2.3	Comparative Anaphora .....	249

8.3	Determinants in Anaphora Resolution .....	249
8.3.1	Eliminating Factors .....	250
8.3.2	Preferential Factors .....	251
8.3.3	Implementing Features in AR (Anaphora Resolution) Systems .....	252
8.4	The Process of Anaphora Resolution .....	256
8.5	Different Approaches to Anaphora Resolution .....	257
8.5.1	Knowledge-Intensive Versus Knowledge-Poor Approaches .....	257
8.5.2	Traditional Approach .....	259
8.5.3	Statistical Approach .....	259
8.5.4	Linguistic Approach to Anaphora Resolution .....	260
8.6	Recent Work in Anaphora and Coreference Resolution .....	262
8.6.1	Mention-Synchronous Coreference Resolution Algorithm Based on the Bell Tree [24] .....	262
8.6.2	A Twin-Candidate Model for Learning-Based Anaphora Resolution [47, 48] .....	263
8.6.3	Improving Machine Learning Approaches to Coreference Resolution [36] .....	264
8.7	Evaluation of Anaphora Resolution Systems .....	265
8.7.1	MUC [45] .....	265
8.7.2	B-Cube [2] .....	267
8.7.3	ACE (NIST 2003) .....	267
8.7.4	CEAF [23] .....	268
8.7.5	BLANC [40] .....	269
8.8	Anaphora in Semitic Languages .....	269
8.8.1	Anaphora Resolution in Arabic .....	270
8.9	Difficulties with AR in Semitic Languages .....	272
8.9.1	The Morphology of the Language .....	272
8.9.2	Complex Sentence Structure .....	273
8.9.3	Hidden Antecedents .....	273
8.9.4	The Lack of Corpora Annotated with Anaphoric Links .....	273
8.10	Summary .....	274
	References .....	274
9	<b>Relation Extraction .....</b>	279
	Vittorio Castelli and Imed Zitouni	
9.1	Introduction .....	279
9.2	Relations .....	280
9.3	Approaches to Relation Extraction .....	281
9.3.1	Feature-Based Classifiers .....	281
9.3.2	Kernel-Based Methods .....	285
9.3.3	Semi-supervised and Adaptive Learning .....	288

9.4	Language-Specific Issues .....	291
9.5	Data .....	292
9.6	Results .....	294
9.7	Summary.....	295
	References.....	295
<b>10</b>	<b>Information Retrieval .....</b>	<b>299</b>
	Kareem Darwish	
10.1	Introduction.....	299
10.2	The Information Retrieval Task.....	299
10.2.1	Task Definition.....	301
10.2.2	The General Architecture of an IR System .....	302
10.2.3	Retrieval Models.....	303
10.2.4	IR Evaluation .....	305
10.3	Semitic Language Retrieval.....	309
10.3.1	The Major Known Challenges .....	309
10.3.2	Survey of Existing Literature .....	313
10.3.3	Best Arabic Index Terms .....	316
10.3.4	Best Hebrew Index Terms .....	318
10.3.5	Best Amharic Index Terms .....	318
10.4	Available IR Test Collections .....	318
10.4.1	Arabic .....	318
10.4.2	Hebrew .....	319
10.4.3	Amharic .....	319
10.5	Domain-Specific IR .....	319
10.5.1	Arabic–English CLIR .....	320
10.5.2	Arabic OCR Text Retrieval .....	322
10.5.3	Arabic Social Search .....	326
10.5.4	Arabic Web Search .....	328
10.6	Summary.....	329
	References.....	329
<b>11</b>	<b>Question Answering .....</b>	<b>335</b>
	Yassine Benajiba, Paolo Rosso, Lahsen Abouenour, Omar Trigui, Karim Bouzoubaa, and Lamia Belguith	
11.1	Introduction .....	335
11.2	The Question Answering Task .....	336
11.2.1	Task Definition .....	336
11.2.2	The Major Known Challenges .....	338
11.2.3	The General Architecture of a QA System .....	339
11.2.4	Answering Definition Questions and Query Expansion Techniques .....	341
11.2.5	How to Benchmark QA System Performance: Evaluation Measure for QA .....	343

11.3	The Case of Semitic Languages .....	344
11.3.1	NLP for Semitic Languages .....	344
11.3.2	QA for Semitic Languages .....	345
11.4	Building Arabic QA Specific Modules .....	347
11.4.1	Answering Definition Questions in Arabic .....	347
11.4.2	Query Expansion for Arabic QA .....	353
11.5	Summary .....	366
	References .....	367
<b>12</b>	<b>Automatic Summarization .....</b>	<b>371</b>
	Lamia Hadrich Belguith, Mariem Ellouze, Mohamed Hedi Maaloul, Maher Jaoua, Fatma Kallel Jaoua, and Philippe Blache	
12.1	Introduction .....	371
12.2	Text Summarization Aspects .....	372
12.2.1	Types of Summaries .....	374
12.2.2	Extraction vs. Abstraction .....	375
12.2.3	The Major Known Challenges .....	376
12.3	How to Evaluate Summarization Systems .....	376
12.3.1	Insights from the Evaluation Campaigns .....	377
12.3.2	Evaluation Measures for Summarization .....	377
12.4	Single Document Summarization Approaches .....	378
12.4.1	Numerical Approach .....	379
12.4.2	Symbolic Approach .....	379
12.4.3	Hybrid Approach .....	380
12.5	Multiple Document Summarization Approaches .....	380
12.5.1	Numerical Approach .....	381
12.5.2	Symbolic Approach .....	382
12.5.3	Hybrid Approach .....	383
12.6	Case of Semitic Languages .....	385
12.6.1	Language-Independent Systems .....	385
12.6.2	Arabic Systems .....	386
12.6.3	Hebrew Systems .....	388
12.6.4	Maltese Systems .....	388
12.6.5	Amharic Systems .....	389
12.7	Case Study: Building an Arabic Summarization System (L.A.E) .....	389
12.7.1	L.A.E System Architecture .....	390
12.7.2	Source Text Segmentation .....	390
12.7.3	Interface .....	400
12.7.4	Evaluation and Discussion .....	401
12.8	Summary .....	402
	References .....	403

<b>13 Automatic Speech Recognition .....</b>	<b>409</b>
Hagen Soltau, George Saon, Lidia Mangu, Hong-Kwang Kuo, Brian Kingsbury, Stephen Chu, and Fadi Biadsy	
<b>13.1 Introduction.....</b>	<b>409</b>
<b>13.1.1 Automatic Speech Recognition .....</b>	<b>410</b>
<b>13.1.2 Introduction to Arabic: A Speech Recognition             Perspective .....</b>	<b>411</b>
<b>13.1.3 Overview .....</b>	<b>412</b>
<b>13.2 Acoustic Modeling .....</b>	<b>413</b>
<b>13.2.1 Language-Independent Techniques .....</b>	<b>413</b>
<b>13.2.2 Vowelization .....</b>	<b>418</b>
<b>13.2.3 Modeling of Arabic Dialects in Decision Trees .....</b>	<b>423</b>
<b>13.3 Language Modeling .....</b>	<b>428</b>
<b>13.3.1 Language-Independent Techniques for             Language Modeling .....</b>	<b>428</b>
<b>13.3.2 Language-Specific Techniques for Language             Modeling .....</b>	<b>432</b>
<b>13.4 IBM GALE 2011 System Description .....</b>	<b>434</b>
<b>13.4.1 Acoustic Models .....</b>	<b>434</b>
<b>13.4.2 Language Models .....</b>	<b>439</b>
<b>13.4.3 System Combination .....</b>	<b>441</b>
<b>13.4.4 System Architecture .....</b>	<b>441</b>
<b>13.5 From MSA to Dialects .....</b>	<b>443</b>
<b>13.5.1 Dialect Identification .....</b>	<b>443</b>
<b>13.5.2 ASR and Dialect ID Data Selection .....</b>	<b>446</b>
<b>13.5.3 Dialect Identification on GALE Data .....</b>	<b>447</b>
<b>13.5.4 Acoustic Modeling Experiments .....</b>	<b>448</b>
<b>13.5.5 Dialect ID Based on Text Only .....</b>	<b>452</b>
<b>13.6 Resources .....</b>	<b>453</b>
<b>13.6.1 Acoustic Training Data .....</b>	<b>453</b>
<b>13.6.2 Training Data for Language Modeling .....</b>	<b>454</b>
<b>13.6.3 Vowelization Resources .....</b>	<b>454</b>
<b>13.7 Comparing Arabic and Hebrew ASR .....</b>	<b>455</b>
<b>13.8 Summary.....</b>	<b>456</b>
<b>References .....</b>	<b>457</b>



## About the Editor



Dr. Imed Zitouni is a principal researcher at Microsoft leading the Relevance Measurement Sciences group. Imed received his M.Sc. and Ph.D. with the highest honors from the University of Nancy1, France. In 1995, he obtained a MEng degree in computer science from ENSI in Tunisia. He is a senior member of IEEE, served as a member of the IEEE Speech and Language Processing Technical Committee (1999–2011), the information officer of the ACL SIG on Semitic languages, associate editor of TALIP ACM journal, and a member of ISCA and ACL. Imed served as chair and reviewing committee member of several conferences and journals, and he is the author/coauthor of more than 100 patents and papers in international conferences and journals. Imed's research interest is in the area of Multilingual Natural Language Processing (NLP), including information retrieval, information extraction, language modeling, etc. Imed has particular interest in advancing state-of-the-art technology in the area of Semitic NLP, especially Arabic.

Imed's current research interest is in the area of Multilingual Information Retrieval focusing on the use of statistics and machine learning techniques to develop Web scale offline and online metrics. He is also working on the use of NLP to add a layer of semantics and understanding to search engines. Prior to

joining Microsoft, Imed was a senior researcher at IBM for almost a decade, where he led several Multilingual *NLP projects, including Arabic NLP, information extraction, semantic role labeling, language modeling, machine translation, and speech recognition*. Prior to IBM, Imed was a researcher at *Bell Laboratories*, Lucent Technologies, for almost half dozen years working on language modeling, speech recognition, spoken dialog systems, and speech understanding. Imed also experimented the startup experience at DIALOCA in Paris, France, working on e-mail steering and language modeling and served as temporary assistant professor at the University of Nancy 1, France.

**Part I**

**Natural Language Processing**

**Core-Technologies**

# Chapter 1

## Linguistic Introduction: The Orthography, Morphology and Syntax of Semitic Languages

**Ray Fabri, Michael Gasser, Nizar Habash, George Kiraz, and Shuly Wintner**

### 1.1 Introduction

We present in this chapter some basic linguistic facts about Semitic languages, covering orthography, morphology, and syntax. We focus on Arabic (both standard and dialectal), Ethiopian languages (specifically, Amharic), Hebrew, Maltese and Syriac. We conclude the chapter with a contrastive analysis of some of these phenomena across the various languages.

The Semitic family of languages [46, 57, 61] is spoken in the Middle East and North Africa, from Iraq and the Arabian Peninsula in the east to Morocco in the west, by over 300 million native speakers.<sup>1</sup> The most widely spoken Semitic languages today are Arabic, Amharic, Tigrinya and Hebrew. The situation of Arabic

---

<sup>1</sup>Parts of the following discussion are based on Wintner [74].

R. Fabri (✉)  
University of Malta, Msida, Malta  
e-mail: [ray.fabri@um.edu.mt](mailto:ray.fabri@um.edu.mt)

M. Gasser (✉)  
Indiana University, Bloomington, IN, USA  
e-mail: [gasser@cs.indiana.edu](mailto:gasser@cs.indiana.edu)

N. Habash (✉)  
Columbia University, New York, NY, USA  
e-mail: [habash@ccls.columbia.edu](mailto:habash@ccls.columbia.edu)

G. Kiraz (✉)  
Beth Mardutho: The Syriac Institute, Piscataway, NJ, USA  
e-mail: [gkiraz@gorgiaspress.com](mailto:gkiraz@gorgiaspress.com)

S. Wintner  
University of Haifa, Haifa, Israel  
e-mail: [shuly@cs.haifa.ac.il](mailto:shuly@cs.haifa.ac.il)

(and Syriac) is particularly interesting from a sociolinguistic point of view, as it represents an extreme case of *diglossia*: Modern Standard Arabic (MSA) is used in written texts and formal speech across the Arab world, but is not spoken natively. Rather, colloquial Arabic dialects (Levantine, Yemenite, etc.) are used for everyday conversation, but lack an agreed-upon script [53, p. 267].

The most prominent phenomenon of Semitic languages is the reliance on *root-and-pattern* paradigms for word formation. The standard account of word-formation processes in Semitic languages [59] describes words as combinations of two morphemes: a *root* and a *pattern*.<sup>2</sup> The root consists of consonants only, by default three, called *radicals*. The pattern is a combination of vowels and, possibly, consonants too, with ‘slots’ into which the root consonants can be inserted. Words are created by *interdigitating* roots into patterns: the consonants of the root fill the slots of the pattern, by default in linear order (see Shimron [67] for a survey).

Other morphological processes in Semitic languages involve affixation and cliticization. The various languages discussed in this chapter exhibit prefixes, suffixes, infixes and circumfixes; some of these affixes are sometimes referred to as (pro- and en-) clitics; we blur the distinction between affixes and clitics [76, 77] in the sequel.

Root-and-pattern morphology, a non-concatenative mechanism unique to Semitic languages, is one of the main challenges for computational processing. Since the vast majority of the morphological processes known in other languages *are* concatenative, existing computational solutions often fail in the face of Semitic interdigitation. The rich morphology of Semitic languages, coupled with deficiencies of the orthography of many of these languages, result in a high level of *ambiguity*, another hurdle that computational processing must overcome.

This chapter aims to describe basic linguistic facets of Semitic languages, addressing issues of orthography, morphology and syntax. We focus on five languages: Amharic (Sect. 1.2), Arabic (both standard and dialectal, Sect. 1.3), Hebrew (Sect. 1.4), Maltese (Sect. 1.5) and Syriac (Sect. 1.6). We conclude with a contrastive analysis of the similarities and differences among those languages (Sect. 1.7), a discussion that provides a backdrop for future chapters, which focus on computational processing addressing these issues. Throughout this chapter, phonetic forms are given between [square brackets], phonemic forms are between /slashes/ and glosses are listed between “double quotes”.

---

<sup>2</sup>In fact, McCarthy [59] abstracts the pattern further by assuming an additional morpheme, *vocalization* (or *vocalism*), but we do not need this level of abstraction here. Many terms are used to refer to the concept “pattern”. In addition to *pattern* and *template*, researchers may encounter *wazn* (from Arabic grammar), *binyan* (from Hebrew grammar), *form* and *measure*. The term *pattern* is used ambiguously to include or exclude vocalisms, i.e., *vocalism-specified pattern* and *vocalism-free pattern* [41].

## 1.2 Amharic

Amharic is the official language of Ethiopia, the first language of approximately 30 % of the population of the country, 21,631,370 people, according to the 2007 census. As the working language of the federal government and the language of education in many regions of Ethiopia outside the Amhara region, Amharic is also spoken by many Ethiopians as a second language. It is also the native language of perhaps several million Ethiopian immigrants, especially in North America and Israel.

Amharic belongs to the well-established Ethiopian Semitic (or Ethio-Semitic, occasionally also African Semitic) branch of South Semitic. Like the 11<sup>3</sup> other Ethiopian Semitic languages, Amharic exhibits typical Semitic behavior, in particular the pattern of inflectional and derivational morphology, along with some characteristic Ethiopian Semitic features, such as SOV word order, which are generally thought to have resulted from long contact with Cushitic languages.

Among the other Ethiopian Semitic languages, the most important is Tigrinya, with approximately 6.5 million speakers. It is one of the official languages of Eritrea, where it is spoken by more than half of the population, and the official language of Ethiopia's northernmost province of Tigray. Most of what appears in this section applies to Tigrinya as well as Amharic. The most important differences are as follows.

- Like other Semitic languages, but unlike Amharic, Tigrinya has productive broken plurals, as well as external suffixed plurals.
- Alongside the prepositional possessive construction, as in Amharic, Tigrinya has a possessive construction similar to the Arabic Idafa, although neither noun is marked for “state”.
- Tigrinya has compound prepositions corresponding to the preposition-postposition compounds found in Amharic.
- The negative circumfix may mark nouns, adjectives, and pronouns, as well as verbs.
- The definite article is an independent word, similar to a demonstrative adjective, rather than a noun suffix.
- Accusative case is marked with a preposition rather than a suffix.
- Yes-no questions are marked with a suffix on the word being questioned.
- There is an unusually complex tense-aspect-mood system, with many nuances achieved using combinations of the three basic aspectual forms (perfect, imperfect, gerund) and various auxiliary verbs.

---

<sup>3</sup>Because the language boundaries are not well-defined within the Gurage dialects, there is no general agreement on how many Ethiopian Semitic languages there are.

### 1.2.1 Orthography

Unlike Arabic, Hebrew, and Syriac, Amharic is written using a syllabic writing system, one originally developed for the extinct Ethiopian Semitic language Ge'ez and later extended for Amharic and other Ethiopian Semitic languages [18, 65]. As in other *abugida*<sup>4</sup> systems, each character of the Ge'ez (or Ethiopic) writing system gets its basic shape from the consonant of the syllable, and the vowel is represented through more or less systematic modifications of these basic shapes.<sup>5</sup>

Amharic has 26 consonant phonemes, 27 if we count the /v/ that is used in foreign loan-words, and 7 vowels. For four of the consonants (/ʃ/, /h/, /s/, and /s'/), the writing system makes distinctions that existed in Ge'ez but have been lost in Amharic. There is no single accepted way to transliterate Amharic. We use the following symbols to represent consonants (in the traditional order): *h, l, m, s, r, š, q, b, t, č, n, ŋ, ', k, w, z, ž, y, d, j, g, t', č', p', s', f, p*. And we use these symbols for the vowels, again in the traditional order: *ə, u, i, a, e, i, o*. Thus the basic character set consists of one character for each combination of 33 consonants and 7 vowels. There are also 37 further characters representing labialized variants of the consonants followed by particular vowels. The complete system has 268 characters. There is also a set of Ge'ez numerals, but nowadays these tend to be replaced by the Hindu-Arabic numerals used in European languages.

Although the Amharic writing system is far less ambiguous than the Arabic, Hebrew, and Syriac alphabets when these are used without vocalization diacritics, the system does embody two sorts of ambiguity. First, as in other abugida systems, one of the characters in each consonant set has a dual function: it represents that consonant followed by a particular vowel, and it represents a bare consonant, that is, the consonant as the coda of a syllable. In the Ge'ez writing system, the vowel for these characters is the high central vowel /i/, traditionally the sixth vowel in the set of characters for a given consonant. This ambiguity presents no serious problems because the vowel /i/ is usually epenthetic so that its presence or absence is normally predictable from the phonetic context.

A more serious sort of ambiguity results from the failure of the writing system to indicate consonant gemination. Gemination is a characteristic feature of Amharic, possible with all but two of the consonants and in all positions except initially. Gemination has both lexical and grammatical functions and is quite common; most words contain at least one geminated consonant, and spoken Amharic lacking gemination sounds quite unnatural [4]. In practice, there are relatively few minimal pairs because of redundancy but the existing minimal pairs include some very common words, for example, *alə* “he said”, *allə* “there is”. Syntax must be relied on

---

<sup>4</sup>In fact the name *abugida*, representing a category of writing system with scores of exemplars, comes from one name of the Ge'ez script as well as the pronunciations of the first four characters of the script in one traditional ordering.

<sup>5</sup>For the actual character set, see [http://en.wikipedia.org/wiki/Amharic\\_language#Writing\\_system](http://en.wikipedia.org/wiki/Amharic_language#Writing_system).

to disambiguate these words. The fact that there are few minimal pairs differing only in gemination means that gemination can usually be restored from the orthography, permitting text-to-speech systems to incorporate this crucial feature [4].

### 1.2.2 Derivational Morphology

#### Lexicon

As in other Semitic languages, verb roots are the basis of much of the lexicon of Ethiopian Semitic languages.

Alongside verbs proper, there is the category of so-called composite verbs, a defining feature of Ethiopian Semitic languages [3]. These consist of a word in an invariant form, the composite verb “lexeme”, followed immediately by an inflected form of one or another of the verbs *alə* “say”, *adərrəgə* “do, make”, or *assəjŋə* “cause to feel”. A small number of the composite verb lexemes are monomorphemic, underived forms, for example, *qučč’* “sit”, *zimm* “be quiet”, but most are derived from verb roots (see below), for example, *kiffitt* from the root *k.f.t* “open”. In combination with *alə*, the result is an intransitive verb; in combination with *adərrəgə* or *assəjŋə* a transitive verb.

Nouns, adjectives, and numerals have similar morphosyntactic properties. Nouns are specified for definiteness, but there is no counterpart to the “status” (construct vs. absolute) known in Arabic or Hebrew. Many nouns are derived from verb roots, for example, *nəji* “driver” from *n.d.* “drive”, *t’iqs* “quotation” from *t’.q.:s* “quote”, *mələt’t’əfiya* “glue” from *l.t’:f* “stick (together)”. Many others are not, for example, *bet* “house”, *sar* “grass”, *hod* “stomach”.

The great majority of adjectives are derived from verb roots, for example, *kifu* “bad, evil” from *k.f.* “become bad, evil”, *ləflafi* “talkative” from *l.f.l.f* “talk too much”, *birtu* “strong” from *b.r.t.* “become strong”.

Amharic has a small number of monomorphemic, underived adverbs and conjunctions, for example, *məce* “when”, *gin* “but”. However, most expressions with adverbial or conjunctive function consist of nouns, usually with prepositional prefixes and possibly postpositions as well, or subordinate clauses of one sort or another, for example, *məjəmməriya* “first”, lit. “beginning (n.)”; *bəgimmit* “approximately”, lit. “by guess”; *siləzzih* “therefore”, lit. “because of this”. Especially common are clauses with a verb in the gerund form, including composite verbs including a gerund form of the verbs *alə* or *adərrəgə*: *qəss bilo* “slowly (3 pers.sing.masc.)”, *zimm biyye* “silently, without doing anything (1 pers.sing.)”. Note that the gerund, like all Amharic verbs, agrees with its subject, so these adverbials vary with the main clause subject.

Amharic has approximately nine prepositions. The one-syllable prepositions are treated as prefixes; the two-syllable prepositions may be written as prefixes or as separate words. Prepositions occur with and without a much larger set of postpositions. Examples: *betu* “the house”, *i betu* “at/in the house”, *wədə betu* “to

the house”, *i̥betu wist’* “inside the house”, *kəbetu wist’* “from inside the house”, *i̥betu at’əgəb* “next to the house”.

## Root and Pattern Processes

Amharic morphology is very complex (for details, see Leslau [56] or Tefera and Hudson [69]). As in other Semitic languages, word formation relies primarily on root-and-pattern morphology. As an example, consider the root *s.b.r*, which is the basis of many words having to do with the notion of breaking. Within the verb system, each root appears in four different tense-aspect-mood (TAM) forms and up to ten different forms representing various derivational categories such as causative and reciprocal. These derivational categories correspond to the *forms* of Arabic and the *binyanim* of Hebrew. As in the other languages, each has a rough meaning that may or may not hold for particular roots. For example, the “passive-reflexive” pattern denotes the passive for a root like *s.b.r* “break” but the transitive active for a root like *r.k.b* “receive”.

Each combination of TAM and derivational category defines a particular pattern template that combines with roots to form a verb stem. For example, the pattern *C1:aC2:əC3* represents the imperfect TAM and passive-reflexive derivational category (the C’s indicate the consonant slots, and “:” indicates gemination). In combination with the root *s.b.r*, this pattern yields the stem *ssəbbər* “is broken”.

Nouns and adjectives are formed from a verb root through the application of other patterns. For example, the pattern *aC1:əC2aC2əC3* forms a manner noun, *assəbabər* “way of breaking”, and the pattern *məC1C2əC3* forms the infinitive, *məsərər* “to break”.

Most of the members of the composite verb lexeme class [3], discussed above, are also derived from verb roots through root-pattern morphology. For simple three-consonant roots, the main patterns are *C1iC2:iC3:*, *C1əC2əC3:, and C1iC2iC3C2iC3:*. Thus from the root *s.b.r*, we have *sibbirr*, *səbərr*, and *sibbirrīt*. The meanings of these are described as “intensive”, “attenuated”, and “distributive” respectively [3].

As in other Semitic languages, further complexity results from the possibility of four- and five-consonant roots and from roots which may behave as though they consist of one or two consonants because of the special treatment of certain consonants and because of historical changes. For example, the root meaning “kiss” was originally *s.ʃ.m*, but the pharyngeal consonants have been lost in the modern language, resulting in a defective class with only two explicit consonants in each root. For this class, the imperfect passive-reflexive pattern is *C1:aC2*, for example, *ssam* “is/are kissed”.

Within the three-consonant roots, Ethiopian Semitic languages have three lexical subclasses, traditionally referred to as the A, B, and C classes, which correspond to different derivational patterns in other Semitic languages. The B class is characterized by the gemination of the second root consonant in nearly all patterns, the C class by the presence of the vowel a between the first and second consonant in

all patterns. Membership of a root in one or another of these classes is lexical. For example, the A root *t'.b.q* means “be tight”, while the B root *t'.b.:q* means “wait”. In all, there are roughly 25 classes of roots, including subclasses of the A, B, and C classes and multiple classes for four- and five-consonant roots, each with its own set of patterns.

## Other Derivational Processes

Other derivational processes are based on suffixation. For example, many adjectives are formed from nouns through the addition of the suffixes *-mma* and *-awi*: *wiha* “water” ⇒ *wihamma* “watery”, *abiyot* “revolution” ⇒ *abiyotawi* “revolutionary”.

### 1.2.3 Inflectional Morphology

Inflectional morphology consists mostly of suffixes, but sometimes of prefixes or circumfixes, and sometimes of pattern changes.

## Verbs

As noted in Sect. 1.2.2, verbs inflect for TAM through the various patterns that also convey derivational categories such as passive and reciprocal. The four TAM categories are traditionally referred to as perfect(ive), imperfect(ive), jussive-imperative, and gerund(ive). The perfect and imperfect forms, alone or in combination with various auxiliary verbs, function as indicative main clause verbs and, in combination with various prefixes, as subordinate clause verbs as well. The jussive corresponds to one sort of subjunctive in other languages: *yimtu* “let them come”. The gerund, also called converb or continuative, is a subordinate form used in combination with a main clause verb (see Sect. 1.2.4).

Verbs in all four TAM categories also inflect for subject person, number, and (in second and third person singular) gender.<sup>6</sup> Each of the TAM categories has its own set of suffixes and/or prefixes for person–number–gender. The affixes for imperfect and jussive-imperative, a combination of prefixes and suffixes, are almost identical. For perfect and gerund, the inflections are all suffixes. As an example, for the simple derivational category, the third person plural forms for the root *s.b.r* are as follows (the subject agreement affixes are in bold): perfect *səbbəru*, imperfect *yisəbru*, jussive *yisbəru*, gerund *səbrew*.

---

<sup>6</sup>In some other Ethiopian Semitic languages, such as Tigrinya, a distinction is made between masculine and feminine in the second and third person plural as well.

Verbs in all four TAM categories can also take object suffixes. There is a set of direct object suffixes, and there are two sets of indirect object suffixes, distinguished by the two prepositional suffixes *-bb-* and *-ll-* that they begin with. Examples: *səbbərəw* “he broke it”, *səbbərəllat* “he broke (sth.) for her”. Unlike in some other Semitic languages, only one object suffix is possible on a given verb. For more on the function of these suffixes, see Sect. 1.2.4 below.

Verbs in the imperfect, perfect, and jussive-imperative (but not gerund) may also take the negative circumfix (main clause indicative verbs) or prefix (subordinate or jussive-imperative verbs): *alsəbbərəm* “he didn’t break”, *attisbər* “don’t break (sing.masc.)!” Imperfect or perfect verbs may also take the relative prefix *yə(mmi)*, used for verbs in relative clauses and some other subordinate clause types (Sect. 1.2.4): *yəsəbbərəw* “(he) who broke it”, *yəmmitsəbriw* “which you (sing.fem.) break”. Finally, a small set of prepositional and conjunctive prefixes and conjunctive/adverbial suffixes is possible on imperfect and perfect verbs: *indatsebrut* “so that you (pl.) don’t break it” *bisəbrewm* “even if he breaks it”.

Counting the verb stem as two morphemes (root and pattern), an Amharic verb may consist of as many as ten morphemes: *ləmmayanəbbullinim* “also for they who do not read to us” *lə-mm-a-y-{n.b.b+aC1aC2C3}-u-ll-n-m*.

## Nominals

Because of their similar morphosyntax, nouns, pronouns, adjectives, and numerals are grouped together in this section.

Nouns inflect for number, case, and definiteness. There are two number categories, singular and plural. Plurals are usually formed by the suffix *-očč*; broken plurals are rare and in all cases seem to be loanwords from Ge’ez. The plural suffix is not obligatory when plurality is clear from the context: *sost astəmari/sost astəmariwočč* “three teachers”. When it is used, the plural suffix indicates specificity as well as plurality [52].

The definite article takes the form of a suffix; it has the masculine form *-u* and the feminine form *-wa* or *-itu*. The definite suffix follows the plural suffix if there is one. Amharic has no indefinite article.

Amharic subjects are unmarked. Definite, and some indefinite, direct objects take the accusative suffix *-n*. Oblique cases are signaled by prefixed prepositions, sometimes in combination with postpositions. Example: *məskot* “window”, *dingay* “stone”, *yohannis məskotun bədingay səbbərəw* “Yohannis broke the window with a stone”. Amharic nouns have no genitive form; instead genitive and possession are indicated by the prepositional prefix *yə*: *yəbetu wələl* “the floor of the house”.

Noun gender can best be described as flexible [56]. While there are strong tendencies, for example, vehicles, countries, and female animals are normally feminine, it is possible to override these tendencies for pragmatic effect. Even male humans may be referred to with feminine pronouns, feminine demonstratives, and feminine verb agreement affixes, when the speaker wishes to convey affection toward the referent or attribute skillfulness or smallness to the referent.

Nouns may also take one or another of the same set of conjunctive/adverbial suffixes as verbs: *betum* “also the house”.

Demonstrative pronouns and adjectives have singular masculine, singular feminine, and plural forms: *ya bet* “that house”, *yačči kətəma* “that city”. Other adjectives do not inflect for gender, but they may have plural forms, especially in headless noun phrases. Adjective plurals, unlike nouns, are often indicated by reduplication, sometimes in combination with the plural suffix -očč: *tilliq* “big”, *tilliq betočč* “big houses”. It is also possible to use plural adjectives with the singular form of nouns: *addis* “new”, *nəgər* “thing”, *adaddis nəgər* “new things”.

Adjectives and numerals may also take the accusative suffix, the definite suffix, and prepositional prefixes: *kətilliqu* “from the big (one)”.

As in other Semitic languages, possessive adjectives take the form of suffixes on the modified noun. These follow the plural suffix and precede the accusative suffix. Examples: *bete* “my house”, *betoččaččin* “our houses”, *betoččaččew* “their houses”.

### 1.2.4 Basic Syntactic Structure

#### Noun Phrases

An Amharic noun phrase has explicit number, case, and definiteness. The accusative suffix appears obligatorily on definite direct objects and optionally on indefinite direct objects. An unusual feature of the language is the placement of the morphemes marking case (either the accusative suffix or one or another prepositional prefix) and definiteness [37, 55]. These are affixed to the noun itself only when it has no modifiers. If the noun has adjective or relative clause modifiers, the morphemes are normally affixed to the first of these. Examples: *betun* “the house (acc.)”, *tilliqun bet* “the big house (acc.)”, *yəgəzzawn tilliq bet* “the big house (acc.) that he bought”,

Headless noun phrases are common. These consist of one or more relative clauses and adjectives. Examples: *tilliqun* “the big one (acc.)”, *yəgəzzawn* “the one (acc.) that he bought”,

#### Clauses

Unlike in other Semitic languages, all Amharic clauses are headed by verbs. The copula, *nəw*, is a defective verb with only main clause present forms. Its past is filled by the defective verb *nəbbər*, which also serves as the past of the defective verb of existence *allə*. In other cases, the copula is replaced by the perfect, imperfect, jussive-imperative, or gerund of either the verb *norə* “live” or the verb *honə* “become”.

The basic word order of all Ethiopian Semitic languages is subject-object-verb (SOV), a feature that probably results from contact with Cushitic languages. As is common in SOV languages, the order of subject, object, and oblique arguments of the verb is somewhat flexible. In particular, for pragmatic reasons the subject can follow another argument: *yohannis məskotun səbbərəw*, *məskotun yohannis səbbərəw*, “Yohannis broke the window”.

As in other Semitic languages, verbs agree with their subjects in person, number, and (in second and third person singular) gender.<sup>7</sup> Verbs also agree with definite direct or indirect objects, but not both. As discussed in Sect. 1.2.3, there are three sets of object agreement suffixes. This three-way split within the verbal morphology system maps in complex ways onto explicit syntactic arguments of the verb [45]. That is, direct object verb suffixes correspond usually but not always to accusative syntactic arguments, and indirect object verb suffixes correspond usually but not always to explicit prepositional phrases (noun phrases with prepositional prefixes). The *-ll-* indirect object suffix agrees with dative and benefactive arguments, while the *-bb-* suffix agrees with locative, instrumental, and adversative arguments. Examples: *Iəhirut sət'tat* “he gave (it) to Hirut” (direct object suffix (3 pers.sing.fem.) agrees with explicit syntactic dative), *Iəhirut sərrallat* “he made (it) for Hirut” (indirect object suffix agrees with explicit syntactic benefactive).

Impersonal verbs [56] complicate the picture further. The morphological subject of these verbs is always third person singular masculine, and they take an obligatory direct object agreement suffix that refers to the experiencer: *hirut rabat* “Hirut is hungry”. The verb of possession behaves similarly; it makes use of the verb of existence *allə* with direct object suffixes referring to the possessor. The morphological subject of the verb is the possessed object or objects. *hirut sost wəndimmočč alluwat* “Hirut has a brother” (subject agreement 3 pers.plur., object agreement 3 pers.sing.fem.)

As in other Semitic languages, pronoun subjects and pronoun objects are omitted unless they are emphasized. This fact, in combination with the elaborate derivational and inflectional verb morphology, means that sentences consisting of a verb alone or a main verb and an auxiliary verb are not uncommon: *alt'əyyəqnatim* “we didn’t visit her”, *laflallačihu* “shall I boil (sth.) for you (pl.)?”, *awwaddədun* “they made us like each other”.

Main clause verbs are either in the perfect or a compound imperfect formed from the simple imperfect plus conjugated suffix forms of the defective verb of existence *allə*. Subordinate clause verbs are in the perfect, simple imperfect, or gerund. *tifəlligiyalləš* “you (fem.sing.) want”, *bittifəlligi* “if you (fem.sing.) want”.

Cleft constructions are very common in Amharic [51]. Indeed for questions, cleft constructions are probably more common than non-cleft constructions. In a cleft sentence, the focused argument is placed first, followed by the conjugated copula,

---

<sup>7</sup>Some other Ethiopian Semitic languages distinguish masculine and feminine in the second and third person plural as well.

followed by other arguments of the original verb, followed by the verb in relativized form: *mindin nəw yəsəbbərəw* “what did he break?” lit. “what is it that he broke it”.

Relative clauses make use of the relative imperfect or perfect form of the verb: *yəsəbbərəw* “that he broke”, *yəmmisəbrəw* “that he breaks”.

Adverbial clauses are usually indicated through the use of prefix conjunctions on the relative form of the verb (in which case the initial *yə* is dropped) or the bare imperfect: *siləmmisəbrəw* “because he breaks it”, *bisəbrəw* “if he breaks it”.

Nominalizations in Amharic take two forms. Either the verb of the nominalized clause appears in the (conjugated) relative form with the prefix conjunction *ində* “that”, or the verb appears in the (unconjugated) infinitive form, with the verb’s original subject in the form of a possessive suffix. Examples: *min indəmmigəza alawqim* “I don’t know what he will buy”; *zigiju məhonaččəwn gəlləs’u* “they explained that they were ready”, lit. “they explained **their being (acc.)** ready”.

As is common in SOV languages, Amharic permits the chaining of a number of clauses together in a single sentence without explicit conjunctions indicating the relationship between the clauses. The usual interpretation is sequentiality. All verbs but the final one appear in the gerund form. The final verb may be perfect, compound imperfect, jussive, or imperative. All of the gerund forms agree with the subject of the final verb. Example: *iþet təməlliso rat bəlto təjjia* “He returned home, ate dinner, and went to bed” lit. “Returning (3 pers.sing.masc.) home, eating (3 pers.sing.masc.) dinner, he went to bed”.

## 1.3 Arabic

The term “Arabic language” is often used to refer collectively to Modern Standard Arabic (MSA) and its dialects. MSA is the official language of the Arab World, a region of 22 countries. Arabic is also an official language of Chad, Eritrea and Israel. With a collective population of almost 300 million people, Arabic is the most commonly spoken Semitic language.

While MSA is the primary written language of the media and education, the dialects, which are historically related to MSA, are the truly native informal spoken media of communication for daily life. The dialects, which vary geographically and socially, are not taught in schools or standardized. They are the primary linguistic form used to express modern Arab culture, songs, movies and TV shows. Although the language-dialect situation of Arabic seems similar to many across the world, two aspects set it apart: (a) the high degree of difference between standard Arabic and its dialects, often compared to Latin and the Romance languages [39]; and (b) the fact that standard Arabic is not any Arab’s native language [41]. The two varieties of Arabic coexist along the lines of formality in what is termed a state of *diglossia* [35]: formal written (MSA) versus informal spoken (dialect); there is a large mixing area in between [5, 6]. For more information on MSA, see Holes [47].

### 1.3.1 Orthography

## Arabic Script

Arabic is written using the Arabic script, a right-to-left alphabet also used to write many languages around the world which are not related to Arabic such as Persian, Kurdish, Urdu and Pashto. Arabic dialects are by default written in Arabic script although there are no standard dialectal spelling systems [43].

<sup>8</sup>In IPA and Maltese, the symbol  $\hbar$  is used for *voiceless pharyngeal fricative* or ( $\text{χ}$  H). In HSB, it reflects the two letters that compose the Ta-Marbuta symbol ( $\text{ð} \text{ħ}$ ):  $\text{ð}$  h and  $\text{ħ}$  t.

## Arabic Spelling

Arabic spelling rules utilize its script as an impure *abjad*: although diacritics are omitted most of the time, long vowels are always written in Arabic using a combination of an omittable diacritic short vowel and non-omittable compatible consonant, e.g., [u:] is written as *uw*. Additionally, Arabic uses two morphophonemic letters: ة ه is the Ta-Marbuta, feminine ending morpheme, and ي is the Alif-Maqṣura, derivational marker of the word's root radical ي y realizing as the vowel [a]. Some letters in Arabic are often spelled inconsistently which leads to an increase in both sparsity (multiple forms of the same word) and ambiguity (same form corresponding to multiple words), e.g., variants of Hamzated Alif, أْ or ء، are often written without their Hamza (ء): أ A; and the Alif-Maqṣura (or dotless Ya) ي y and the regular dotted Ya ي y are often used interchangeably in word final position [22].

### 1.3.2 Morphology

In discussing Arabic morphology, it is useful to distinguish between operations related to the type or form of the morphological process as opposed to its function [40, 68].

In terms of morphological form, Arabic, in a typical Semitic manner, uses both *templatic* and *concatenative* morphemes. Concatenative morphemes form words through sequential concatenation, whereas templatic morphemes are interdigitated. Functionally, like most other languages of the world, we can distinguish between derivational morphology and inflectional morphology. In Arabic morphological form and function are independent although most templatic processes are derivational and most concatenative processes are inflectional. There are some important exceptions which we allude to below.

### Templatic Morphology

There are two types of templatic morphemes: roots and templates.<sup>9</sup> The *root* morpheme is a sequence of mostly three consonantal radicals which together signify some abstract meaning. For example, the words كتب katab “to write”, كاتب kAtrib “writer/author/scribe”, مكتوب maktub “written/letter”, مكتب maktab “office” and مكتبة maktabah “library” all share the root morpheme *k.t.b* “writing-related”. Root semantics is often, however, idiosyncratic. For example, the semantically

---

<sup>9</sup>Templates are sometimes further split into patterns and vocalisms [59].

divergent words لحم *laHm* “meat” and سolder *laHam* “to solder” also have the same root *I.H.m* [41]. The template is an abstract pattern in which roots are inserted. The template like the root provides some meaning component. For example, the template *maC1C2aC3*, often indicating *location*, combines with the roots ك.ت.ب. *k.t.b* “writing-related” and ع.م.ل. *m.l* “work-related” to create the nouns مكتب *maktab* “office” and معمل *maṣ mal* “laboratory/factory”.

## Concatenative Morphology

The templatic morphemes form the stem onto which concatenative affixes (prefixes, suffixes and circumfixes) and clitics (proclitics and enclitics) attach. Clitics are morphemes that have the syntactic characteristics of words, but show evidence of being phonologically bound to another word [58], e.g., the proclitic conjunction +و *w+* “and” or the object pronoun enclitic هم+ *+hm* “them”. In this respect, a clitic is distinctly different from an affix, which is phonologically and syntactically part of the word and often represents inflectional features, e.g., the suffixes ة+ *+h* and اات+ *+At* represent the feminine singular and feminine plural inflections, respectively. A word may include multiple affixes and clitics, e.g., the word وسيكتوبونها *wasayaktubuw-nahA* has two proclitics, one circumfix and one enclitic: *wa+sa+y+aktub+uwna+hA* (and + will + 3person + write + masculine – plural + it) “and they will write it”.

The combination of both templatic and concatenative morphemes may involve a number of phonological, morphological and orthographic adjustments that modify the form of the created word beyond simple interleaving and concatenation. For example, the feminine singular morpheme, ة+ *+h*, is turned into ت+ *+t* when followed by a possessive clitic: أميرة+هم *Āamiyrahū+hum* “princess+their” is realized as أميرتهم *Āamiyratuhum* “their princess”.

## Derivational Morphology

Derivational morphology creates new words from other words typically through a template switch and often resulting in a change in part-of-speech (POS). The root remains constant in the process. For example, the Arabic كاتب *kAtib* (*k.t.b+C1AC2iC3*) “writer” can be seen as derived from the verb كتب *katab* (*k.t.b+C1aC2aC3*) “to write” in the same way the English writer can be seen as a derivation from write. Although compositional aspects of derivations do exist, the derived meaning is often idiosyncratic. For example, the masculine noun مكتب *maktab* “office/bureau/agency” and the feminine noun مكتبة *maktabah* “library/bookstore” are derived from the root ك.ت.ب. *k.t.b* “writing-related” with the location template *maC1C2aC3* [41].

## Inflectional Morphology

In inflectional morphology, the core meaning and POS of the word remain intact and the extensions are always predictable and limited to a set of possible features. Inflectional features are all obligatory and must have a specific (non-nil) value for every word. In Arabic, there are eight inflectional features. *Aspect* (perfective, imperfective, imperative), *mood* (indicative, subjunctive, jussive), *person* (1st, 2nd, 3rd) and *voice* (active, passive) only apply to verbs, while *case* (nominative, accusative, genitive) and *state* (definite, indefinite, construct) only apply to nouns/adjectives. *Gender* (feminine, masculine) and *number* (singular, dual, plural) apply to both verbs and nouns/adjectives.

Clitics are similar to inflectional features in that they do not change the core meaning of the word; however, they are all optional. Arabic clitics include conjunctions, verb particles, nominal prepositions, the definite article and pronominal enclitics that can serve as possessives of nouns and objects of verbs and prepositions.

## Form-Function Independence

Arabic morphological form and function are independent. Both templatic and concatenative morphemes can function derivationally or inflectionally, with the exception of the roots, which are always derivational. The majority of derivational morphology is templatic and the majority of inflectional morphology is concatenative. The most important exception is the templatic plural, often called “broken plural”, which is formed through change of templates as opposed to the sound plurals formed through affixation. For example, compare the following two plurals of the noun كاتب *kAtib* “writer”: كتاب *kut~Ab* (broken) “writers” and كتابات *kAtib+At* (sound) “writers [fem]”. More than half of all plurals in Arabic are broken [2]. Another example of form-function independence is the derivational suffix *Ya of Nisba* يـ + *iy~*, which maps nouns to adjectives related to them, e.g., كتبـي *kutub+iy~* “book-related” is derived from the noun كتب *kutub* “books”, a broken plural itself of the noun كتاب *kitAb* “book”. For other aspects of form-function independence in Arabic, see Habash [41].

## Dialectal Arabic Morphology

Arabic dialect morphology is simpler in some respects and more complex in others compared to MSA. Dialects overall lost the case and mood inflections and merged feminine and masculine plurals and duals in verbs among other changes. However, Arabic dialects introduce additional clitics, some with new functionality. For example, the non-MSA circum-clitic شـ + ما mA + + š which is used to mark

negation appears in several dialects. Another example of change is the MSA future proclitic  $+سَ$  *sa+*, which is replaced by  $+حَ$  *Ha+* in Levantine and  $+يَ$  *ya+* in Moroccan. For more information on Arabic dialects, see Cowell [17], Erwin [23], Abdel-Massih et al. [1], and Harrell [44].

## Morphological Ambiguity

Arabic's optional diacritics, inconsistent spelling of some letters, together with the language's complex morphology lead to a high degree of ambiguity: the Buckwalter Arabic Morphological Analyzer (BAMA), for instance, produces an average of 12 analyses per word [15] corresponding to almost 7 diacritizations and almost 3 lemmas.

### 1.3.3 Basic Syntactic Structure

#### Morphology and Syntax

In morphologically rich languages, such as Arabic, many syntactic phenomena are expressed not only in terms of word order but also morphology. For example, Arabic subjects of verbs have a nominative case and adjectival modifiers of nouns generally agree with the noun they modify in case, gender, number and definiteness.

#### Sentence Structure

Arabic has two types of sentences: verbal sentences (VS) and nominal (sometimes called copular, or equational) sentences (NS). The prototypical VS form is Verb-Subject-Object(s). Arabic is a pro-drop language: in the case of a pronominal subject, the verb expresses the person, gender and number of the subject. However, with non-pronominal subjects, the verb agrees in person (3rd) and gender only, while number defaults to singular. This is referred to as partial agreement.

The prototypical NS has the form of Subject-Predicate or Topic-Complement. The subject is typically a *definite* nominative noun, proper noun or pronoun and the predicate is an *indefinite* nominative noun, proper noun or adjective that agrees with the subject in number and gender. The predicate can be a prepositional phrase (PP), in which case no agreement is shown.

A complex sentence structure is formed from an NS with a VS predicate producing a Subject-Verb-Object order. Here, the subject and verb agree in full (gender, number and person).

## Nominal Phrase Structure

Arabic adjectives follow the nouns they modify in a nominal phrase (NP). Adjectives and nouns agree in gender, number, definiteness and case, with the exception of irrational (non-human) plural nouns whose adjectives are feminine singular.

Arabic also has a possessive/genitive construction, called *Idafa*, which relates two nouns: the first is the possessed and the second is the possessor. The first noun is in the construct state, and the second takes a genitive case and is typically definite. An *Idafa* chain can be formed by adding an additional element to the beginning of the NP. For example, مفاتيح سيارة الرجل *mfAtyH syArħ Alrjl* (keys-car-the+man) translates as “The man’s car keys”. Adjectives modifying the head of an *Idafa* construction agree with it in case; but they agree with its dependent in definiteness.

## Relative Clauses

Relative clauses modify the noun they follow. If the modified noun is definite, the relative clause is introduced by a relative pronoun which agrees with the noun it modifies in gender and number (irrationality gets exceptional agreement). Relative clauses of indefinite nouns are not introduced with a relative pronoun.

## Arabic Dialect Syntax

Arabic dialects are not very different syntactically from MSA. For example, both SVO and VSO orders exist in the dialects although the SVO order is more dominant. Some of MSA’s complex syntactic phenomena such as irrational plural agreement are maintained in the dialects, while case agreement is gone since dialects do not inflect for case. For more information on dialectal Arabic syntax, see Brustad [13].

## 1.4 Hebrew

Hebrew is one of the two official languages of the State of Israel,<sup>10</sup> spoken natively by half of the population and fluently by virtually all the (over seven million) residents of the country. Hebrew exhibits clear Semitic behavior<sup>11</sup>; in particular, its lexicon, word formation and inflectional morphology are predominantly Semitic.<sup>12</sup>

<sup>10</sup>The other is Arabic.

<sup>11</sup>In spite of some recent claims to the contrary [48, 75].

<sup>12</sup>Parts of the discussion in this section are based on Itai and Wintner [49].

Hebrew is unique among the Semitic languages (indeed, among the world’s languages) in that it has been ‘dormant’ for several centuries, used primarily for religious and academic purposes, and little, if at all, as a daily spoken language. Modern Hebrew was ‘revived’ in the end of the nineteenth century; while it is undoubtedly based on the infrastructure of older layers of Hebrew, it was heavily influenced by Yiddish, Slavic languages, and other languages spoken by Jews during nearly two millennia. In this chapter, the term ‘Hebrew’ refers to Modern Hebrew only.

### 1.4.1 Orthography

Hebrew is written in the Hebrew alphabet, a 22-letter *abjad* [18, 65]. To facilitate readability we use a straightforward transliteration of Hebrew in this chapter, where the characters אַבְגָּדְהַזְּחָטִיכְלָמְנָסְעַפְצָרְשָׁת (in Hebrew alphabetic order) are transliterated thus: *abgdhwzxTiklmnsupcqršt*. There are two main standards for the Hebrew script: one in which vocalization diacritics, known as *niqqud* “dots”, decorate the words, and one in which the dots are missing, and other characters represent some, but not all of the vowels.<sup>13</sup> Most of the texts in Hebrew are of the latter kind. While the Academy for the Hebrew Language publishes guidelines for transcribing undotted texts [36], they are only partially observed. Thus, the same word can be written in more than one way, sometimes even within the same document. For example, *chriim* “noon” can be spelled *chrim*.

The script dictates that many particles, including four of the most frequent prepositions, the definite article *h* “the”, the coordinating conjunction *w* “and” and some subordinating conjunctions, all attach to the words that immediately follow them. When the definite article *h* is prefixed by one of the prepositions *b* “in”, *k* “as”, or *l* “to”, it is assimilated with the preposition and the resulting form becomes ambiguous as to whether or not it is definite. For example, *bth* can be read either as *b+th* “in tea” or as *b+h+th* “in the tea”. Consequently, the form *šbth* can be read as an inflected stem (the verb “capture”, third person singular feminine past), as *š+bth* “that+field”, *š+b+th* “that+in+tea”, *š+b+h+th* “that in the tea”, *šbt+h* “her sitting” or even as *š+bt+h* “that her daughter”. See Table 1.1.

These features of the writing system imply that Hebrew texts tend to be highly ambiguous. First, the first and last few characters of each token may be either part of the stem or bound morphemes (prefixes or suffixes). Second, the lack of explicitly marked vowels yields many homographs. See a detailed discussion in Sect. 1.4.4.

---

<sup>13</sup>The undotted script is sometimes referred to as *ktiv male* “full script”, whereas the dotted script, but with the diacritics removed, is called *ktiv xaser* “lacking script”. These terms are misleading, as any representation that does not depict the diacritics would lack many of the vowels.

Table 1.1 The various morphological analyses of the form *šbt*

Lemma	POS	Number	Gender	Person	Tense	State	Def	Prefix	Suffix	Gloss
<i>šbt</i>	Noun	Sing	Fem	n/a	n/a	abs	No		h	"her Saturday"
<i>bt</i>	Noun	Sing	Fem	n/a	n/a	abs	No		h	"that her daughter"
<i>bth</i>	Noun	Sing	Fem	n/a	n/a	abs	No		š	"that a field"
<i>th</i>	Noun	Sing	Masc	n/a	n/a	abs	Yes	š+b+h		"that in the tea"
<i>th</i>	Noun	Sing	Masc	n/a	n/a	abs	No	š+b		"that in tea"
<i>th</i>	Noun	Sing	Masc	n/a	n/a	cons	No	š+b		"that in tea-of"
<i>šbh</i>	Verb	Sing	Fem	3	Past	n/a	n/a			"(she) captured"
<i>šbt</i>	Verb	Sing	Fem	3	Past	n/a	n/a			"(she) went on strike"

### 1.4.2 Derivational Morphology

#### Root and Pattern Processes

Hebrew morphology is rich and complex. The major word formation machinery is root-and-pattern. As an example of root-and-pattern morphology, consider the root *k.t.b*, which denotes a notion of writing. Hebrew has seven verbal patterns, which contribute to the meaning of the stem in productive but not fully predictable ways. Thus, construed in the *pa'el* pattern *CaCaC* (where the 'C's indicate the slots), the root *k.t.b* yields *ktb* [*katav*] “write”; whereas in the *hif'il* pattern *hiCCiC*, which is typically a causative, it yields *hktib* [*hiktiv*] “dictate”. Similarly, the (nominal) pattern *haCCaCa* usually denotes nominalization; hence *hktbh* [*haktava*] “dictation”. The pattern *maCCeCa* often denotes instruments; construed in this pattern, the root *k.t.b* yields *mktbh* [*makteva*] “writing desk”.

After the root combines with the pattern, some morpho-phonological alternations take place, which may be non-trivial: for example, the *hitCaCCut* pattern triggers assimilation when the first consonant of the root is *t* or *d*: thus, *d.r.š+hitCaCCut* yields [*hidaršut*]. The same pattern triggers metathesis when the first radical is *s* or *š*: *s.d.r+hitCaCCut* yields [*histadrut*] rather than the expected \*[*hitsadrut*]. Semivowels such as *w* or *y* in the root are frequently combined with the vowels of the pattern, so that *q.w.m+haCCaCa* yields [*haqama*], etc. Frequently, root consonants such as *w* or *y* are altogether missing from the resulting form.

#### Other Derivational Processes

While root-and-pattern is the main word formation process in Hebrew, other processes are also operational [63]. These include several regular pattern-relating processes, such as nominalization: each verbal pattern in Hebrew is related to a pattern whose meaning is typically the nominalization of the meaning of the corresponding verb. For example, *hitCaCeC* is related to *hitCaCaCut*, so that from *htlhb* [*hitlahev*] “be enthusiastic about” one obtains *htlhbw* [*hitlahavut*] “enthusiasm”. Similarly, *CiCeC* is related to *CiCuC*, so that from *šipr* [*šiper*] “improve” one obtains *šipwr* [*šipur*] “improvement”.

Other processes are more concatenative in nature, based primarily on suffixation. Thus, the suffix *wt* [*ut*] is frequently used to convert adjectives to nouns, as in *bria* [*bari*] *healthy* ⇒ *briawt* [*bri'ut*] “health”. The suffix *wn* [*on*] can be used to construct the diminutive, as in *db* [*dov*] “bear” ⇒ *dwbwn* [*dubon*] “teddy bear”. Interestingly, Hebrew also has a non-concatenative diminutive operator that is based on reduplication: *klb* “dog” ⇒ *klblb* “doggie”; *xtwl* “cat” ⇒ *xtltwl* “kittie”; *irwq* “green” ⇒ *irqrq* “light green”; etc.

### 1.4.3 Inflectional Morphology

Inflectional morphology is highly productive and consists mostly of suffixes, but sometimes of prefixes or circumfixes, and sometimes of pattern changes.

#### Verbs

Verbs inflect for number, gender and person (first, second and third) and also for a combination of tense and aspect/mood, referred to simply as ‘tense’ below. Some of these variations are obtained by simple concatenation rules, whereas others are better explained in terms of stem changes. The citation form of Hebrew verbs is the third person, masculine, singular past form. Consider *sgr* [*sagar*] “close”, for example. Its past tense forms include *sgrti* [*sagarti*], *sgrt* [*sagarta*], *sgrnw* [*sagarnu*], etc., all of which can be obtained from the citation form by concatenation. Other past tense forms, such as *sgrh* [*sagra*] or *sgrw* [*sagru*], can still be explained in terms of concatenation and simple morpheme-boundary alternations (in this case, reduction of the last vowel of the stem). However, in the present some inflected forms require a change in the stem, as in *swgr* [*soger*], *swgrim* [*sogrim*], etc. The same applies to the future, some of whose forms are *tsgwr* [*tisgor*], *isgrw* [*yisgeru*], etc. Consequently, a good way of accounting for entire verb paradigms is by specifying, for each verb, not only the citation form but also secondary stems for some of the inflections.

Fortunately, this is not difficult because the secondary stems are determined by the root and the pattern of the verb. As noted above, verbs can belong to one of seven patterns, and each pattern is inflected in exactly the same way. Root consonants may trigger some variations, but these, too, are systematic and regular.

Verb patterns (‘*binyanim*’) are associated with (vague) meanings. Traditionally, the *nif’al* pattern is considered to be the passive of the *pa’al* pattern, whereas *pi’el* denotes reinforcement or intensification, and *hif’il* denotes causativization. However, these correspondences are not always clear [63]. They are highly productive in two cases: *pu’al* is almost with no exception the passive of *pi’el*, and *huf’al* is the passive of *hif’il*.

Verbs can take pronominal suffixes, which are interpreted as direct objects. Such processes, however, become less frequent in contemporary Hebrew, and are usually associated with archaic language or a high register. In some cases, verbs can also take nominative pronominal suffixes, but this is now limited to participle forms, which may be interpreted as nouns.

Participle forms indeed deserve special attention, as they can be used as present tense verbs, but also as nouns or adjectives. For example, *mTps* [*metapes*] “climbing” can be used as a verb (*hwa mTps yl hqir* “he is climbing on the wall”), a noun (*hmTps hgij al hpsgh* “the climber reached the summit”), or an adjective (*qniti cmx mTps lginh* “I bought a vine [= climbing plant] for the garden”).

## Nominals

Several morphological properties are common to nouns, adjectives, numerals and even prepositions; they are therefore grouped together in this section. Nouns inflect for number (singular, plural and dual); the dual form is not productive and is only preserved on a few nouns, such as those denoting body parts (*id* [yad] “hand” ⇒ *idim* [yada’yim] “hands”). Even in these cases, the dual is semantically plural, except for time expressions (e.g., *šntim* [shnata’yim] “two years”). Nouns that denote animate entities inflect for gender (masculine or feminine), although some exceptions are known [62].

Adjectives inflect for number (only singular and plural) and gender; the feminine suffix is either *h* [a], *t* [et], or *it* [it]. Numerals also inflect for gender, and ordinals also inflect for number; several idiosyncrasies occur, especially with the low numerals.

In addition, all these three types of nominals have two phonologically distinct forms, known as the *absolute* and *construct* states; the latter are used in compounds [9, 10]. For example, *šmlh* [simla] “dress” vs. *šmlt* [simlat], as in *šmlt klh* “bridal gown”; or *qcr* [qacar] “short” vs. *qcr* [qcar] in [*qcar ru’ax*] “impatient [=short tempered]”. In the standard Hebrew orthography approximately half of the nominals appear to have identical forms in both states, a fact which substantially adds to the ambiguity.

The proto-Semitic three-case system, with explicit indication of nominative, accusative and genitive cases, is not preserved in Hebrew. Personal pronouns, however, reflect traces of cases. Distinct pronominal forms exist for the nominative, accusative, dative and genitive. All these pronouns inflect for number, gender, and person. In addition, prepositions inflect for exactly these features, in a way that fully resembles the inflectional paradigm of pronouns (and, to some extent, that of nouns and adjectives), so that the distinction between inflected pronouns and prepositions is blurred. For example, the second person, feminine, singular, dative pronoun *lk* [la $\bar{k}$ ] can be viewed as an inflected form of the preposition *l* “to”, with a cliticized pronominal suffix indicating the number, gender and person.

A related phenomenon allows nouns to take possessive pronominal suffixes that inflect for number, gender and person. The base form for such inflections is the construct state. Thus, *šmlt* can combine with *k* [e $\bar{k}$ ] to yield *šmltk* “your dress”. In sum, then, the morphological characterization of many nominal forms includes the specification of their number and gender, as well as the person, number and gender of potential pronominal suffixes.

## Other Closed-Class Items

The main remaining part-of-speech category is that of adverbs. Generally, adverbs do not inflect, but few exceptions exist (e.g., *laT* “slowly” and *lbd* “alone”, which inflect for person, number and gender). Many adverbs are created from nouns by adding the preposition *b* “in”, e.g., *b+mhirwt* “in+speed=quickly”.

### 1.4.4 Morphological Ambiguity

The deficient orthography, including the lack of vowels and the attachment of frequent particles to the words that follow them, and the morphological complexity outlined above, greatly contribute to the ambiguity of Hebrew word forms. Itai and Wintner [49] report that their morphological analyzer produces 2.64 analyses per word token, on average, on a corpus of newspaper articles, with many tokens associated with more than a dozen analyses. More recent results show an average ambiguity level of 3.4 (excluding punctuation), with some tokens associated with over 20 analyses (Alon Itai, p. c.). These analyses can differ at the level of segmentation; or they can reflect different morphological features (e.g., one can be a construct state noun, while the other is absolute); or, in few extreme cases, they can be identical up to the identity of the stem.

As an example, consider the output produced by the morphological analyzer of Itai and Wintner [49] on the form *šbth*, depicted in Table 1.1 (adapted from Itai and Wintner [49]). Note in particular the last two analyses, which only differ in the lemma.

### 1.4.5 Basic Syntactic Structure

The dominant, unmarked word order in Hebrew is subject–verb–object (SVO), although many variations are possible [8, 38]. In particular, a very common construction (especially in journalistic jargon) is “verb-second”, whereby the verb follows some constituent (typically, an adverbial, but sometimes an object) and precedes the subject and the rest of its complements.

Verbs agree with their subjects in number, gender and person. This facilitates some flexibility in constituent order, even without explicit case marking. When the subject is a pronoun, it may be omitted in certain cases, especially in the past and future tenses.

While clauses may be headed by verbs, this is not mandatory, and “verbless” predicates, whose heads are adjectives, prepositional phrases or nouns, abound [20]. Typically, in such cases the predicate is indefinite but the subject is definite. Clauses can also be headed by modals such as *crik* “it is necessary” or *aswr* “it is forbidden”, typically followed by infinitival verb phrases or by finite clauses introduced by *š* “that”. A unique construction involves the existentials *iš* “there is” and *ain* “there isn’t”, that behave like verbs in some respects, but are highly idiosyncratic.

Yes/no questions are either formed with the explicit interrogative *ham* “is it true that”, or by changing the intonation pattern of the declarative counterpart, with no change in word order. Wh-questions are formed by fronting an interrogative pronoun (e.g., *mi* “who”, *lmh* “why”) but with no auxiliaries and no change in word order.

Within the noun phrase, nouns typically agree with their adjuncts (adjectives, demonstrative pronouns, and numerals) in gender and number, but also on definiteness. Hebrew only has a definite article, which is marked (as a prefix, *h*) on nouns,

adjectives, numerals and demonstrative pronouns, and has to be distributed over most of the components of a definite noun phrase [73].

Hebrew provides three different constructions for specifying genitive (possessive) constructions. First, using the genitive preposition *šl* “of”: *hsprim šl hmšwrr* “the-books of the-poet”. Second, by using the *construct* state of the head noun: *spri hmšwrr* “books-of the-poet”. In such constructions, the definite article is only marked on the complement, and is “inherited” by the head noun [73]. As noted above, when the complement is a pronoun, it is realized as a cliticized suffix on the head noun: *spriw* “books-his”. Finally, a double-genitive construction combines both the pronominal suffix *and* an explicit genitive complement, as in *spriw šl hmšrr* “books-his-of the-poet”.

Relative clauses are introduced by an explicit relativizer *š* or *ašr* “that”. A third relativizer, *h*, is used in relative clauses that begin with a present-tense verb; such constructions can also be viewed as adjectival phrases, as present-tense verbs can be viewed as adjectives, and the relativizer *h* is a homograph of the definite article. Resumptive pronouns in the relative clause must agree with the head noun, but may be omitted in certain cases (and *must* be omitted in others). Resumptive pronouns are obligatory as complements of prepositions and as possessors. When the resumptive pronoun is preceded by a preposition, and opens the relative clause, the relativizer may be omitted.

## 1.5 Maltese

Maltese belongs to the South Arabic branch of Central Semitic [46, 47]. It has an Arabic stratum, a Romance (Sicilian, Italian) superstratum and an English adstratum [12, 30]. The influence of the non-Semitic element is most obvious in the lexis, while the most salient basic grammatical structures are of Arabic origin. Maltese is the native language of approximately 400,000 people who live in Malta and Gozo, but it is also spoken abroad in communities in Australia, Canada, the USA and the UK.

### 1.5.1 Orthography

Maltese is the only Semitic language that uses a Latin-based alphabet. There are 31 letters in the Maltese alphabet. Two of these are digraphs, namely, *ie* /i:/ and *għ*, which is generally silent but can also be pronounced /ħ/ under certain conditions (see below). Three symbols require diacritic marks, namely, *ċ* /č/, *ż* /z/, *ħ* /ħ/ (including *għ*). The digraph *għ* and the unbarred *h* are generally silent, but they are sounded in certain contexts, as, for example, when they occur together as *għħ*, as in *tagħha* “hers” [’teħħe]. Some speakers also sound the unbarred *h* in certain contexts in which the *h* is a part of the object pronominal clitic *-ha* “her” or *hom* “them”, as in *raha* [’reħħe] “he saw her” instead of [rv :]. The latter is considered to be the standard variety, but the former is also very widespread.

Some graphemes are ambiguous in terms of pronunciation. Thus, double z, i.e., zz, can either be geminate [ts:], as in *razza* “race” [‘rə ts:və], or geminate [dz:], as in *gazzetta* “newspaper” [gə dz:’zɛtə]. Similarly, x can be [ʃ], as in *rixa* “feather” [’ri:ʃə], or [ʒ], as in *televixin* “television” [tʒlʒ’vɪʒɪn]. The grapheme i also has the values [ɪ] and [i:], as in *fitt* “nuisance” [fɪt] and [fti:t] *fitt* “a little”, but it can also be pronounced as [ɪ:] in certain contexts, e.g., when followed by the sound corresponding to graphemic *q*, *gh*, *h* or *ħ*, as in *riħ* “wind” [ri:ħ]. The letters generally retain their sound values but there are some surface phonetic/phonological rules that bring about changes in pronunciation that are not reflected in the spelling. For example, the rule of word final obstruent devoicing is not reflected in the orthography, which retains the underlying root sound. For example, [br:p] is rendered orthographically as *bieb* “door”, not \**biep*, because of an alternation with [’br:ba] *bieba* “a door leaf”.

### 1.5.2 Derivational Morphology

#### Mixed Root-Based and Stem-Based Morphology

At different phases of its history, Maltese borrowed profusely both from Romance (Sicilian, Tuscan, and Modern Italian), as well as from English, especially in recent times (see, in particular, Brincat [12] for a historical perspective; for a recent descriptive grammar of Maltese, see Borg and Azzopardi-Alexander [11]). As a result, Maltese displays a great deal of mixture, especially at the lexical level. While some of these borrowings have been integrated into largely Semitic/Arabic morphological and syntactic patterns, others have, in turn, had innovative effects on all levels, namely, phonological, grammatical (morpho-syntactic) and semantic. The result is that Maltese morphology appears to be both root-based and stem-based, at least on a surface analysis (see Fabri [28]; Twist [70]; Ussishkin and Twist [71] for various analyses and discussion).

The Arabic vocabulary of Maltese still retains the root-and-pattern characteristic typical of Semitic languages, whereby derivational and inflectional word forms are characterized by both internal changes to the basic consonant and vowel structure, i.e., non-concatenatively, as well as through affixation, i.e., concatenatively. For example, in derivation, a number of forms are related to the trilateral (tri-consonantal) radical *q.s.m*, including *qasam* “split” (1st verbal form: CVCVC), *qassam* “share out” (2nd verbal form: CVCiCiVC), *tqassam* “get shared out” (5th verbal form: t+ CVCiCiVC), *nqasam* “broke” (7th verbal form: n+CVCVC), *qasma* “a split” (feminine singular nominal form: CVCC-a). In contrast to words of Semitic/Arabic origin, non-Semitic borrowings, especially recent ones, often retain their stem-based, purely concatenative properties. For example, the suffix -ata, borrowed from Italian, can combine with stems of Arabic origin to form new lexemes, as *żiblata* “a booze up”, which is made up of *żibel* “thrash” and -ata.

In one period of its history, borrowings, especially from Sicilian and, later, Tuscan, were often integrated into the Semitic root-and-pattern system. To take an example, the word *pejjep* “to smoke” is of Romance origin from *pipa* “pipe”, but was turned into a weak verb (with a middle weak consonant *j*) and made to undergo gemination of the second consonant to be turned into a verb of the second form, which generally characterizes the causative form of a basic verbal (form one) or nominal form. The productivity of these forms in Modern Maltese is a matter of discussion and research. There is some evidence that these forms might not be actively productive anymore.

One important way in which Maltese differs from Arabic and other Semitic languages is in the morphemic status of the vowel melody. While in Modern Standard Arabic (MSA), the vowel melody itself carries morphological information, e.g., *a* for perfective active and *u-i* for perfective passive, this is not the case in Maltese. In Maltese, the vowel melody either stays the same throughout, or changes for phonological reasons or unpredictably. Thus, e.g., the imperfective active and passive, and the perfective active and passive for a verb like *kiser* “break” are *jikser* “he breaks”, *jinkiser* “he is broken”, *kiser* “he broke”, *nkiser* “he was broken” with the melody *i-e* throughout. In other words, Maltese has lost the typical morphologically motivated vowel ‘insertion’; instead, it can be argued that vowel ‘insertion’ is purely phonological, allowing root syllabification [28, 64].

In Modern Maltese, new verbs are generally ‘derived’ through loan words which take on a very specific verbal form associated with so-called ‘lacking’ verbs (*verbi neqsin*), i.e., verbs that traditionally are considered to have a missing final weak consonant *j* in their citation form, e.g., the verb *tefa* “extinguish” (see, in particular, Mifsud 1995 on loan verbs). Thus, an English loan verb like *to monitor* becomes *immoniterja* “to monitor” and takes on the number, person, gender inflectional affixes, as, e.g., in *jien nimmoniterja* “I monitor”, *int immoniterjajt* “you monitored”.

Nominal derivations (nouns and adjectives) are also based on originally Arabic patterns as well as non-Arabic ones. To give an example, one way of forming nouns in Arabic is through the prefixation of *m-*. Similarly, in Maltese one finds *miżbla* “landfill”, which is derivationally related to *żibel* “rubbish”. Another, different example is *ħad(d)em* “to (make to) work” and *ħaddiem* “worker”. With non-Semitic-based derivation, an intriguing question in Maltese is whether what appears to be an affix actually has affix status in the language. Since most of the forms that carry derivational affixes are loan words, one could argue that the affixes do not have independent status but have simply been imported with their stems as words. Of course, if one were to use the standard contrastiveness criterion to isolate morphemes, one would be able to isolate affixes in most cases, as in the case of *-(z)joni* in the following examples: *ammira* “admire”, *ammirazzjoni* “admiration”, *applika* “apply”, *applikazzjoni* “application”. However, one can only unequivocally assume that an affix is identified as such when it produces new and, therefore, productive local formations, which cannot have been imported as wholes. A number of such formations exist. For example, the Romance suffix *-ata*, as in *spagettata* “a spaghetti meal”, as mentioned above, has recently been used to form the new word *żiblata* from *żibel* “rubbish” meaning “a booze up”.

### 1.5.3 Inflectional Morphology

#### Verbs

As in derivation, in inflection, there are two main types: one that is (or at least behaves as if it were) root-based, and one that is stem-based. Thus, for example, from *tefa* “extinguish” (CVCV), we get *titfi* “she extinguishes” (*t*-VCCV), *tfejt* “I extinguished” (CCV-jt) and *tfiet* “she extinguished” (CCV-Vt), with varying stem patterns. In contrast, a verb like *immoniterja* “to monitor”, has an invariant stem, thus *n-immoniterja* “I monitor”, *immoniterja-jt* “I monitored” and *immoniterja-t* “she monitored” [28, 60]. The two basic paradigms are the imperfective, which, except for the plural suffix -*u*, is mainly prefixing: *n-* “1 person”, *t-* “2 person”, *j-* “3 person masculine”, *t-* “3 person feminine”, and the perfective, which is wholly suffixing: *-t* “1/2 person singular”, *-Vt* “3 person feminine singular”, *-na* “1 person plural”, *-tu* “2 person plural”, *-u* “3 person plural”. Apart from the affixes which trigger subject-verb agreement, Maltese also has a set of personal pronoun enclitics that agree with a direct or indirect object topic: *-ni* “me”, *-k* “you”, *-u* “him”, *-ha* “her”, *-na* “us”, *-kom* “you/pl”, *-hom* “them”. Thus: *Dik il-mara qra-t-ha l-ittra* “That woman read the letter”; literally “That woman, she read it/her the letter”. The pronominal clitics also attach to nouns in the construct (possessive) and to prepositions: *ras-ha* “her head”, *ħdej-ha* “near-her” [16].

A number of contexts trigger different kinds of allomorphy both in subject agreement affixes and topic object clitics, and also in root-based stems. The default meanings of the basic imperfective and perfective forms are the present habitual and the past tense, respectively. Other tense and aspect forms are formed through the use of particles and the tense marker *kien*; thus, *nisraq* “I steal”, *qed nisraq* “I am stealing”, *sa nisraq* “I am going to steal”, *kont qed nisraq* “I was stealing”, *kont sa nisraq* “I was going to steal”, *sraqt* “I stole”, *kont sraqt* “I had stolen” [21, 25].

Verbs are negated by means of a preposed *ma* and either a suffix -*x* or some other negator such as *ħadd* “nobody”, *xejn* “nothing”, and *mkien* “nowhere”. The following are examples: *ma mortx* “I/you didn’t go”, *ma ġie ħadd* “nobody came”. Interestingly, negative imperative forms do not have a *ma* preceding them, e.g., *tiftaħx* “do not open”, and positive imperatives lack person marking, thus *iftaħ* “open” and not *\*iftaħ*.

#### Nominals

Nouns and adjectives are generally marked for gender (masculine/feminine) and number (singular, plural for adjectives; singular, plural, collective, dual for nouns). Plural forms are undefined for gender, and some forms are unspecified for number and gender, such as *martri* “martyr/s” and *interessanti* “interesting”, or for gender only, e.g., *ticer* “male/female teacher” with the plural *ticers*. With respect to number, singular masculine is generally unmarked, feminine is generally marked by a final

-a (though not exclusively), while the plural can be either marked by means of a number of different suffixes (strong plural: e.g., *karozz-a* “car”, *karozz-i* “cars”, *bahri* “sailor”, *bahri-n* “sailors”), or non-concatenatively, through stem change (broken plural: e.g., *barmil* “bucket”, *bramel* “buckets”) (see Farrugia [33] on gender, and Schembri [66] and Farrugia [32] for a discussion of the broken plural). A number of nouns also display a collective form, which is morpho-syntactically masculine singular and denotes a mass or a collective set of objects; for example, *basal* “onion”, as opposed to *basla* “one onion” and *basliet* “a number of onions”. An even smaller set of nouns has a dual form with the suffix -ajn/ejn. In Modern Maltese this is restricted to words referring to objects that typically occur in pairs (*għajnejn* “two eyes”, *idejn* “two hands”), as well as to a mixed bag of objects such as time words, e.g., *xahrejn* “two months”, *ġimagħtejn* “two weeks”, and measure terms, e.g., *uqitejn* “two ounces”, but also *ħbiżtejn* “two loaves of bread” (see Fenech [34] for a full list and discussion). The dual has not only become restricted in Modern Maltese, but is also losing its meaning and being used as a plural form. Thus, *erba’ għajn-ejn* “four eyes” is acceptable. Finally, there are also a few remnants of the so-called plural-of-the-plural forms, such as *tarf* “edge”, *truf* “edges”, *truf-ijiet* “several edges”. The difference in meaning between the last two is not so easy to characterize. To conclude, one should also mention subtractive forms like *Lħudi* “Jew”, *Lhud* “Jews”, and suppletive forms like *mara* “woman”, *nisa* “women”, *tifel* “boy”, *subien* “boys” and *tifla* “girl”, *bniet* “girls”.

### Other Closed Class Items

There is no specific morphologically marked class of adverbs in Maltese. Thus, e.g., *tajjeb* can be used to modify a noun (i.e., functions as an adjectival modifier), as in *kejk tajjeb* “a good cake” or *pasta tajba* “a good pastry”, in which case it triggers agreement, or it can be used to modify a verb, as in *ikanta tajjeb* “he sings well”, in which case it has the default masculine singular form. Very often a fused form with the prepositional *bi* “with” is used as an adverb, as in *gie bil-għaġġla* “he came hurriedly/in a hurry”, *mar bil-mod* “he went slowly”.

#### 1.5.4 Basic Syntactic Structure

Generally, given specific intonational melodies, all S, V, and O orders, except for VSO, are possible in Maltese, with SVO being arguably the unmarked case (though not SV). Maltese is a topic-oriented language, especially in the spoken form. This means that any complement phrases, including the subject noun phrase and object phrases (direct, indirect, prepositional, locational, etc.) can be placed in different positions within the sentence. Topicality for direct and indirect objects is obligatorily marked by means of the pronominal enclitics mentioned in Sect. 1.5.3 (see also Fabri and Borg [31]). The following is an example: *Il-ktieb il-biera*

*Marija xtra-t-u* “Maria bought the book yesterday”; literally “the book, yesterday, Maria she bought it/him”. The same can be expressed as: *il-bieraħ Marija xtra-t-u l-ktieb* and *Marija l-bieraħ il-ktieb xtra-t-u* (among others). As a result, Maltese has a structurally free constituent order in terms of S(subject), V(erb) and O(object) on sentence level.

It is clear from the above that Maltese displays subject verb agreement and verb (topic) object agreement [24, 29]. Moreover, agreement in Maltese is strong enough to allow both subject and object pro-drop, i.e., it allows sentences without an explicit pronominal subject or (topic) object NP. Its rich agreement morphology allows the language to reconstruct the subject/(topic) object in every case from the agreement affixes and clitics on the verb. The same applies to prepositional and nominal possessor complement phrases when a pronominal enclitic occurs on the head preposition or noun. The following are examples: *bgħat-t-hu-lha* ‘I sent him to her’, *fuq-ha* ‘on her’, *xagħar-ha* ‘her hair’. Like other pro-drop languages, Maltese also lacks expletive pronouns, i.e., it does not have anything that corresponds to *it* in *it seems that John is tired* in English. It also allows subject object inversion and extraction of the subject from a subordinate clause, two other properties associated with subject pro-drop. Maltese lacks morphological case, but has a case marker, *lil*, which marks specific, human direct objects: *qrajt il-ktieb* ‘I read the book’ but *rajt lil Pawlu* ‘I saw Paul’, and indirect objects: *bgħatt il-ktieb lil Pawlu* ‘I sent the book to Paul’.

Within the noun phrase there is agreement between the demonstrative and the adjective, on the one hand, and the noun, on the other. The definite article can also occur on the adjective as well as on the noun. However, this is not a case of agreement in terms of definiteness, because the definite article on the adjective is triggered under specific pragmatically driven conditions [27]. A form that is homophonous with the numeral *wieħed* ‘one’ can sometimes occur pre-nominally as a specificity marker (a certain *X*). Another ‘typical’ noun internal feature is the *construct state* construction, by which two juxtaposed nouns stand in a possessive relation. However, unlike Modern Standard Arabic, the possessor noun is not marked for genitive case in Maltese, since Maltese lacks morphological case altogether. Moreover, unlike Arabic, in Maltese this construction is limited, i.e., not any noun can enter into the construct relation with any other noun. In particular, typically *inalienable* relations, i.e., body parts and family relations are found in this construction. The construct is not usually possible with alienable relations, in which case a periphrastic construction with the possessive preposition *ta'* ‘of’ must be used (however, see Fabri [26] and Koptjevskaja-Tamm [54] for a more detailed discussion): *xagħar it-tifel* ‘the boy’s hair’, *il-ktieb ta' Pawlu* ‘Paul’s book’.

Yes/no questions are generally distinguished from declaratives through specific intonational patterns. Wh-questions are formed by fronting the question word: *lil min rajt?* ‘who(m) did you see?’ Relative clauses are introduced by the complementiser *li*, equivalent to ‘that’ in English: *il-ktieb li qrajt* ‘the book that I read’. Just like ‘that’, *li* can also introduce a subordinate clause: *naf li rebaħ* ‘I know he won’. Resumptive pronouns can occur under specific conditions in the form of the pronominal enclitics mentioned above.

## 1.6 Syriac

Syriac is the official liturgical language of a number of Eastern Churches in the Middle East and the Malabar Coast of India. It is the ethnic language of the Assyrians/Chaldeans/Syriacs which may number over one million users, none of whom (apart from a few known family experiments) speak it natively (that is not to say that some do not speak Neo-Aramaic dialects natively). First attested in A.D. 6, Syriac literature continues to be produced to the present day. Syriac has two dialects: Eastern and Western. The dialects differ mostly in orthography and phonology, but are quite similar in morphology and syntax.

### 1.6.1 Orthography

Syriac employs the usual Semitic consonantary which consists of 22 consonantal letters. Three scripts exist: *Estrangela* ‘rounded’ is the oldest, and is used today in most scholarly text editions. *Serto* or West Syriac has its roots in an early informal cursive hand, but becomes more dominant after the seventh century. *East Syriac* became a distinct script around the thirteenth century. Like Arabic and Hebrew, Syriac texts are mostly consonantal with three letters (*Alaph*, *Waw*, and *Yudh*) playing two roles: consonantal but also representing vowels. Since the earliest dated manuscript from A.D. 411, there is evidence of the use of a point to distinguish homographs, which is the origin of later pointing systems. The consonantal string *mn*, for example, can be either /man/ “who” or /men/ “from”: a supralinear point on *m*, or between *m* and *n*, indicates /man/, while a sublinear point indicates /men/. By the seventh century, this pointing system developed into a more comprehensive one where each vocalic phoneme had its own set of points. Around the tenth century, and entirely restricted to West Syriac, a new vocalization system was developed where letters from Greek were borrowed and placed as supralinear or sublinear symbols to indicate vowels. None of the vocalization systems superseded previous ones. Today, one finds phrases that employ the single diacritic point, along with full pointing, along with the Greek symbols. Having said that, most texts are unmarked and appear in consonantal form only. Two morphological diacritics, however, are mandatory: a supralinear two-point grapheme, similar to the German umlaut, marks plural words, and a supralinear point grapheme on the suffix *-h* indicates gender (it is absent in the masculine, and present in the feminine).

The four letters in the string *bdwl* act as prefixes and in such cases are attached to the word; e.g., *byt?* /baytā/ “house”, *Ibyt?* /lbaytā/ “to the house”, *wlbyt?* /walbaytā/ “and to the house”. Possessive pronouns and object pronominal suffixes are also attached to words; e.g., *ktb* /ktab/ “he wrote”, *ktbh* /katbeh/ “he wrote it”. Spacing in Syriac has its own complexities. Two and sometimes three words are attached without space, especially in familiar liturgical phrases. Syriac does not have special graphemes for numerals; instead, the alphabet is used to denote numerals.

A sublinear, sometimes supralinear, line has a number of functions, but primarily marks silent letters.

### 1.6.2 *Derivational Morphology*

Like the rest of the Semitic languages, Syriac morphology is that of root-and-pattern morphology. In the verbal system, the main patterns are:

1. *CCaC* (*Pʕal*)
2. *CaCCeC* in East Syriac and *CaCeC* in West Syriac (*Paʕfel*)
3. *?aCCeC* (*Aphfel*).

Each of these patterns has a corresponding reflexive pattern which is marked with the prefix *?et-*; hence, *?etCCeC*, *?etCaCCaC* (or *?etCaCaC* in West Syriac), and *?ettaCCaC* (here the *?/* in *?aCCeC* assimilates into */t/*).

The vocalization of the first pattern differs from one verb to another and is usually lexically marked. The vocalization of the remaining patterns is more or less invariant. Having said that, phonological processes may affect the vocalism; e.g., */e/* turns into */a/* when the third consonant is */r/*. The derivational mechanism here follows the same processes described in other sections of this chapter.

### 1.6.3 *Inflectional Morphology*

Conjugation patterns marking aspect (or tense), number, person, and gender are almost invariant within a pattern. However, the patterns may differ from one root class to the next. Root classes are defined by the values of the root consonants. Typically, the consonants *?*, *w*, and *y*, when they fall in any position within the root, cause idiosyncratic conjugation patterns.

Number, person, and gender are marked with suffixation in the perfect (almost equal to past tense), or circumfixation in the case of the imperfect mood (almost equal to the future tense). Hence, this part of the process is entirely concatenative, though it may cause phonological processes to be triggered within the root-and-pattern part of the stem.

Roots in which the first radical is */n/* have a different derivation in the imperfect, considering that the imperfect prefix also begins with */n/*; hence, while the imperfect of the root morpheme *k.t.b* is */nektub/*, that of the root morpheme *n.s.b* is */nessab/* < *\*nensab/*. Two phonological processes take place here, the */n/* of the root is deleted, and the second radical is duplicated. A similar process takes place with roots whose second and third radical are the same, such as *b.z.z*. Here, the imperfect is */nebbaz/*.

In addition to the inflectional morphology, prefixation and suffixation takes place. Prefixation is limited to the *bdwl* letters mentioned above. Suffixation is limited to object pronominal suffixes (in the case of nouns) and possessive pronouns in the case

of nouns and prepositions. There are two sets of possessive pronoun suffixes: one attaches to singular nouns and the other to plural nouns. In the case of prepositions, each preposition attaches to one of the sets only and the choice is idiosyncratic and lexically marked.

### **1.6.4 Syntax**

The field of Syriac syntax has not been fully explored, and most of the available studies pertain mostly to the genre of biblical texts [50, 72]. In particular, we are unaware of any computational work on Syriac syntax. We therefore do not address Syriac syntax in this chapter.

## **1.7 Contrastive Analysis**

Much research has been done in the area of comparative Semitic linguistics [46, 57, 61]. We summarize in this section the similarities and differences among the various Semitic languages across the dimensions outlined in previous sections, with an eye to the impact of those similarities and differences on computational processing.

Overall, the dimension in which Semitic languages vary the most is perhaps their writing systems, followed by phonology, morphology and then syntax. The similarities that most uniquely define the Semitic family are their morphology and to a lesser degree syntax and some of their orthographic choices. The variations among languages in the Semitic family are generally comparable to variations among members of other families.

### **1.7.1 Orthography**

In terms of their scripts, the Semitic languages show much variation: each of the languages we discussed above has its own script: Arabic, Hebrew, Syriac (has three by itself), Amharic and Maltese (Latin script). In the case of Arabic, Hebrew, Syriac and Maltese, the scripts of these four languages are historically related to the ancient Phoenician alphabet. Though these related languages use different scripts primarily, their scripts have been used successfully and for extended periods for a variety of other languages from other families: Arabic script used for Persian, Urdu, Pashto, Ottoman Turkish, among others and Hebrew used for Yiddish and Ladino. Semitic scripts have also been known to be used for writing other Semitic languages/dialects: e.g., Judeo-Arabic is Arabic written in Hebrew, Garshuni is Arabic written in Syriac script, and Aramaic (a variant of Syriac) is also written in Hebrew script.

Although Arabic, Hebrew and Syriac have distinctly different scripts, they share very similar orthographic conventions. Most prominent is the use of optional diacritic marks for short vowels and consonantal gemination, or what Daniels and Bright [19] call an *Abjad*. Maltese uses the Latin script with alphabetical spelling conventions that attempt one-to-one mapping of grapheme to phoneme. The Amharic writing system is quite distinct from Arabic, Hebrew and Syriac on one hand and from Maltese on the other hand. Amharic is a syllabary (*Abugida* writing system), that can be argued to be less ambiguous than Arabic/Hebrew/Syriac, but more so than Maltese.

Diacritic optionality is a big part of the challenge of handling Semitic languages of the Abjad family. All three Semitic sisters are quite morphologically rich and cliticizing only adds to the ambiguity space. Arabic, in contrast with Hebrew, has a morphophonemic spelling system where some affixes and clitics are spelled in a morphemic form that abstracts away from various allomorphs: e.g., the definite article, the feminine singular suffix and the masculine plural suffix. Syriac has two morpho-phonemic symbols for marking plurals and feminine singulars. Hebrew in contrast is phonemic in its spelling (modulo the missing diacritics). Hebrew spelling however has some unique challenges: first is the various overlapping allophones for some phonemes (*b:b/v, k:k/x, p:p/f, v:v/u, s:s/š*) and several graphemes with the same pronunciations (*q/k, T/t, x/k*), which have no parallel in Arabic or Maltese, although a similar phenomenon occurs in Syriac. Second, Modern Hebrew uses two spelling standards, with or without diacritic symbols that mark the vowels. Arabic dialects have no official standard orthographies and as such add a higher degree of complexity to computational processing. These orthographic issues (optional diacritics, clitics, complex phonology-orthography mapping and inconsistent spellings) contribute to why Semitic languages are challenging in the context of morphological analysis and disambiguation, speech recognition and text-to-speech, not to mention language modeling.

### 1.7.2 Phonology

Semitic phonology, like Semitic orthography, appears quite diverse. There are, however, many shared features. Emphatic, uvular and pharyngeal sounds are an important marker for Arabic and its dialects. For emphatics, Arabic has four, Syriac has two only, while Hebrew and Maltese have lost them completely (although they are retained in the script). Hebrew retains a couple in the script but not in pronunciation. The Arabic emphatics appear as ejectives in Amharic. Hebrew and Syriac, unlike Arabic and Maltese, have a few phonemes with distinct allophones: the so called *beged-kefet* (*bgdkpt*) phonemes. Syriac still makes the distinctions fully, but Modern Hebrew only does so for three (*bkp*). Arabic dialects give an interesting living example of phonological language change as the old and new pronunciations of Arabic coexist as part of the diglossic situation in the Arab world.

In terms of the Semitic vowel inventory, there is much diversity. Classical Arabic has three short and three long vowels. Arabic dialects have a wide range: Moroccan Arabic has three vowels (no length distinction) and Levantine has eight (three short and five long). Hebrew used to have long/short distinctions, some of which are retained in the script, but Modern Hebrew has no length distinction any more. Amharic and Classical Syriac have seven vowels; modern dialects of Syriac seem to have fewer (between three and five).

Shared phonemes, such as /m/ and /n/, and regularly mappable phonemes, such as /s/ and /š/, are important in establishing etymological connection across different Semitic languages. Scripts that preserve some of the historical distinction are quite important as well, especially in the case of Hebrew, establishing the relation of Arabic /q/ to Hebrew /k/ (written *q*).

### 1.7.3 Morphology

Morphology is the core of the “Semitic” linguistic classification. In contrast to the variable orthographies and phonologies of the Semitic languages, Semitic morphology has unique unifying aspects that define the Semitic language family and distinguishes it from other language families. The landmark of Semitic morphology is the use of templatic morphemes in addition to concatenative ones.

Although Arabic is the language famous for its templatic “broken” plurals, similar phenomena exist in Hebrew also to a limited extent. Templatic morphology is highly productive in Semitic languages and it depends on the concept of the *root* morpheme, a typically trilateral abstraction that captures some general meaning. Many of these roots seem to have shared meaning across multiple Semitic languages, e.g., the famous *k.t.b* (writing-related) root. Others have different or polysemous meanings, some of which are shared (more on this in Sect. 1.7.5 below). Different root types typically involving gemination of second and third radical, or weak radicals (*w/y/h/’*), seem to create similar challenges for the different Semitic sisters.

Concatenative morphology in Semitic languages has some shared features: verbs have typically two basic forms: the prefixing-stem (Arabic imperfective, Hebrew future) and the suffixing-stem (Arabic perfective, Hebrew past). The map from the morphemes to their meaning or function will vary however: the perfective/past is similar across Semitic languages, but the imperfective has a wider range, including present, future, subjunctive, etc.

The set of inflectional features (as opposed to morphemes which realize these features) are generally similar. Arabic has a larger set of inflectional features since it includes nominal case and verbal mood. Arabic has been described as a “conservative” language as opposed to other “progressive” languages, which would include Hebrew and other Arabic dialects which have lost case/mood in a similar fashion to what happened in the transition from Latin to Romance languages. It is often thought that Arabic (classical, modern standard) may reflect

some earlier forms of the proto-Semitic morphology that gave birth to the Semitic family. Of course, Arabic dialects and Maltese provide an interesting insight into how language evolves, since unlike Hebrew, which was “academic/religious” for centuries before being revived, these languages evolved naturally so to speak. The dialects seem to add to the complexities of Arabic morphology, but at the same time they remove some: case/mood are gone, but negative suffixes and indirect object clitics are introduced. This suggests that the Semitic family is continuing to change and evolve in new directions.

### 1.7.4 Syntax

In general, the syntactic properties of Semitic languages do not introduce specific difficulties for computational processing. There is much variety in the syntax of the Semitic family. In the extreme, Amharic is distinct from the rest of the languages discussed in this chapter. Amharic is a head-final language: its dominant sentential order is S-O-V and nominal phrases are Adjective-Noun. In contrast, Arabic is strongly head-initial: the dominant sentential order is V-S-O and nominal phrases are Noun-Adj. Arabic also allows S-V-O order; and Arabic dialects and Hebrew tend to be more prominently S-V-O with some V-S-O cases. Maltese and Syriac are also primarily S-V-O languages. All of the Semitic languages in this chapter, except Amharic, have post-nominal modifiers.

In Arabic dialects, Hebrew, and Maltese, the possessive construction (*idafa*, *smikhut* or *construct*) co-exists with an alternative prepositional possessive construction. Standard Arabic does not allow the prepositional construction. Both Hebrew and some Arabic dialects allow different forms of the double possession construction (e.g., *his house of the man*). Syriac primarily uses the prepositional possessive construction.

In Arabic, Hebrew, and Maltese, adjectives follow and agree with their head nouns in definiteness, gender and number. Arabic adds case agreement and odd rules for irrational plurality. Different combinations of definite/indefinite nominals are important in forming and parsing different types of syntactic constructions.

### 1.7.5 Lexicon

Semitic languages share numerous lexical items with cognate roots that are readily identifiable or easily identifiable once some of the phonological correspondences are adjusted for: *ktb* has to do with “writing”, *qwl* with voice/speech, etc. Other roots are faux amis: *rqd* is “dance” in Hebrew or “lie down” in Arabic. Other roots have multiple senses, some of which match and some do not: the root *lHm/lxm* means “bread” in Hebrew or “meat” in Arabic in its first sense, but it means “solder” in both languages in its second sense.

The spread of the Semitic languages and colonization by other language groups has led to extensive borrowing. Arabic dialects have several colonizing English, French, and Turkish influences as well as colonized Berber, Coptic and Syriac influences. Maltese can be seen as a hybrid of Arabic and Italian, with much English influence. Hebrew has numerous Russian borrowings, as well as Arabic borrowings (by design as part of its revival and by ongoing interactions with Arabic speakers). Hebrew borrowing in standard Arabic is rather ancient and restricted to religious concepts, but it is prevalent in contemporary Palestinian Arabic.

## 1.8 Conclusion

We provided in this chapter a necessarily brief overview of some of the most prominent linguistic features of Semitic languages, covering the living languages in this family for which some computational work has been done. Our aim was not to give a thorough, extensive account of any language or any particular phenomenon (many books do precisely this). Rather, we tried to provide sufficient detail that would emphasize the challenges involved in computational processing of Semitic languages, the similarities and also the differences among them. We hope that this will prove useful for readers of subsequent chapters of this book.

## References

1. Abdel-Massih ET, Abdel-Malek ZN, Badawi ESM (1979) A reference grammar of Egyptian Arabic. Georgetown University Press, Washington, DC
2. Alkuhlani S, Habash N (2011) A corpus for modeling morpho-syntactic agreement in Arabic: gender, number and rationality. In: Proceedings of the 49th annual meeting of the Association for Computational Linguistics (ACL'11), Portland
3. Amberber M (2002) Verb classes and transitivity in Amharic. Lincom, Munich
4. Anberbir T, Takara T, Gasser M, Yoon KD (2011) Grapheme-to-phoneme conversion for Amharic text-to-speech system. In: Proceedings of conference on human language technology for development, Bibliotheca Alexandrina, Alexandria
5. Badawi ESM (1973) Mustawayat al-‘Arabiyya al-mu‘asira fi Misr (the levels of modern Arabic in Egypt). Dar al-Ma‘arif, Cairo
6. Bassiouney R (2009) Arabic sociolinguistics: topics in diglossia, gender, identity, and politics. Georgetown University Press, Washington, DC
7. Beesley KR (1997) Romanization, transcription and transliteration. <http://www.xrce.xerox.com/Research-Development/Historical-projects/Linguistic-Demos/Arabic-Morphological-Analysis-and-Generation/Romanization-Transcription-and-Transliteration>
8. Berman RA (1978) Modern Hebrew structure. University Publishing Projects, Tel Aviv
9. Borer H (1988) On the morphological parallelism between compounds and constructs. In: Booij G, van Marle J (eds) Yearbook of morphology 1. Foris, Dordrecht, pp 45–65
10. Borer H (1996) The construct in review. In: Lecarme J, Lowenstamm J, Shlonsky U (eds) Studies in Afroasiatic grammar. Holland Academic Graphics, The Hague, pp 30–61
11. Borg A, Azzopardi-Alexander M (1997) Maltese. Routledge, London

12. Brincat JM (2011) Maltese and other languages: a linguistic history of Malta. Midsea, Malta
13. Brustad K (2000) The syntax of spoken Arabic: a comparative study of Moroccan, Egyptian, Syrian, and Kuwaiti dialects. Georgetown University Press, Washington, DC
14. Buckwalter T (2002) Buckwalter Arabic morphological analyzer. Linguistic Data Consortium (LDC), Philadelphia. Catalog number LDC2002L49 and ISBN 1-58563-257-0
15. Buckwalter T (2004) Buckwalter Arabic morphological analyzer version 2.0. LDC, Philadelphia. Catalog number LDC2004L02, ISBN 1-58563-324-0
16. Camilleri M (2009) Clitics in Maltese. BA thesis, University of Malta
17. Cowell MW (1964) A reference grammar of Syrian Arabic. Georgetown University Press, Washington, DC
18. Daniels PT (1997) Scripts of Semitic languages. In: Hetzron R (ed) *The Semitic languages*. Routledge, London/New York, chap 2, pp 16–45
19. Daniels PT, Bright W (eds) (1996) *The World's writing systems*. Oxford University Press, New York
20. Doron E (1983) Verbless predicates in Hebrew. PhD thesis, University of Texas at Austin
21. Ebert K (2000) Aspect in Maltese. In: Dahl O (ed) *Tense and aspect in the languages of Europe*. Mouton de Gruyter, Berlin
22. El Kholy A, Habash N (2010) Techniques for Arabic morphological detokenization and orthographic denormalization. In: Proceedings of LREC-2010, Malta
23. Erwin W (1963) A short reference grammar of Iraqi Arabic. Georgetown University Press, Washington, DC
24. Fabri R (1993) Kongruenz und die Grammatik des Maltesischen. Niemeyer, Tübingen
25. Fabri R (1995) The tense and aspect system of Maltese. In: Thieroff R (ed) *Tempussysteme in Europäischen Sprachen II*. Niemeyer, Tübingen, pp 327–343
26. Fabri R (1996) The construct state and the pseudo-construct state in Maltese. *Rivista di Linguistica* 8(1):229–244
27. Fabri R (2001) Definiteness marking and the structure of the NP in Maltese. *Verbum* 23(2):153–172
28. Fabri R (2009) Stem allomorphy in the Maltese verb. *Ilsienna – Our Language* 1/2009:1–20
29. Fabri R (2009) To agree or not to agree: suspension of formal agreement in Maltese. In: Fabri R (ed) *Maltese linguistics: a snapshot; in memory of Joseph A. Cremona (1922–2003)*. Il-Lingwa Tagħna – Our Language. Brockmeyer, Bochum, pp 35–61
30. Fabri R (2010) Maltese. *Revue Belge de Philologie et d'Histoire* 88(3):791–816
31. Fabri R, Borg A (2002) Topic, focus and word order in Maltese. In: Abderrahim Y, Benjelloun F, Dahbi M, Iraqui-Sinaceur Z (eds) *Aspects of the dialects of Arabic today. Proceedings of the 4th conference of the international Arabic Dialectology Association (AIDA)*, Amapatril, Rabat, pp 354–363
32. Farrugia A (2008) Maltimorph: a computational analysis of the Maltese broken plural. BSc thesis, University of Malta
33. Farrugia G (2010) Il-Ġens grammatikali fil-Malti. PhD thesis, University of Malta
34. Fenech E (1996) Functions of the dual suffix in Maltese. *Rivista di Linguistica* 8:89–100
35. Ferguson CF (1959) Diglossia. *Word* 15(2):325–340
36. Gadish R (ed) (2001) *Klalei ha-Ktiv Hasar ha-Niqqud*, 4th edn. Academy for the Hebrew Language, Brooklyn (in Hebrew)
37. Gasser M (2010) A dependency grammar for Amharic. In: *Proceedings of the workshop on language resources and human language technologies for Semitic languages*, Valletta
38. Glinert L (1989) The grammar of modern Hebrew. Cambridge University Press, Cambridge
39. Habash N (2006) On Arabic and its dialects. *Multiling Magazi* 17(81). Getting Started Guide: Middle East Insert, pp 12–15
40. Habash N (2007) Arabic morphological representations for machine translation. In: van den Bosch A, Soudi A (eds) *Arabic computational morphology: knowledge-based and empirical methods*. Springer, Dordrecht
41. Habash N (2010) Introduction to Arabic natural language processing. In: *Synthesis lectures on human language technologies*. Morgan & Claypool, San Rafael. doi:<http://dx.doi.org/10.2200/S00277ED1V01Y201008HLT010>

42. Habash N, Soudi A, Buckwalter T (2007) On Arabic transliteration. In: Soudi A, Neumann G, van den Bosch A (eds) Arabic computational morphology, text, speech and language technology, vol 38. Springer, Dordrecht, chap 2, pp 15–22. [http://dx.doi.org/10.1007/978-1-4020-6046-5\\_2](http://dx.doi.org/10.1007/978-1-4020-6046-5_2)
43. Habash N, Diab M, Rabnow O (2012) Conventional orthography for dialectal Arabic. In: Proceedings of the language resources and evaluation conference (LREC), Istanbul
44. Harrell R (1962) A short reference grammar of Moroccan Arabic. Georgetown University Press, Washington, DC
45. Hetzron R (1970) Towards an Amharic case grammar. *Stud Afr Linguist* 1:301–354
46. Hetzron R (ed) (1997) The Semitic languages. Routledge, London/New York
47. Holes C (2004) Modern Arabic: structures, functions, and varieties. Georgetown University Press, Washington, DC
48. Horvath J, Wexler P (1997) Relexification in Creole and Non-Creole languages – with special attention to Haitian Creole, Modern Hebrew, Romani, and Rumanian, Mediterranean Language and culture monograph series, vol xiii. Harrassowitz, Wiesbaden
49. Itai A, Wintner S (2008) Language resources for Hebrew. *Lang Resour Eval* 42(1):75–98
50. Joosten J (1996) The Syriac language of the Peshitta and Old Syriac versions of Matthew: syntactic structure, inner-Syriac developments and translation technique. In: Studies in Semitic languages and linguistics. Brill, Leiden
51. Kapeliuk O (1988) Nominalization in Amharic. Franz Steiner, Stuttgart
52. Kapeliuk O (1994) Syntax of the noun in Amharic. O. Harrassowitz, Wiesbaden
53. Kaye AS, Rosenhouse J (1997) Arabic dialects and Maltese. In: Hetzron R (ed) The Semitic languages. Routledge, London/New York, chap 14, pp 263–311
54. Koptjevskaia-Tamm M (1996) Possessive NPs in Maltese: alienability, iconicity and grammaticalization. *Riv di Linguist* 8(1):245–274
55. Kramer R (2009) Definite markers, Phi features, and agreement: a morphosyntactic investigation of the Amharic DP. PhD thesis, University of California
56. Leslau W (1995) Reference grammar of Amharic. Harrassowitz, Wiesbaden
57. Lipiński E (2001) Semitic languages, outline of a comparative grammar. Peeters, Leuven
58. Loos EE, Anderson S, Dwight HJ Day, Jordan PC, Wingate JD (2004) Glossary of linguistic terms. <http://www.sil.org/linguistics/GlossaryOfLinguisticTerms/>
59. McCarthy JJ (1981) A prosodic theory of nonconcatenative morphology. *Linguist Inq* 12(3):373–418
60. Mifsud M (1995) Loan verbs in Maltese: a descriptive and comparative study. Brill, New York
61. Moscati S (1969) An introduction to the comparative grammar of the Semitic languages, phonology and morphology. Otto Harrassowitz, Wiesbaden
62. Ordan N, Wintner S (2005) Representing natural gender in multilingual lexical databases. *Int J Lexicogr* 18(3):357–370
63. Ornan U (2003) The final word. University of Haifa Press, Haifa (in Hebrew)
64. Puech G (Forthcoming) Syllabic structure and stress in Maltese. In: Caruana S, Fabri R, Stolz T (eds) Variation and change: the dynamics of Maltese in space, time, and society. Akademie Verlag, Berlin
65. Rogers H (2005) Writing systems: a linguistic approach. Blackwell Publishing, Malden
66. Schembri T (2006) The broken plural in Maltese – an analysis. B.A. thesis, University of Malta
67. Shimron J (ed) (2003) Language processing and acquisition in languages of semitic, root-based, morphology. In: Language acquisition and language disorders, vol 28. John Benjamins, Amsterdam/Philadelphia
68. Smrž O (2007) Functional Arabic morphology. Formal system and implementation. PhD thesis, Charles University in Prague
69. Teferra A, Hudson G (2007) Essentials of Amharic. Rüdiger Köppe Verlag, Köln
70. Twist AE (2006) A psycholinguistic investigation of the verbal morphology of Maltese. PhD thesis, University of Arizona
71. Ussishkin A, Twist A (2009) Auditory and visual lexical decision in Maltese. In: Comrie B, Fabri R, Mifsud M, Hume E, Mifsud M, Stolz T, Vanhove M (eds) Introducing Maltese linguistics. Benjamins, Amsterdam, pp 207–231

72. Williams PJ (2001) Studies in the Syntax of the Peshitta of 1 kings. In: Monographs of the Peshitta institute. Brill, Leiden
73. Wintner S (2000) Definiteness in the Hebrew noun phrase. *J Linguit* 36:319–363
74. Wintner S (2009) Language resources for Semitic languages: challenges and solutions. In: Nirenburg S (ed) Language engineering for lesser-studied languages. IOS, Amsterdam, pp 277–290
75. Zuckermann G (2003) Language contact and lexical enrichment in Israeli Hebrew. Palgrave Macmillan, London/Ney York
76. Zwicky AM (1985) Clitics and particles. *Language* 61(2):283–305
77. Zwicky AM, Pullum GK (1983) Cliticization vs. inflection: English n't. *Language* 59(3): 502–513

# Chapter 2

## Morphological Processing of Semitic Languages

Shuly Wintner

### 2.1 Introduction

This chapter addresses morphological processing of Semitic languages. In light of the complex morphology and problematic orthography of many of the Semitic languages, the chapter begins with a recapitulation of the challenges these phenomena pose on computational applications. It then discusses the approaches that were suggested to cope with these challenges in the past. The bulk of the chapter, then, discusses available solutions for morphological processing, including analysis, generation, and disambiguation, in a variety of Semitic languages. The concluding section discusses future research directions.

Semitic languages are characterized by complex, productive morphology, with a basic word-formation mechanism, root-and-pattern, that is unique to languages of this family. Morphological processing of Semitic languages therefore necessitates technology that can successfully cope with these complexities.<sup>1</sup> Several linguistic theories, and, consequently, computational linguistic approaches, are often developed with a narrow set of (mostly European) languages in mind. The adequacy of such approaches to other families of languages is sometimes sub-optimal. A related issue is the long tradition of scholarly work on some Semitic languages, notably Arabic [109] and Hebrew [117], which cannot always be easily consolidated with contemporary approaches.

Inconsistencies between modern, English-centric approaches and traditional ones are easily observed in matters of lexicography. In order to annotate corpora or produce tree-banks, an agreed-upon set of part-of-speech (POS) categories is

---

<sup>1</sup>Parts of this introduction are based on and adapted from [137].

S. Wintner (✉)  
University of Haifa, Haifa, Israel  
e-mail: [shuly@cs.haifa.ac.il](mailto:shuly@cs.haifa.ac.il)

required. Since early approaches to POS tagging were limited to English, resources for other languages tend to use “tag sets”, or inventories of categories, that are minor modifications of the standard English set. Such an adaptation is problematic for Semitic languages. As noted in the previous chapter, there are good reasons to view nouns, adjectives and numerals as sub-categories of a single category, *nominals*. Furthermore, the distinction between verbs and nominals is blurry. Netzer et al. [101] discuss a similar issue related to the correct tagging of modals in Hebrew. Even the correct citation form to use in dictionaries is a matter of some debate, as Arabic traditional dictionaries are root-based, rather than lemma-based [43].

These issues are complicated further when morphology is considered. The rich, non-concatenative morphology of Semitic languages frequently requires innovative solutions that standard approaches do not always provide. After a brief introduction of some basic notions (Sect. 2.2), we recapitulate some of these challenges in Sect. 2.3, and review the long line of proposed computational solutions in Sect. 2.4. Section 2.5 lists some available computational implementations for the morphology of various Semitic languages. Section 2.6 then discusses implementations of morphological disambiguation for several Semitic languages. We conclude the chapter with directions for future research.

## 2.2 Basic Notions

The word ‘word’ is one of the most loaded and ambiguous notions in linguistic theory [76]. Since most computational applications deal with written texts (as opposed to spoken language), the most useful notion is that of an *orthographic* word. This is a string of characters, from a well-defined alphabet of *letters*, delimited by spaces, or other delimiters, such as punctuation. A *text* typically consists of sequences of orthographic words, delimited by spaces or punctuation; orthographic words in a text are often referred to as *tokens*.

Orthographic words are frequently not atomic: they can be further divided to smaller units, called *morphemes*. Morphemes are the smallest meaning-bearing linguistic elements; they are elementary pairings of form and meaning. Morphemes can be either *free*, meaning that they can occur in isolation, as a single orthographic word; or *bound*, in which case they must combine with other morphemes in order to yield a word. For example, the word *two* consists of a single (free) morpheme, whereas *dogs* consists of two morphemes: the free morpheme *dog*, combined with the bound morpheme *-s*. The latter form indicates the fact that it must combine with other morphemes (hence the preceding dash); its function is, of course, denoting plurality. When a word consists of some free morpheme, potentially with combined bound morphemes, the free morpheme is called a *stem*, or sometimes *root*.

Bound morphemes are typically *affixes*. Affixes come in many varieties: *prefixes* attach to a stem before the stem (e.g., *re-* in *reconsider*), *suffixes* attach after the stem (*-ing* in *dreaming*), *infixes* occur inside a stem (e.g., the *t* in Arabic *ijtahada*, from *jahada*), and *circumfixes* surround the stem they combine with (e.g., Hebrew *ti-u* in *tigdalu* “you will grow”). Some bound morphemes are likely *clitics* [140], but as in the previous chapter, we blur the distinction between clitics and affixes here.

Morphological processes define the shape of words. They are usually classified to two types of processes. *Derivational* morphology deals with word formation; such processes can create new words from existing ones, potentially changing the category of the original word. For example, the processes that create *faithfulness* from *faithful*, and *faithful* from *faith*, are derivational. Such processes are typically not highly productive; for example, one cannot derive *\*loveful* from *love*. In contrast, *inflectional* morphology yields *inflected forms*, variants of some *base*, or *citation* form, of words; these forms are constructed to adhere to some syntactic constraints, but they do not change the basic meaning of the base form. Inflectional processes are usually highly productive, applying to most members of a particular word class. For example, English nouns inflect for *number*, so most nouns occur in two forms, the singular (which is considered the citation form) and the plural, regularly obtained by adding the suffix *-s* to the base form.

Word formation in Semitic languages is based on a unique mechanism, known as *root-and-pattern*. Words in this language family are often created by the combination of two bound morphemes, a *root* and a *pattern*. The root is a sequence of consonants only, typically three; and the pattern is a sequence of vowels and consonants with open slots in it. The root combines with the pattern through a process called *interdigitation*: each letter of the root (*radical*) fills a slot in the pattern. For example, the Hebrew root *p.t.x*, denoting a notion of opening, combines with the pattern *maCCeC* (the slots are denoted by *C*), typically denoting tools and instruments, to yield *maptex* “key”.

In addition to the unique root-and-pattern morphology, Semitic languages are characterized by a productive system of more standard affixation processes. These include prefixes, suffixes, infixes and circumfixes, which are involved in both inflectional and derivational processes (see the previous linguistic-introduction chapter). Consider the Arabic word *wasayaktubuwnaha* “and they will write it”. A possible analysis of this complex word defines the stem as *aktub* “write”, with an inflectional circumfix, *y—uwna*, denoting third person masculine plural, an inflectional suffix, *-ha* “it”, and two prefixes, *sa-* “will” and *wa-* “and”. For more information on Arabic morphology from a computational perspective, see [127]; [63, Chap. 4]. For a good introduction to computational morphology in general, consult [112, 128].

## 2.3 The Challenges of Morphological Processing

Morphological processing is a crucial component of many natural language processing (NLP) applications. Whether the goal is information retrieval, question answering, text summarization or machine translation, NLP systems must be aware of word structure. For some languages and for some applications, simply stipulating a list of surface forms is a viable option; this is not the case for languages with complex morphology, in particular Semitic languages, both because of the huge number of potential forms and because of the difficulty of such an approach to

handle out-of-lexicon items (in particular, proper names), which may combine with prefix or suffix particles. For example, the Hebrew prefix *I-* “to” can combine with any proper name denoting a location, an organization or a person.

An alternative solution would be a dedicated morphological analyzer, implementing the morphological and orthographic rules of the language. Ideally, a morphological analyzer for any language should reflect the rules underlying derivational and inflectional processes in that language. Of course, the more complex the rules, the harder it is to construct such an analyzer. The main challenge of morphological analysis of Semitic languages stems from the need to faithfully implement a complex set of interacting rules, some of which are non-concatenative.

Once such a grammar is available, it typically produces more than one analysis for any given surface form; in other words, Semitic languages exhibit a high degree of *morphological ambiguity*, which has to be resolved in a typical computational application. The level of morphological ambiguity is higher in many Semitic languages than it is in English, due to the rich morphology and deficient orthography. This calls for sophisticated methods for disambiguation. While in English (and other European languages) morphological disambiguation may amount to POS tagging, Semitic languages require more effort, since determining the correct POS of a given token is intertwined with the problem of segmenting the token to morphemes, the set of morphological features (and their values) is larger, and consequently the number of classes is too large for standard classification techniques. Several models were proposed to address these issues.

Contemporary approaches to part-of-speech tagging are all based on machine learning: a large corpus of text is manually annotated with the correct POS for each word; then, a classifier is trained on the annotated corpus, resulting in a system that can predict POS tags for unseen texts with high accuracy. The state of the art in POS tagging for English is extremely good, with accuracies that are indistinguishable from human level performance. Various classifiers were built for this task, implementing a variety of classification techniques, such as Hidden Markov Models (HMM) [26], Average Perceptron [37], Maximum Entropy [111, 130, 131, 133, 134], Support Vector Machines (SVM) [58], and others.

For languages with complex morphology, and Semitic languages in particular, however, these standard techniques do not perform as well, for several reasons:

1. Due to issues of orthography, a single token in several Semitic languages can actually be a sequence of more than one lexical item, and hence be associated with a sequence of tags. For example, the Hebrew form *šbth* can be interpreted as *š+b+h+th* “that+in+the+tea”, corresponding to a tag sequence consisting of a subordinating conjunction, followed by a preposition, a determiner and a noun.
2. The rich morphology implies a much larger tagset, since tags reflect the wealth of morphological information which words exhibit. The richer tagset immediately implies problems of data sparseness, since most of the tags occur only rarely, if at all, in a given corpus. For example, the MILA Hebrew morphological analyzer [80] produces 22 different parts of speech, some with subcategories; 6 values for the number feature (including disjunctions of values), 4 for gender, 5 for person,

7 for tense and 3 for nominal state. Possessive pronominal suffixes can have 15 different values, and prefix particle sequences can theoretically have hundreds of different forms. While not all the combinations of these values are possible, the number of possible analyses (i.e., the size of the tagset) is in the thousands.

3. As a result of both orthographic deficiencies and morphological wealth, word forms in Semitic languages tend to be ambiguous. Itai and Wintner [80] report an average of 2.6 analysis per word in their corpora. In some cases, different analyses are identical in all their features, except the lexical item, a phenomenon that makes morphological disambiguation closer to the problem of word sense disambiguation than to standard POS tagging.
4. Word order in Semitic is relatively free, and in any case freer than in English.

An additional challenge of morphological processing of Semitic languages, with an emphasis on Arabic, stems from the *form–function* discrepancy. The *form* of words in these languages typically provides good hints for some of the morphological features of the word, or its *function*; in many cases, however, the form and the function are in disagreement. A concrete example is the phenomenon of *broken plural* forms in Arabic. For a significant number of nouns, the plural form is not obtained by the concatenation of a plural suffix, but rather by an internal change (not unlike umlauting) that renders the surface form, which is plural in function, singular in form. A related phenomenon involves gender agreement in Arabic: while adjectives must agree with their head nouns in gender, when the noun is plural and irrational (non-human), the adjective must be feminine singular (see a detailed discussion in [3]).

Somewhat similarly, Hebrew nouns are marked for gender by a small number of suffixes; but several masculine-appearing nouns are actually feminine, and vice versa. Furthermore, Hebrew has two plural suffixes, *-im* for plural nouns and *-wt* for feminine nouns, but a non-negligible number of feminine nouns take the masculine suffix and vice versa.

Finally, it is important to note that morphological processing of Semitic languages is often handicapped by subtle orthographic issues [28]. Hebrew, for example, has a writing system that encodes vocalic information using a large set of diacritics; this system, however, is rarely in use, and most contemporary texts are written without the diacritics. Unfortunately, while a standard for non-vocalized Hebrew exists [53], it is not adhered to by most authors, and consequently the same word may be spelled in different ways, sometimes even within the same document. Arabic suffers from related problems, especially when the various dialects are considered, in which standardized forms do not exist [71].

## 2.4 Computational Approaches to Morphology

No single method exists that provides an adequate solution for the challenges involved in morphological processing of Semitic languages. The most common approach to morphological processing of natural language is *finite-state technology*

[22, 81, 83, 89, 113]. The adequacy of this technology for Semitic languages has frequently been challenged, but clearly, with some sophisticated developments, such as flag diacritics [19], multi-tape automata [88] or registered automata [36], finite-state technology has been effectively used for describing the morphological structure of several Semitic languages [8, 16, 17, 68, 85, 88, 138]. We survey this technology in the present section.

### 2.4.1 Two-Level Morphology

Two-level morphology was “the first general model in the history of computational linguistics for the analysis and generation of morphologically complex languages” [84]. Developed by Koskenniemi [89], this technology facilitates the specification of rules that relate pairs of surface strings through systematic rules. Such rules, however, do not specify how one string is to be derived from another; rather, they specify mutual constraints on those strings. Furthermore, rules do not apply sequentially. Instead, a set of rules, each of which constrains a particular string-pair correspondence, is applied in parallel, such that all the constraints must hold simultaneously. In practice, one of the strings in a pair would be a surface realization, while the other would be an underlying form. Thus, for example, the Hebrew surface form *[xicim]* “arrows” can correspond to the underlying form *xec+im* through the mapping:

```
x i c 0 i m
x e c + i m
```

where ‘0’ is the empty string. The example reflects a rule that maps *[i]* to *e* in the context of the plural suffix *im*; the upper string is the surface realization, and the lower is its underlying form. The underlying forms are further constrained by consulting a *lexicon*.

One of the greatest advantages of two-level morphology is that rules are entirely declarative: indeed, the original formulation of [89] allows for both analysis and generation within the same grammar. The formalism was later implemented as part of the Xerox tools (Sect. 2.4.3); two-level rules are compiled to finite-state transducers, which indeed allow for both analysis and generation.

### 2.4.2 Multi-tape Automata

Two-level morphology was applied to one Semitic language, Akkadian, by Kataja and Koskenniemi [85]. However, the applicability of the technology to Semitic languages in general was challenged by Lavie et al. [91], who describe some difficulties of this technology in accounting for Hebrew verb inflections. Lavie et al. [91] conclude: “The Two Level rules are not the natural way to describe... verb

inflection process. The only alternative choice... is to keep all bases... it seems wasteful to save all the secondary bases of verbs of the same pattern.”

Addressing such issues, [88] expands the traditional two-level model to  $n$ -tape automata, following insight originally suggested by Kay [86] and Kataja and Koskenniemi [85]. The two levels of expression are expanded: one of them is retained for the surface form, but the lexical string can now be spread across multiple representations (e.g., one for the root and one for the pattern). Thus, elements that are separated on the surface (such as the root’s consonants) can be adjacent on a particular lexical level.

Using multi-tape automata, [88] provides elegant solutions for derivational and inflectional morphology of two Semitic languages, Syriac and Arabic. The same approach is then extended by Habash et al. [68], who define a multi-tape automaton consisting of five tapes: one for the pattern and affixational morphemes, one for the root, one for the vocalism, one for phonological information and one for the orthography. This model is then successfully applied to both MSA and dialectal Arabic [6, 65, 68]. The model is detailed in [7].

Hulden [79], however, notes that no other systems were built using this technology, and conjectures that the reason may be that “when the number of tapes grows, the required joint symbol alphabet grows with exponential rapidity unless special mechanisms are devised to curtail this growth. This explosion in the number of transitions in an  $n$ -tape automaton can in many cases be more severe than the growth in the number of states of a complex grammar.” To alleviate the problem, [79] describes an algorithm that simulates an  $n$ -tape automaton with a simple single-tape finite-state machine. Consequently, the elegant representation of multi-tape automata can be retained, while the conversion algorithm facilitates an implementation using standard tools such as the ones discussed in Sect. 2.4.3. Indeed, [79] uses Foma [78] to efficiently implement a grammar of Arabic verbal morphology over 2,000 roots.

### 2.4.3 The Xerox Approach

One of the most popular toolboxes for developing finite-state grammars comes from Xerox, and is discussed in detail by Beesley and Karttunen [22]. The Xerox tools consist of several description languages, including a formalization of two-level morphology, but also a variant, XFST, of the calculus proposed by Kaplan and Kay [83]. Along with the description languages come compilers that convert morphological grammars to finite-state transducers, and programs that implement analysis and generation with these transducers.

To address the special needs of languages with non-concatenative morphology, XFST provides two special mechanisms. First, the *compile-replace* mechanism [21] facilitates the reapplication of the regular-expression compiler to its own output. This allows for a compact definition of some non-concatenative morphological processes, and [19] uses it to construct a morphological grammar of Arabic. Second,

Beesley [20] proposes a method, called *flag diacritics*, which adds features to symbols in regular expressions to enforce dependencies between separated parts of a string. These dependencies are then enforced by different kinds of unification actions.

While the Xerox tools have for many years been the de-facto standard of finite-state technology, they have also been proprietary, a fact that limited their distribution and popularity. Several competing formalisms were developed over the years, of which we note Foma [78] because it is, to a large extent, compatible with the syntax of several Xerox tools, while being completely open-source.

#### 2.4.4 Registered Automata

*Finite-state registered automata* [36] were developed with the goal of facilitating the expression of various non-concatenative morphological phenomena in an efficient way. The main idea is to augment standard finite-state automata with (finite) amount of memory, in the form of *registers* associated with the automaton transitions. This is done in a restricted way that saves space but does not add expressivity. The number of registers is finite, usually small, and eliminates the need to duplicate paths as it enables the automaton to ‘remember’ a finite number of symbols. Technically, each arc in the automaton is associated (in addition to an alphabet symbol) with an action on the registers. Cohen-Sygal and Wintner [36] define two kinds of actions, *read* and *write*. The former allows an arc to be traversed only if a designated register contains a specific symbol. The latter writes a specific symbol into a designated register when an arc is traversed.

Cohen-Sygal and Wintner [36] show that finite-state registered automata can efficiently model several non-concatenative morphological phenomena, including circumfixation, root and pattern word formation in Semitic languages, vowel harmony, limited reduplication etc. The representation is highly efficient: for example, to account for all the possible combinations of  $r$  roots and  $p$  patterns, an ordinary FSA requires  $O(r \times p)$  arcs whereas a registered automaton requires only  $O(r + p)$  arcs. Unfortunately, no implementation of the model exists as part of an available finite-state toolkit.

#### 2.4.5 Analysis by Generation

Most of the approaches discussed above allow for a *declarative* specification of (morphological) grammar rules, from which both analyzers and generators can be created automatically. A simpler, less generic yet highly efficient approach to the morphology of Semitic languages had been popular with actual applications. In this framework, which we call *analysis by generation* here, the morphological rules that describe word formation and/or affixation are specified in a way that supports

generation, but not necessarily analysis. Coupled with a lexicon of morphemes (typically, base forms and concatenative affixes), such rules can be applied in one direction to generate all the surface forms of the language. This can be done off-line, and the generated forms can then be stored in a database; analysis, in this paradigm, amounts more or less to simple table lookup.

Some of the very first morphological processors of Semitic languages were developed in this way. Probably the first example is the Hebrew morphological system of [29, 122], see Sect. 2.5.3. Exactly the same approach is now used in the MILA morphological analyzer of Hebrew [80] (Sect. 2.5.3). And a very similar approach underlies the most popular morphological analyzer of Arabic, BAMA [27]: Again, a set of rules (called the *compatibility table*) determines how lexemes and affixes (stored in separate lexicons) can combine; at analysis time, a surface form is divided to a sequence of prefix + lexeme + suffix in all possible ways, and the lexicons are consulted to determine which potential combination is indeed valid (see Sect. 2.5.2).

#### 2.4.6 Functional Morphology

*Functional morphology* [51] is a computational framework for defining language resources, in particular lexicons. It is a language-independent tool, based on a *word-and-paradigm* model, which allows the grammar writer to specify the inflectional paradigms of words in a specific language in a similar way to printed paradigm tables. A lexicon in functional morphology consists of a list of words, each associated with its paradigm name, and an inflection engine that can apply the inflectional rules of the language to the words of the lexicon.

This framework was used to define morphological grammars for several languages, including modeling of non-concatenative processes such as vowel harmony, reduplication, and templatic morphology [50]. In particular, [125] uses this paradigm to implement a morphological processor of Arabic.

### 2.5 Morphological Analysis and Generation of Semitic Languages

We survey in this section the current state of the art in morphological analysis and generation of various Semitic languages. While much effort was put into the development of systems for processing (Modern Standard) Arabic and Hebrew, for other languages the development of such tools lags behind.

We use the term *analysis* in this chapter to refer to the task of producing *all* the possible analyses of a given word form, independently of its context. The problem of producing the *correct* analysis in the context is called *disambiguation* here, and is discussed in detail in Sect. 2.6.

### 2.5.1 Amharic

Computational work on Amharic began only recently. Fissaha and Haller [49] describe a preliminary lexicon of verbs, and discuss the difficulties involved in implementing verbal morphology with XFST. XFST is also the framework of choice for the development of an Amharic morphological grammar [8, 9]; but evaluation on a small set of 1,620 words reveal that while the coverage of the grammar on this corpus is rather high (85–94 %, depending on the part of speech), its precision is low and many word forms (especially verbs) are associated with wrong analyses.

Argaw and Asker [11] describe a stemmer for Amharic. Using a large dictionary, the stemmer first tries to segment surface forms to sequences of prefixes, stem, and affixes. The candidate stems are then looked up in the dictionary, and the longest found stem is chosen (ties are resolved by the frequency of the stem in a corpus). Evaluation on a small corpus of 1,500 words shows accuracy of close to 77 %.

The state of the art in Amharic, however, is most probably *HornMorpho* [56, 57]: it is a system for morphological processing of Amharic, as well as Tigrinya (another Ethiopian Semitic language) and Oromo (which is not Semitic). The system is based on finite-state technology, but the basic transducers are augmented by feature structures, implementing ideas introduced by Amtrup [10]. Manual evaluation on 200 Tigrinya verbs and 400 Amharic nouns and verbs shows very accurate results: in over 96 % of the words, the system produced all and only the correct analyses.

### 2.5.2 Arabic

Recent years saw an increasing interest in computational approaches to Arabic morphology [5]. Attempts to automatically analyze the structure of Arabic words date back over 50 years [34]. Several early works were done in the finite-state framework of the Xerox tools (Sect. 2.4.3). Beesley [17] describes an early system for morphological analysis and generation. The input is given in the standard Arabic script, either vocalized or not, and the output includes the root, the pattern, a list of affixes and a plethora of morphological information in the form of feature-value pairs. The implementation was carried out in an early version of the Xerox tools, which resembles to a high degree the two-level formalism. Beesley [18] uses the newly-introduced *flag diacritics* in XFST to provide a more elegant morphological grammar, whose implementation as an online web-based service is described in [19]. In a similar vein, [88] demonstrates the utility of multi-tape automata (Sect. 2.4.2) by providing examples from Arabic.

Other works from approximately the same period seem to be more focused on an actual application, rather than on elegant and efficient representation of morphological processes. Al-Shalabi and Evens [4] extend an earlier system and

a large-scale lexicon to an analyzer for (mainly regular, but also some irregular) verb forms. Berri et al. [25] describe a morphological analyzer that uses an object-oriented model to represent morphological rules affecting both verbs and nouns, along with a dedicated algorithm that identifies affixes and separates them from the stem. Rules are divided into *regular* and *exception* handling. No details are provided on the coverage of the system. Darwish [39] discusses the rapid development of a shallow morphological analyzer. Given a large set of word–root pairs, the system learns to identify the roots of (mainly regular) word forms. Evaluation on a large set reveals high coverage but, unsurprisingly, rather low accuracy.

The state of the art in Arabic morphological analysis, however, is most likely *BAMA*, the morphological analyzer of Buckwalter [27], which combines wide coverage with detailed, linguistically informative analyses. *BAMA* is based on a large-scale lexicon of base forms, along with lists of prefixes and suffixes. A second part of this database includes a list of compatibility rules, which govern the combination of stems with affixes. Finally, an efficient engine implements the rules as well as lexical lookup. The result is a highly-efficient, broad-coverage (and freely-available) analyzer. *BAMA* (most recently called *SAMA*, or *Standard Arabic Morphological Analyzer*) is the official morphological analyzer used by the Linguistic Data Consortium (LDC) for the *Penn Arabic Treebank* [93], a language resource used by most practitioners interested in Arabic disambiguation and parsing.

Based on this analyzer, [61] has built *Aragen*, a system for generating Arabic word forms from underlying morphological descriptions. Using the same databases of [27], *Aragen* implements a different engine that reverses the operation of the analyzer. The current state of the art in Arabic morphological generation is a revised version of *Aragen*, called *Almorgeana* [62].

A different approach was advanced in the context of the NooJ platform [123]. NooJ is a linguistic development environment that facilitates the definition of large-coverage dictionaries and grammars, compiling them into systems that can efficiently parse real-world corpora. NooJ has been used for the construction of Arabic morphological and syntactic processors [23], as well as for part-of-speech tagging and morphological analysis [82].

More recent approaches to Arabic morphology are done with an eye to syntactic processing. For example, [125, 126] addresses Arabic morphology in the framework of Functional Morphology [51]. His system, *ElixirFM*, extends the original functionality of the framework by addressing the specific needs of Arabic morphology. The system not only provides (derivational and inflectional) analyses of word forms, but can also recognize their grammatical functions. A different system, *Kawaakib* [12], combines a set of both morphological and syntactic operators that are represented as finite-state automata.

For a full, detailed and lucid exposition of computational processing of Arabic, with a focus on morphology, refer to [63].

### 2.5.3 Hebrew

Computational work on Hebrew began almost fifty years ago.<sup>2</sup> Very early approaches [29, 122] were superseded by a large-scale project dealing with various aspects of computational linguistics, natural language processing and information retrieval: the *Responsa* project [30, 31, 52]. Algorithms were developed for automatic generation of all the possible inflected and derived forms of all the bases in Hebrew, including those obtained by the combination of prepositions, conjunctions, articles etc. Based on the generation algorithm, a file was created which included all the possible Hebrew word forms, approximately 2,500,000 words. The analyzer implements a program which strips the possible affixes off the input word and checks whether the obtained result is indeed a legal word. Thus, morphological analysis and generation are incorporated in a complete system for computational processing of Hebrew (albeit not Modern, contemporary Hebrew). A more modern implementation of this system was later commercialized [32, 33].

A different approach to Hebrew morphology is based on the *Phonemic Script* [105], which is an unambiguous writing system for Hebrew, preserving the deep structure of the words. Based on this script, a wide variety of programs were developed, including a program for vocalization [104], a program for the preparation of concordances and indexes [103], especially developed for a database of legal texts [106], a series of programs for morphological analysis and generation [60, 108, 120, 121] and programs for converting phonemic script to the standard Hebrew script [107].

Morphological analysis is one aspect of a commercial system, *Context*, designed for information retrieval [110]. Another commercial system, *Avgad* [24], is based on a dictionary of 25,000 entries, which form the base for “hundreds of thousands” of Hebrew words (including inflected forms). It was used by Segal (Morphological analyzer for unvocalized Hebrew words, <http://www.cs.technion.ac.il/~erelsgl/hmntyx.zip>, unpublished work, 1997) in order to construct a freely available morphological analyzer: the analyzer was built by automatically generating possible base forms, inflecting them in all possible ways and verifying the results against the existing analyzer.

The current state of the art in Hebrew morphological analysis is based on the *HAMSAH* morphological grammar [138], which is implemented in XFST (Sect. 2.4.3). This grammar was reimplemented in Java (for the rationale behind the reimplementation, see [136]) and is currently being maintained and distributed by the Knowledge Center for Processing Hebrew [80]. The coverage of the system is high, and it is constantly being tested on new documents, in order to extend its lexicon as needed.

It is worth mentioning here that a different morphological grammar was developed for Hebrew, focusing on transcribed *spoken* interactions of children

---

<sup>2</sup>This section is adapted from [135].

and adults [102]. In the context of the CHILDES project [95], corpora of such interactions are being developed for dozens of languages, many of which are also accompanied by morphological annotations. Nir et al. [102] describe such a corpus, transcribed in a way that reflects not only the consonantal distinctions that the standard Hebrew script makes, but also vocalic distinctions that it does not, including the location of the main stress on each word. This transcription makes the morphological grammar harder to develop, but it results in a very low degree of ambiguity. The grammar now has full lexical and rule coverage of the two corpora it is applied to, and more corpora are expected to be analyzed in the near future.

### 2.5.4 Other Languages

Morphological resources for other Semitic languages are almost nonexistent. A few notable exceptions include Biblical Hebrew, for which morphological analyzers are available from several commercial enterprises; Akkadian, for which some morphological analyzers were developed [16, 85, 94]; Syriac, which inspired the development of a new model of computational morphology [88]; and dialectal Arabic [44, 65, 68, 72].

### 2.5.5 Related Applications

Also worth mentioning here are a few works that address other morphology-related tasks. These include a shallow morphological analyzer for Arabic [39] that basically segments word forms to sequences of (at most one) prefix, a stem and (at most one) suffix; a system for identifying the roots of Hebrew and Arabic (possibly inflected) words [40]; various programs for vocalization, or restoring diacritics, in Arabic [66, 97, 100, 118, 139] and in other Semitic languages [73]; determining case endings of Arabic words [69]; and correction of optical character recognizer (OCR) errors [96].

When downstream applications are considered, such as chunking, parsing, or machine translation, the question of *tokenization* gains much importance. Morphological analysis determines the lexeme and the inflections (and, sometimes, also the derivational) morphemes of a surface form; but the way in which a surface form is broken down to its morphemes for the purpose of further processing can have a significant impact on the accuracy of such applications. For example, it is convenient to assume that Arabic and Hebrew prefixes are separate tokens; but what about suffixes? Should there be a distinction between the plural suffixes and the pronominal enclitics of nouns? Several works address these questions, usually in the context of a specific application.

Several works investigate various pre-processing techniques for Arabic, in the context of developing Arabic-to-English statistical machine translation systems [45, 46, 67, 116]. In the reverse direction, [13] and [2] explore the impact of morphological segmentation on English-to-Arabic machine translation. The effect of multiple pre-processing schemes on statistical word alignment for machine translation is explored by Elminger and Habash [47]. And Diab [41] investigates the effect of differently defined POS tagsets (more or less refined) on the task of base phrase chunking (shallow parsing).

## 2.6 Morphological Disambiguation of Semitic Languages

Early attempts at POS tagging and morphological disambiguation of Semitic languages relied on more “traditional” approaches, borrowed directly from the general (i.e., English) POS tagging literature. The first such work is probably [87], who defined a set of 131 POS tags, manually annotated a corpus of 50,000 words and then implemented a tagger that combines statistical and rule-based techniques that performs both segmentation and tag disambiguation. Similarly, [42] use SVM to automatically tokenize, POS-tag, and chunk Arabic texts. To this end, they use a reduced tag set of only 24 tags, with which the reported results are very high. The set of tags is extended to 75 in [41].

For Hebrew, two HMM-based POS taggers were developed. The tagger of [14] is trained on an annotated corpus [80]. The most updated version of the tagger, trained on a treebank of 4,500 sentences, boasts 97.2 % accuracy for segmentation (detection of underlying morphemes, including a possibly assimilated definite article), and 90.8 % accuracy for POS tagging [15]. Adler and Elhadad [1] train an HMM-based POS tagger on a large-scale *unannotated* corpus of 6 million words, the reported accuracy being 92.32 % for POS tagging and 88.5 % for full morphological disambiguation, including finding the correct lexical entry.

As for Amharic, [48] uses condition random fields for POS tagging. As the annotated corpus used for training is extremely small (1,000 words), it is not surprising the accuracy is rather low: 84 % for segmentation, and only 74 % for POS tagging. Two other works use a recently-created 210,000-word annotated corpus [54] to train Amharic POS taggers with a tag set of size 30. Gambäck et al. [55] experiment with HMM, SVM and Maximum Entropy; accuracy ranges between 88 and 95 %, depending on the test corpus. Similarly, [129] investigate various classification techniques, using the same corpus for the same task. The best accuracy, achieved with SVM, is over 86 %, but other classification methods, including conditional random fields and memory-based learning, perform well.

The challenge of morphological disambiguation in Semitic languages, however, as discussed in Sect. 2.3 above, prompted several novel approaches to the task. Many of them are based on the work of [75] and [74], who describe morphological disambiguation of Czech, Estonian, Hungarian, Romanian and Slovene within a single approach. The main idea is to define separate classifiers for each feature of the

morphological analysis (e.g., POS, number, person, tense, case, etc.) The predictions of all the classifiers are then combined with a weighted log-linear model to produce a single, unified analysis. If a morphological analyzer for the language is available, its output is used to constrain the possible analyses predicted by the classifiers.

This approach exactly has been adapted to Arabic by Habash and Rambow [64] and to Hebrew by Shacham and Wintner [119]. Habash and Rambow [64] start with the output of a morphological analyzer [27]. They define ten classifiers, one for each feature of the morphological analysis, namely POS, gender, number, person, voice, aspect, a pronominal enclitic and two classifiers for conjunction and particle proclitics. The classifiers are implemented with SVM, using all the features of the morphological analyses of words within a  $\pm 2$  window of the target word as features. The predictions of the ten classifiers are combined to yield the most likely analysis for each word. The best results are achieved by a rule-based classifier, learned from the training data, that decides when an analysis is “good” based on the predictions of the basic classifiers. The state of the art in Arabic morphological disambiguation is represented by the MADA + TOKAN system [70, 115], which implements these ideas.

Shacham and Wintner [119] basically adapt this approach to Hebrew. They define classifiers for POS, gender, number, person, tense, definiteness, status, prefixes and suffixes, implemented with SNoW [114], using all the features of the morphological analysis in a varying window around the target word as features. They, too, investigate a few methods for combining the results of the classifiers, but the naïve, unweighted combination yields the best results.

A different approach is proposed by Smith et al. [124]. While they also use a morphological analyzer (in the case of Arabic, [27]) to constrain the possible analyses, prediction is done in the source–channel model, where the source is a factored, conditionally-estimated random field [90]. The model is applied to Arabic (and also to Czech and Korean), and the results are competitive with [64] (the same tag set of 139 tags is used).

Recently, [98, 99] proposed an alternative approach to POS tagging of Arabic, which they refer to as *full-word tagging*. Given a large annotated corpus of some 500,000 words, they observe that almost 1,000 different (complex) tags occur in the corpus. They use Memory-based Learning [38] to train a classifier to assign any one of those tags. This is a difficult task: almost one quarter of the tags occur only once in the corpus, so data sparseness is a serious issue. On the other hand, the ambiguity of full word forms is low: only 1.1 analyses per word, on average, with a maximum of 7. This approach results in more accurate disambiguation than any other approach. Furthermore, projecting the complex POS tags to simpler ones, in this case the extra-reduced tagset of [64], results in more accurate “basic” POS tagging than a direct approach that predicts the simpler tags only. Most interestingly, the results show that running either a segmentation classifier or a vocalization one as a pre-process does not improve the accuracy of morphological disambiguation.

Again, the reader is referred to [63] for a full discussion of Arabic morphological disambiguation. For other Semitic languages than the ones described here, unfortunately, we are unaware of any works addressing morphological disambiguation.

## 2.7 Future Directions

The discussion above establishes the inherent difficulty of morphological processing with Semitic languages, as one instance of languages with rich and complex morphology. Having said that, it is clear that with a focused effort, contemporary computational technology is sufficient for tackling the difficulties. As should be clear from Sect. 2.5, the two Semitic languages that benefitted from most attention, namely MSA and Hebrew, boast excellent computational morphological analyzers and generators. Similarly, Sect. 2.6 shows that morphological disambiguation of these two languages can be done with high accuracy, nearing the accuracy of disambiguation with European languages.

However, for the less-studied languages, including Amharic, Maltese and others, much work is still needed in order to produce tools of similar precision. Resembling the situation in Arabic and Hebrew, this effort should focus on two fronts: development of formal, computationally-implementable sets of rules that describe the morphology of the language in question; and collection and annotation of corpora from which morphological disambiguation modules can be trained.

As for future technological improvements, we note that “pipeline” approaches, whereby the input text is fed, in sequence, to a tokenizer, a morphological analyzer, a morphological disambiguation module and then a parser, have probably reached a ceiling, and the stage is ripe for more elaborate, unified approaches. Several works indeed explore such possibilities, focusing in particular on joint morphological disambiguation and parsing [35, 59, 92, 132]. We defer an extensive discussion of these (and other) approaches to the next Chapter on parsing.

**Acknowledgements** I am tremendously grateful to Nizar Habash for his help and advice; it would have been hard to complete this chapter without them. All errors and misconceptions are, of course, solely my own.

## References

1. Adler M, Elhadad M (2006) An unsupervised morpheme-based HMM for Hebrew morphological disambiguation. In: Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the Association for Computational Linguistics, Sydney. Association for Computational Linguistics, pp 665–672. <http://www.aclweb.org/anthology/P/P06/P06-1084>
2. Al-Haj H, Lavie A (2010) The impact of Arabic morphological segmentation on broad-coverage English-to-Arabic statistical machine translation. In: Proceedings of the conference of the Association for Machine Translation in the Americas (AMTA), Denver
3. Alkuhlani S, Habash N (2011) A corpus for modeling morpho-syntactic agreement in Arabic: gender, number and rationality. In: Proceedings of the 49th annual meeting of the Association for Computational Linguistics: human language technologies, Portland. Association for Computational Linguistics, pp 357–362. <http://www.aclweb.org/anthology/P11-2062>

4. Al-Shalabi R, Evans M (1998) A computational morphology system for Arabic. In: Rosner M (ed) Proceedings of the workshop on computational approaches to Semitic languages, COLING-ACL'98, Montreal, pp 66–72
5. Al-Sughaiyer IA, Al-Kharashi IA (2004) Arabic morphological analysis techniques: a comprehensive survey. *J Am Soc Inf Sci Technol* 55(3):189–213
6. Altantawy M, Habash N, Rambow O, Saleh I (2010) Morphological analysis and generation of Arabic nouns: a morphemic functional approach. In: Proceedings of the seventh international conference on language resources and evaluation (LREC), Valletta
7. Altantawy M, Habash N, Rambow O (2011) Fast yet rich morphological analysis. In: Proceedings of the 9th international workshop on finite-state methods and natural language processing (FSMNLP 2011), Blois
8. Amsalu S, Gibbon D (2005) A complete finite-state model for Amharic morphographemics. In: Yli-Jyrä A, Karttunen L, Karhumäki J (eds) FSMNLP. Lecture notes in computer science, vol 4002. Springer, Berlin/New York, pp 283–284
9. Amsalu S, Gibbon D (2005) Finite state morphology of Amharic. In: Proceedings of RANLP, Borovets, pp 47–51
10. Amtrup JW (2003) Morphology in machine translation systems: efficient integration of finite state transducers and feature structure descriptions. *Mach Transl* 18(3):217–238. doi:<http://dx.doi.org/10.1007/s10590-004-2476-5>
11. Argaw AA, Asker L (2007) An Amharic stemmer: reducing words to their citation forms. In: Proceedings of the ACL-2007 workshop on computational approaches to Semitic languages, Prague
12. Audebert C, Gaubert C, Jaccarini A (2009) Minimal resources for Arabic parsing: an interactive method for the construction of evolutive automata. In: Choukri K, Maegaard B (eds) Proceedings of the second international conference on Arabic language resources and tools, The MEDAR Consortium, Cairo
13. Badr I, Zbib R, Glass J (2008) Segmentation for English-to-Arabic statistical machine translation. In: Proceedings of ACL-08: HLT, short papers, Columbus. Association for Computational Linguistics, pp 153–156. <http://www.aclweb.org/anthology/P/P08/P08-2039>
14. Bar-Haim R, Sima'an K, Winter Y (2005) Choosing an optimal architecture for segmentation and POS-tagging of Modern Hebrew. In: Proceedings of the ACL workshop on computational approaches to Semitic languages, Ann Arbor. Association for Computational Linguistics, pp 39–46, <http://www.aclweb.org/anthology/W/W05/W05-0706>
15. Bar-haim R, Sima'an K, Winter Y (2008) Part-of-speech tagging of Modern Hebrew text. *Nat Lang Eng* 14(2):223–251
16. Barthélémy F (1998) A morphological analyzer for Akkadian verbal forms with a model of phonetic transformations. In: Proceedings of the Coling-ACL 1998 workshop on computational approaches to Semitic languages, Montreal, pp 73–81
17. Beesley KR (1996) Arabic finite-state morphological analysis and generation. In: Proceedings of COLING-96, the 16th international conference on computational linguistics, Copenhagen
18. Beesley KR (1998) Arabic morphological analysis on the internet. In: Proceedings of the 6th international conference and exhibition on multi-lingual computing, Cambridge
19. Beesley KR (1998) Arabic morphology using only finite-state operations. In: Rosner M (ed) Proceedings of the workshop on computational approaches to Semitic languages, COLING-ACL'98, Montreal, pp 50–57
20. Beesley KR (1998) Constraining separated morphotactic dependencies in finite-state grammars. In: FSMNLP-98, Bilkent, pp 118–127
21. Beesley KR, Karttunen L (2000) Finite-state non-concatenative morphotactics. In: Proceedings of the fifth workshop of the ACL special interest group in computational phonology, SIGPHON-2000, Luxembourg
22. Beesley KR, Karttunen L (2003) Finite-state morphology: xerox tools and techniques. CSLI, Stanford
23. Belguith LH, Aloulou C, Ben Hamadou A (2008) MASPAR: De la segmentation à l'analyse syntaxique de textes arabes. *Rev Inf Interact Intell* 13(2):9–36

24. Bentur E, Angel A, Segev D (1992) Computerized analysis of Hebrew words. *Hebrew Linguist* 36:33–38. (in Hebrew)
25. Berri J, Zidoum H, Atif Y (2001) Web-based Arabic morphological analyzer. In: Gelbukh A (ed) CICLing 2001. Lecture notes in computer science, vol 2004. Springer, Berlin, pp 389–400
26. Brants T (2000) TnT: a statistical part-of-speech tagger. In: Proceedings of the sixth conference on applied natural language processing, Seattle. Association for Computational Linguistics, pp 224–231. doi:10.3115/974147.974178, <http://www.aclweb.org/anthology/A00-1031>
27. Buckwalter T (2004) Buckwalter Arabic morphological analyzer version 2.0. Linguistic Data Consortium, Philadelphia
28. Buckwalter T (2004) Issues in Arabic orthography and morphology analysis. In: Farghaly A, Megerdoomian K (eds) COLING 2004 computational approaches to Arabic script-based languages, COLING, Geneva, pp 31–34
29. Choueka Y (1966) Computers and grammar: mechanical analysis of Hebrew verbs. In: Proceedings of the annual conference of the Israeli Association for Information Processing, Rehovot, pp 49–66. (in Hebrew)
30. Choueka Y (1972) Fast searching and retrieval techniques for large dictionaries and concordances. *Heb Comput Linguist* 6:12–32. (in Hebrew)
31. Choueka Y (1980) Computerized full-text retrieval systems and research in the humanities: the Responsa project. *Comput Humanit* 14:153–169
32. Choueka Y (1990) MLIM – a system for full, exact, on-line grammatical analysis of Modern Hebrew. In: Eizenberg Y (ed) Proceedings of the annual conference on computers in education, Tel Aviv, p 63. (in Hebrew)
33. Choueka Y (1993) Response to “computerized analysis of Hebrew words”. *Heb Linguist* 37:87. (in Hebrew)
34. Cohen D (1970) *Essai d'une analyse automatique de l'arabe*. In: Etudes de linguistique sémitique et arabe, De Gruyter, Germany, pp 49–78
35. Cohen SB, Smith NA (2007) Joint morphological and syntactic disambiguation. In: Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL), Prague. Association for Computational Linguistics, pp 208–217. <http://www.aclweb.org/anthology/D/D07/D07-1022>
36. Cohen-Sygal Y, Wintner S (2006) Finite-state registered automata for non-concatenative morphology. *Comput Linguist* 32(1):49–82
37. Collins M (2002) Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In: Proceedings of the ACL-02 conference on empirical methods in natural language processing, EMNLP '02, Philadelphia, Vol 10. Association for Computational Linguistics, pp 1–8. doi:<http://dx.doi.org/10.3115/1118693.1118694>
38. Daelemans W, van den Bosch A (2005) Memory-based language processing. Studies in natural language processing. Cambridge University Press, Cambridge
39. Darwisch K (2002) Building a shallow Arabic morphological analyzer in one day. In: Rosner M, Wintner S (eds) ACL'02 workshop on computational approaches to Semitic languages , Philadelphia, pp 47–54
40. Daya E, Roth D, Wintner S (2007) Learning to identify Semitic roots. In: Soudi A, Neumann G, van den Bosch A (eds) Arabic computational morphology: knowledge-based and empirical methods, text, speech and language technology, vol 38. Springer, Dordrecht, pp 143–158
41. Diab M (2007) Improved Arabic base phrase chunking with a new enriched POS tag set. In: Proceedings of the 2007 workshop on computational approaches to Semitic languages: common issues and resources, Prague, pp 89–96. <http://www.aclweb.org/anthology/W/W07/W07-0812>
42. Diab M, Hacioglu K, Jurafsky D (2004) Automatic tagging of Arabic text: from raw text to base phrase chunks. In: Proceedings of HLT-NAACL 2004, Boston

43. Dicky J, Farghaly A (2003) Roots and patterns vs. stems plus grammar-lexis specifications: on what basis should a multilingual lexical database centered on Arabic be built. In: Proceedings of the MT-Summit IX workshop on machine translation for Semitic languages, New Orleans, pp 1–8
44. Duh K, Kirchhoff K (2005) POS tagging of dialectal Arabic: a minimally supervised approach. In: Proceedings of the ACL workshop on computational approaches to Semitic languages, Ann Arbor. Association for Computational Linguistics, pp 55–62. <http://www.aclweb.org/anthology/W/W05/W05-0708>
45. El Kholy A, Habash N (2010) Orthographic and morphological processing for English-Arabic statistical machine translation. In: In actes de traitement automatique des langues naturelles (TALN), Montréal
46. El Kholy A, Habash N (2010) Techniques for Arabic morphological detokenization and orthographic denormalization. In: Proceedings of LREC-2010, Valletta (Malta)
47. Elming J, Habash N (2007) Combination of statistical word alignments based on multiple preprocessing schemes. In: Human language technologies 2007: the conference of the North American chapter of the Association for Computational Linguistics, Companion Volume, Short Papers, Prague, pp 25–28. <http://www.aclweb.org/anthology/N/N07/N07-2007>
48. Fissaha Adafre S (2005) Part of speech tagging for Amharic using conditional random fields. In: Proceedings of the ACL workshop on computational approaches to Semitic languages, Ann Arbor. Association for Computational Linguistics, pp 47–54. <http://www.aclweb.org/anthology/W/W05/W05-0707>
49. Fissaha S, Haller J (2003) Amharic verb lexicon in the context of machine translation. In: Proceedings of the TALN workshop on natural language processing of minority languages, Batz-sur-Mer
50. Forsberg M (2007) Three tools for language processing: BNF converter, functional morphology, and extract. PhD thesis, Göteborg University and Chalmers University of Technology
51. Forsberg M, Ranta A (2004) Functional morphology. In: Proceedings of the ninth ACM SIGPLAN international conference on functional programming (ICFP'04), Snowbird. ACM, New York, pp 213–223
52. Fraenkel AS (1976) All about the Responsa retrieval project – what you always wanted to know but were afraid to ask. *Jurimetrics* J 16(3):149–156
53. Gadish R (ed) (2001) Klalei ha-Ktiv Hasar ha-Niqqud, 4th edn. Academy for the Hebrew Language, Jerusalem. (in Hebrew)
54. Gambäck B, Olsson F, Argaw AA, Asker L (2009) An Amharic corpus for machine learning. In: Proceedings of the 6th world congress of African linguistics, Cologne
55. Gambäck B, Olsson F, Argaw AA, Asker L (2009) Methods for Amharic part-of-speech tagging. In: Proceedings of the first workshop on language technologies for African languages, Athen. Association for Computational Linguistics, Stroudsburg, pp 104–111
56. Gasser M (2009) Semitic morphological analysis and generation using finite state transducers with feature structures. In: Proceedings of the 12th conference of the European chapter of the ACL (EACL 2009), Athens. Association for Computational Linguistics, pp 309–317. <http://www.aclweb.org/anthology/E09-1036>
57. Gasser M (2011) HornMorpho: a system for morphological processing of Amharic, Oromo, and Tigrinya. *Bibliotheca Alexandrina*, Alexandria, pp 94–99
58. Giménez J, Márquez L (2004) SVMTool: a general POS tagger generator based on support vector machines. In: Proceedings of 4th international conference on language resources and evaluation (LREC), Lisbon, pp 43–46
59. Goldberg Y, Tsarfaty R (2008) A single generative model for joint morphological segmentation and syntactic parsing. In: Proceedings of ACL-08: HLT, Columbus. Association for Computational Linguistics, pp 371–379. <http://www.aclweb.org/anthology/P/P08/P08-1043>
60. Goldstein L (1991) Generation and inflection of the possession inflection of Hebrew nouns. Master's thesis, Technion, Haifa (in Hebrew)
61. Habash N (2004) Large scale lexeme based arabic morphological generation. In: Proceedings of traitement automatique du langage naturel (TALN-04), Fez

62. Habash N (2007) Arabic morphological representations for machine translation. In: van den Bosch A, Soudi A (eds) Arabic computational morphology: knowledge-based and empirical methods. Springer, Dordrecht
63. Habash N (2010) Introduction to Arabic natural language processing. Synthesis lectures on human language technologies. Morgan & Claypool, San Rafael. doi:<http://dx.doi.org/10.2200/S00277ED1V01Y201008HLT010>
64. Habash N, Rambow O (2005) Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In: Proceedings of the 43rd annual meeting of the Association for Computational Linguistics (ACL'05), University of Michigan. Association for Computational Linguistics, Ann Arbor, pp 573–580. <http://www.aclweb.org/anthology/P/P05/P05-1071>
65. Habash N, Rambow O (2006) MAGEAD: a morphological analyzer and generator for the Arabic dialects. In: Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the Association for Computational Linguistics, Sydney. Association for Computational Linguistics, pp 681–688. <http://www.aclweb.org/anthology/P/P06/P06-1086>
66. Habash N, Rambow O (2007) Arabic diacritization through full morphological tagging. In: Human language technologies 2007: the conference of the North American chapter of the association for computational linguistics; Companion Volume, Short Papers, Rochester. Association for Computational Linguistics, pp 53–56. <http://www.aclweb.org/anthology/N/N07/N07-2014>
67. Habash N, Sadat F (2006) Arabic preprocessing schemes for statistical machine translation. In: Moore RC, Bilmes JA, Chu-Carroll J, Sanderson M (eds) HLT-NAACL, New York. The Association for Computational Linguistics
68. Habash N, Rambow O, Kiraz G (2005) Morphological analysis and generation for Arabic dialects. In: Proceedings of the ACL workshop on computational approaches to Semitic languages, Ann Arbor. Association for Computational Linguistics, pp 17–24. <http://www.aclweb.org/anthology/W/W05/W05-0703>
69. Habash N, Gabbard R, Rambow O, Kulick S, Marcus M (2007) Determining case in Arabic: learning complex linguistic behavior requires complex linguistic features. In: Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL 2007), Prague
70. Habash N, Rambow O, Roth R (2009) MADA+TOKAN: a toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In: Choukri K, Maegaard B (eds) Proceedings of the second international conference on Arabic language resources and tools, Cairo, The MEDAR Consortium
71. Habash N, Diab M, Rabnow O (2012) Conventional orthography for Dialectal Arabic. In: Proceedings of the language resources and evaluation conference (LREC), Istanbul
72. Habash N, Eskander R, Hawwari A (2012) A morphological analyzer for Egyptian Arabic. In: Proceedings of the twelfth meeting of the special interest group on computational morphology and phonology, Montréal. Association for Computational Linguistics, pp 1–9. <http://www.aclweb.org/anthology/W12-2301>
73. Haertel RA, McClanahan P, Ringger EK (2010) Automatic diacritization for low-resource languages using a hybrid word and consonant CMM. In: Human language technologies: the 2010 annual conference of the north american chapter of the Association for Computational Linguistics, HLT '10, Stroudsburg. Association for Computational Linguistics, pp 519–527
74. Hajic J (2000) Morphological tagging: Data vs. dictionaries. In: Proceedings of ANLP-NAACL conference, Seattle, pp 94–101
75. Hajic J, Hladká B (1998) Tagging inflective languages: prediction of morphological categories for a rich, structured tagset. In: Proceedings of the 36th annual meeting of the Association for Computational Linguistics and 17th international conference on computational linguistics, Montreal. Association for Computational Linguistics, Stroudsburg, pp 483–490. doi:<http://dx.doi.org/10.3115/980845.980927>, <http://dx.doi.org/10.3115/980845.980927>

76. Harley HB (2006) English words: a linguistic introduction. The language library. Wiley-Blackwell, Malden
77. Hetzron R (ed) (1997) The Semitic languages. Routledge, London/New York
78. Hulden M (2009) Foma: a finite-state compiler and library. In: Proceedings of the demonstrations session at EACL 2009, Athens. Association for Computational Linguistics, pp 29–32. <http://www.aclweb.org/anthology/E09-2008>
79. Hulden M (2009) Revisiting multi-tape automata for Semitic morphological analysis and generation. In: Proceedings of the EACL 2009 workshop on computational approaches to Semitic languages, Athens. Association for Computational Linguistics, pp 19–26. <http://www.aclweb.org/anthology/W09-0803>
80. Itai A, Wintner S (2008) Language resources for Hebrew. *Lang Resour Eval* 42(1):75–98
81. Johnson CD (1972) Formal aspects of phonological description. Mouton, The Hague
82. Kammoun NC, Belguith LH, Mesfar S (2010) Arabic POS tagging based on NooJ grammars and the Arabic morphological analyzer MORPH2. In: Proceedings of NooJ 2010, Komotini
83. Kaplan RM, Kay M (1994) Regular models of phonological rule systems. *Comput Linguist* 20(3):331–378
84. Karttunen L, Beesley KR (2001) A short history of two-level morphology. In: Talk given at the ESSLLI workshop on finite state methods in natural language processing. <http://www.helsinki.fi/esslli/evening/20years/twol-history.html>
85. Kataja L, Koskenniemi K (1988) Finite-state description of Semitic morphology: a case study of ancient Akkadian. In: COLING, Budapest, pp 313–315
86. Kay M (1987) Nonconcatenative finite-state morphology. In: Proceedings of the third conference of the European chapter of the Association for Computational Linguistics, Copenhagen, pp 2–10
87. Khoja S (2001) APT: Arabic part-of-speech tagger. In: Proceedings of the student workshop at the second meeting of the North American chapter of the Association for Computational Linguistics (NAACL2001), Pittsburgh
88. Kiraz GA (2000) Multitiered nonlinear morphology using multitape finite automata: a case study on Syriac and Arabic. *Comput Linguist* 26(1):77–105
89. Koskenniemi K (1983) Two-level morphology: a general computational model for word-form recognition and production. The Department of General Linguistics, University of Helsinki
90. Lafferty J, McCallum A, Pereira F (2001) Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of the 18th international conference on machine learning (ICML-01), Williamstown, pp 282–289
91. Lavie A, Itai A, Ornan U, Rimon M (1988) On the applicability of two-level morphology to the inflection of Hebrew verbs. In: Proceedings of the international conference of the ALLC, Jerusalem
92. Lee J, Naradowsky J, Smith DA (2011) A discriminative model for joint morphological disambiguation and dependency parsing. In: Proceedings of the 49th annual meeting of the Association for Computational Linguistics: human language technologies, Portland. Association for Computational Linguistics, pp 885–894. <http://www.aclweb.org/anthology/P11-1089>
93. Maamouri M, Bies A, Buckwalter T, Mekki W (2004) The Penn Arabic treebank: building a large-scale annotated Arabic corpus. In: NEMLAR conference on Arabic language resources and tools, Cairo, pp 102–109
94. Macke A (2002) Parsing Akkadian verbs with Prolog. In: Proceedings of the ACL-02 workshop on computational approaches to Semitic languages, Philadelphia
95. MacWhinney B (2000) The CHILDES project: tools for analyzing talk, 3rd edn. Lawrence Erlbaum Associates, Mahwah
96. Magdy W, Darwish K (2006) Arabic OCR error correction using character segment correction, language modeling, and shallow morphology. In: Proceedings of the 2006 conference on empirical methods in natural language processing, Sydney. Association for Computational Linguistics, pp 408–414. <http://www.aclweb.org/anthology/W/W06/W06-1648>

97. Mohamed E, Kübler S (2009) Diacritization for real-world Arabic texts. In: Proceedings of the international conference RANLP-2009, pp 251–257. <http://www.aclweb.org/anthology/R09-1047>
98. Mohamed E, Kübler S (2010) Arabic part of speech tagging. In: Proceedings of the seventh conference on international language resources and evaluation (LREC'10), European Language Resources Association (ELRA), Valletta
99. Mohamed E, Kübler S (2010) Is Arabic part of speech tagging feasible without word segmentation? In: Human language technologies: the 2010 annual conference of the North American chapter of the Association for Computational Linguistics, HLT'10, Los Angeles. Association for Computational Linguistics, Stroudsburg, pp 705–708. <http://dl.acm.org/citation.cfm?id=1857999.1858104>
100. Nelken R, Shieber SM (2005) Arabic diacritization using weighted finite-state transducers. In: Proceedings of the ACL workshop on computational approaches to Semitic languages, Ann Arbor. Association for Computational Linguistics, pp 79–86. <http://www.aclweb.org/anthology/W/W05/W05-0711>
101. Netzer Y, Adler M, Gabay D, Elhadad M (2007) Can you tag the modal? You should. In: Proceedings of the ACL-2007 workshop on computational approaches to Semitic languages, Prague
102. Nir B, MacWhinney B, Wintner S (2010) A morphologically-analyzed CHILDES corpus of Hebrew. In: Proceedings of the seventh conference on international language resources and evaluation (LREC'10), Valletta. European Language Resources Association (ELRA), pp 1487–1490
103. Ornan U (1985) Indexes and concordances in a phonemic Hebrew script. In: Proceedings of the ninth world congress of Jewish studies, World Union of Jewish Studies, Jerusalem, pp 101–108. (in Hebrew)
104. Ornan U (1985) Vocalization by a computer: a linguistic lesson. In: Luria BZ (ed) Avraham Even-Shoshan book, Kiryat-Sefer, Jerusalem, pp 67–76. (in Hebrew)
105. Ornan U (1986) Phonemic script: a central vehicle for processing natural language – the case of Hebrew. Technical report 88.181, IBM Research Center, Haifa
106. Ornan U (1987) Computer processing of Hebrew texts based on an unambiguous script. *Mishpatim* 17(2):15–24. (in Hebrew)
107. Ornan U, Katz M (1995) A new program for Hebrew index based on the Phonemic Script. Technical report LCL 94-7, Laboratory for Computational Linguistics, Technion, Haifa
108. Ornan U, Kazatski W (1986) Analysis and synthesis processes in Hebrew morphology. In: Proceedings of the 21 national data processing conference, Israel. (in Hebrew)
109. Owens J (1997) The Arabic grammatical tradition. In: Hetzron R (ed) *The Semitic languages*. Routledge, London/New York, chap 3, pp 46–58
110. Pinkas G (1985) A linguistic system for information retrieval. *Maase Hoshev* 12:10–16. (in Hebrew)
111. Ratnaparkhi A (1996) A maximum entropy model for part-of-speech tagging. In: Brill E, Church K (eds) *Proceedings of the conference on empirical methods in natural language processing*, Copenhagen. Association for Computational Linguistics, pp 133–142
112. Roark B, Sproat RW (2007) Computational approaches to morphology and syntax. Oxford University Press, New York
113. Roche E, Schabes Y (eds) (1997) Finite-state language processing. Language, speech and communication. MIT, Cambridge
114. Roth D (1998) Learning to resolve natural language ambiguities: a unified approach. In: Proceedings of AAAI-98 and IAAI-98, Madison, pp 806–813
115. Roth R, Rambow O, Habash N, Diab M, Rudin C (2008) Arabic morphological tagging, diacritization, and lemmatization using lexeme models and feature ranking. In: Proceedings of ACL-08: HLT, Short Papers, Columbus. Association for Computational Linguistics, pp 117–120. <http://www.aclweb.org/anthology/P/P08/P08-2030>

116. Sadat F, Habash N (2006) Combination of Arabic preprocessing schemes for statistical machine translation. In: Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the Association for Computational Linguistics, Sydney. Association for Computational Linguistics, pp 1–8. <http://www.aclweb.org/anthology/P/P06/P06-1001>
117. Schippers A (1997) The Hebrew grammatical tradition. In: Hetzron R (ed) *The Semitic languages*. Routledge, London/New York, chap 4, pp 59–65
118. Shaalan K, Abo Bakr HM, Ziedan I (2009) A hybrid approach for building Arabic diacritizer. In: Proceedings of the EACL 2009 workshop on computational approaches to Semitic languages, Semitic'09, Athens. Association for Computational Linguistics, Stroudsburg, pp 27–35
119. Shacham D, Wintner S (2007) Morphological disambiguation of Hebrew: a case study in classifier combination. In: Proceedings of EMNLP-CoNLL 2007, the conference on empirical methods in natural language processing and the conference on computational natural language learning, Prague. Association for Computational Linguistics
120. Shany-Klein M (1990) Generation and analysis of Segolate noun inflection in Hebrew. Master's thesis, Technion, Haifa. (in Hebrew)
121. Shany-Klein M, Ornan U (1992) Analysis and generation of Hebrew Segolate nouns. In: Ornan U, Arieli G, Doron E (eds) *Hebrew computational linguistics*. Ministry of Science and Technology, Jerusalem, chap 4, pp 39–51. (in Hebrew)
122. Shapira M, Choueka Y (1964) Mechanographic analysis of Hebrew morphology: possibilities and achievements. *Leshonenu* 28(4):354–372. (in Hebrew)
123. Silberstein M (2004) NooJ: an object-oriented approach. In: Muller C, Royauté J, Silberstein M (eds) *INTEX pour la linguistique et le traitement automatique des Langues*, cahiers de la MSH Ledoux, Presses Universitaires de Franche-Comté, pp 359–369
124. Smith NA, Smith DA, Tromble RW (2005) Context-based morphological disambiguation with random fields. In: Proceedings of human language technology conference and conference on empirical methods in natural language processing, Vancouver. Association for Computational Linguistics, Morristown, pp 475–482
125. Smrž O (2007) ElixirFM: implementation of functional Arabic morphology. In: Proceedings of the 2007 workshop on computational approaches to Semitic languages: common issues and resources, Prague. Association for Computational Linguistics, Stroudsburg, pp 1–8
126. Smrž O (2007) Functional Arabic morphology. *Prague Bull Math Linguist* 88:5–30
127. Soudi A, van den Bosch A, Neumann G (2007) Arabic computational morphology: knowledge-based and empirical methods. Springer, Dordrecht
128. Sproat RW (1992) Morphology and computation. MIT, Cambridge
129. Tachbelie MY, Abate ST, Besacier L (2011) Part-of-speech tagging for under-resourced and morphologically rich languages – the case of Amharic. *Bibliotheca Alexandrina*, Alexandria, pp 50–55. <http://aflat.org/files/HLTD201109.pdf>
130. Toutanova K, Manning CD (2000) Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In: Proceedings of the 2000 joint SIGDAT conference on empirical methods in natural language processing and very large corpora, Morristown. Association for Computational Linguistics, pp 63–70. doi:<http://dx.doi.org/10.3115/1117794.1117802>
131. Toutanova K, Klein D, Manning CD, Singer Y (2003) Feature-rich part-of-speech tagging with a cyclic dependency network. In: NAACL '03: Proceedings of the 2003 conference of the North American chapter of the Association for Computational Linguistics on human language technology, Edmonton. Association for Computational Linguistics, Morristown, pp 173–180. doi:<http://dx.doi.org/10.3115/1073445.1073478>
132. Tsarfaty R (2006) Integrated morphological and syntactic disambiguation for Modern Hebrew. In: Proceedings of the COLING/ACL 2006 student research workshop, Sydney. Association for Computational Linguistics, pp 49–54. <http://www.aclweb.org/anthology/P/P06/P06-3009>

133. Tsuruoka Y, Tsujii J (2005) Bidirectional inference with the easiest-first strategy for tagging sequence data. In: Proceedings of the conference on human language technology and empirical methods in natural language processing, HLT'05, Vancouver. Association for Computational Linguistics, Stroudsburg, pp 467–474. doi:<http://dx.doi.org/10.3115/1220575.1220634>
134. Tsuruoka Y, Tateishi Y, Kim JD, Ohta T, McNaught J, Ananiadou S, Tsujii J (2005) Developing a robust part-of-speech tagger for biomedical text. In: Bozanis P, Houstis EN (eds) Advances in informatics. LNCS, vol 3746. Springer, Berlin/Heidelberg, chap 36, pp 382–392. doi:10.1007/11573036\_36, [http://dx.doi.org/10.1007/11573036\\_36](http://dx.doi.org/10.1007/11573036_36)
135. Wintner S (2004) Hebrew computational linguistics: past and future. Artif Intell Rev 21(2):113–138. doi:<http://dx.doi.org/10.1023/B:AIRE.0000020865.73561.bc>
136. Wintner S (2008) Strengths and weaknesses of finite-state technology: a case study in morphological grammar development. Nat Lang Eng 14(4):457–469. doi:<http://dx.doi.org/10.1017/S1351324907004676>
137. Wintner S (2009) Language resources for Semitic languages: challenges and solutions. In: Nirenburg S (ed) Language engineering for lesser-studied languages. IOS, Amsterdam, pp 277–290
138. Yona S, Wintner S (2008) A finite-state morphological grammar of Hebrew. Nat Lang Eng 14(2):173–190
139. Zitouni I, Sorensen JS, Sarikaya R (2006) Maximum entropy based restoration of Arabic diacritics. In: Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the Association for Computational Linguistics, Sydney. Association for Computational Linguistics, pp 577–584. <http://www.aclweb.org/anthology/P/P06/P06-1073>
140. Zwicky AM, Pullum GK (1983) Cliticization vs. inflection: English *n't*. Language 59(3): 502–513

# Chapter 3

## Syntax and Parsing of Semitic Languages

Reut Tsarfaty

### 3.1 Introduction

Parsing, the task of automatically analyzing the syntactic structure of natural language sentences, is a core task in natural language processing (NLP). Syntactic analysis unravels the way in which words combine to form phrases and sentences. The syntactic analysis of sentences is important from a linguistic point of view, as syntactic structures map language-specific phenomena (the form of words, their order, their grouping) onto an abstract representation of meaning-bearing elements such as subject, predicate, etc. It is also important from a technological viewpoint, as it helps to recover the notions of “who did what to whom” from unstructured texts.

The best parsing systems to date are supervised, data-driven and statistical. Many state-of-the-art statistical parsers have been tuned to, and excel at, parsing English. Such parsers will not necessarily perform as well when trained on data from a language with different characteristics [106, 108], and indeed, existing parsing technologies do not lend themselves to parsing Semitic texts so easily.

Semitic languages such as Arabic, Hebrew, Amharic or Maltese belong to the Afro-Asian family, and they are assumed to be descendants of the same ancient ancestor, called the *Proto-Semitic*. The linguistic structure of Semitic languages is very different from that of English. Most of the Semitic languages are written right-to-left and each language utilizes its own writing systems. These languages introduce vocalization patterns and orthography that are particular to the Semitic family (Chap. 1), and manifest rich morphology and word-order patterns that are different from that of English (this chapter). Even though Semitic languages have an important typological and sociological status, they have been under-studied in

---

R. Tsarfaty (✉)

Department of Linguistics and Philology, Uppsala University, Box 635, 75126 Uppsala, Sweden  
e-mail: [tsarfaty@stp.lingfil.uu.se](mailto:tsarfaty@stp.lingfil.uu.se)

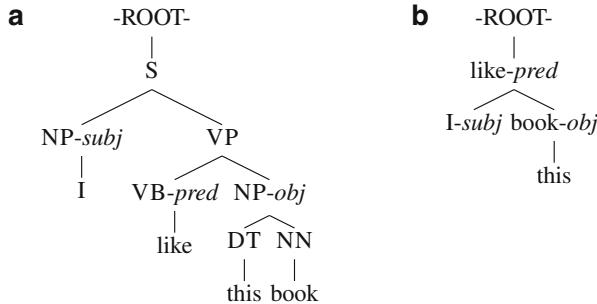
terms of language technology. With availability of annotated corpora for Hebrew and Arabic and recent advances made in parsing technology for morphologically rich languages [106], the time is ripe for an illuminating discussion of effective techniques for parsing Semitic languages.

This chapter takes up the opportunity to look at statistical parsing technology from the Semitic perspective. We ask questions such as: How can we build parsing systems that utilize the massive technological advances made in parsing English while doing justice to the linguistic phenomena exhibited by Semitic languages? How well do these models perform relative to general-purpose ones in real-world parsing scenarios? We cannot hope to do justice here to the vast literature on parsing technology and theoretical studies on Semitic syntax, but if you follow this chapter through, we expect that you will be able to successfully conduct (at least) the following: (i) put together a baseline parsing system that takes into account the Semitic challenges, (ii) effectively employ modeling techniques that proved useful in parsing Semitic texts, and (iii) use a range of evaluation metrics in order to get a fair grasp of parse quality and system bottlenecks. Many of the presented systems are downloadable from the web and may be used out of the box. If you do not intend to develop a parsing architecture yourself, this chapter will endow you with a deeper understanding that will help you set up your data, your choice of parameters, the feature design, etc., so as to avoid the common pitfalls of porting general-purpose parsing systems across languages.

The remainder of this chapter is organized as follows.<sup>1</sup> Section 3.1 first charts the parsing world in terms of representations, models and algorithms (Sect. 3.1.1). It then describes Semitic orthographic, morphological and syntactic phenomena (Sect. 3.1.2). It finally outlines the overarching challenges in parsing Semitic languages: the architectural challenge, the modeling challenge, and the lexical challenge (Sect. 3.1.3). In Sect. 3.2 we present in detail a case study from the constituency-based parsing paradigm. After formally defining the basic generative parsing system (Sect. 3.2.1), we present variations that address the architectural challenge (Sect. 3.2.2), the modeling challenge (Sect. 3.2.3), and the lexical challenge (Sect. 3.2.4) for Semitic languages. Section 3.3 surveys the main parsing results that have been reported for Semitic languages so far. Section 3.4 summarizes the current state-of-affairs and concludes with open questions for further research.

---

<sup>1</sup>This chapter assumes knowledge of set theory and probability theory, and familiarity with formal language theory. Good references would be [71] or [55].



**Fig. 3.1** (a) A phrase-structure tree for (1). (b) A dependency tree for (1). Each *emphasized* grammatical function label describes the relation indicated by the dominating arc

### 3.1.1 Parsing Systems

#### Syntactic Analysis

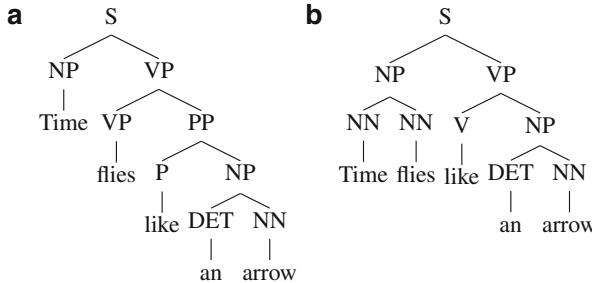
A parsing system is a computer program that takes a sentence in a natural language as input and provides a representation of its human perceived syntactic analysis as output. For example, the syntactic analysis of sentence (1) below should identify the syntactic entities “I” and “this book”, and formally represent the relations between them, that is, that “I” is the *subject* of “like”, and that “this book” is its *object*.

(1) I like this book.

The syntactic analysis of a sentence is formally represented as a connected graph which represents entities as nodes and relations as arcs. These representations build on formal linguistic frameworks that have been developed by linguists over the last century.

One way to represent this information is by means of *constituency structures* [11, 27]. Formally, constituency structures are linearly-ordered trees in which internal nodes are labeled with types from a finite set of categories. Phrase-structure trees are constituency trees labeled with phrase types – Noun Phrase (NP), Verb Phrase (VP), Sentence (S) etc. The phrase-structure tree for the sentence in Example (1) is illustrated in Fig. 3.1a. In some linguistic traditions it is common to derive the grammatical function of an element from its tree position [27]. For instance, one can identify the leftmost NP under S (“I”) as the *subject* of the sentence, and the rightmost NP daughter of a VP (“this book”) as its *object*.

Instead of deriving grammatical relations from positions, it is possible to represent them explicitly by means of *dependency structures*, which follow on the rich linguistic tradition of [98]. A dependency structure is composed of binary arcs, each arc connects a pair of words. When the arc is labeled, the label defines the type of grammatical relation between the words. A labeled dependency structure delivers



**Fig. 3.2** Syntactic ambiguity in the sentence “Time flies like an arrow”

an explicit representation of the grammatical relations that define the *argument-structure*.<sup>2</sup> A dependency structure for sentence (1) is illustrated in Fig. 3.1b.

Formally, we treat parsing as a *structure prediction* task, where  $\mathcal{X}$  is a set of sentences in a language and  $\mathcal{Y}$  is a set of parse-tree representations of sentences in the language. A parsing system implements a prediction function  $h$  from sentences to parse trees, that is:

$$h : \mathcal{X} \rightarrow \mathcal{Y} \quad (3.1)$$

Different parsing architectures can be thought of as different instantiations of  $h$ . For example, if  $\mathcal{X}$  is the space of sentences in English and  $\mathcal{Y}$  is the space of constituency trees with English words in their leaves, then (3.1) defines a constituency-based parser for English. If  $\mathcal{X}$  is a set of sentences in French and  $\mathcal{Y}$  is the set of dependency trees where internal nodes are French words, then (3.1) defines a dependency-based parser for French.

A pervasive problem in the automatic analysis of natural language sentences is ambiguity. A natural language sentence may be assigned different syntactic analyses, for example, the sentence “Time flies like an arrow” may admit at least the analyses demonstrated in Fig. 3.2.

A parsing system aims to find a single syntactic analysis reflecting the human perceived interpretation of the sentence. That is, for the sentence “Time flies like an arrow” we would like to pick the (Fig. 3.2a) analysis, while for the sentence “Fruit flies like a banana”, though superficially similar, we would pick the analysis reflected in Fig. 3.2b.

---

<sup>2</sup>Constituency-based trees and dependency-based trees are the most common syntactic representation types that are produced by current parsing systems, and these are the structures we discuss in this chapter. It is however worth mentioning that there exist parsing systems that produce structures assigned by so-called *deep grammars*, such as Lexical-Functional Grammar (LFG, [14]) Head-Driven Phrase-Structure Grammars (HPSG [91]) and Combinatorial Categorial Grammars (CCG [97]). The methods that we discuss (data driven and statistical) can be applied effectively to these other types of graphs too. See, for instance, [18, 50, 79].

## Models and Algorithms

Given a *treebank*, that is, a finite set of example sentences annotated with their correct parse-trees  $\{(x, y) | x \in \mathcal{X}, y \in \mathcal{Y}\}$ , a parsing model aims to induce a prediction function  $h$  such that  $h(x') \in \mathcal{Y}$  is a parse-tree for  $x' \in \mathcal{X}$ . Since a parse-tree is a complex structure, we cannot hope to induce a model that predicts a syntactic parse tree as a whole. The model thus represents such structures by means of simpler events that can be observed in annotated texts, and which can be used to construct novel analyses for unseen texts. These events may be the rules of a formal grammar [21], transitions in a state machine [80], pieces of the parse tree itself [12] and so on.

Formally, the parsing model  $M$  may be represented as a triplet, where  $\Gamma$  is a set of constraints over the formal representation,  $\lambda$  is a set of model parameters and  $h$  is an algorithmic implementation of the prediction function [62].

$$M = [\Gamma, \lambda, h] \quad (3.2)$$

The constraints in  $\Gamma$  define the form of the syntactic analysis (for instance, whether they are dependency trees or constituency trees). The scores or weights of the simple events are the model parameters in  $\lambda$ , and they are estimated from annotated data based on corpus statistics. The algorithm  $h$  predicts a parse-tree given the input sentence, and it technically referred to as the decoding algorithm.

Due to syntactic ambiguity, a sentence  $x \in \mathcal{X}$  may admit more than one syntactic analysis. So a crucial task of the model is to disambiguate, to pick out the correct syntactic analysis for a particular sentence  $x \in \mathcal{X}$ . Let us constrain the set  $\mathcal{Y}_x$  to be the set of all possible trees for a given sentence  $x \in \mathcal{X}$ . In the general case,  $h$  implements an algorithm that finds the highest scoring analysis for a given sentence.

$$h(x) = \operatorname{argmax}_{y \in \mathcal{Y}_x} score(y) \quad (3.3)$$

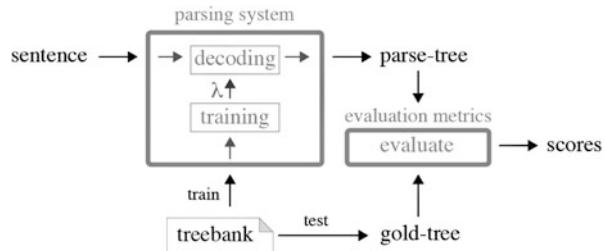
The score of an analysis  $y$  is defined to be a multiplication of the model parameters  $\lambda_i$  with a feature vector  $f(x, y)$  representing the events in the parse tree  $y \in \mathcal{Y}_x$ .

$$score(y) = \sum_i \lambda_i f_i(x, y) \quad (3.4)$$

In order to implement a parsing system as in (3.3), we need to define the following:

- The modeled events: That is, the relevant pieces of information that make up the parse tree, and can be observed in annotated data.
- The training algorithm: That is, a statistical method that is used to estimate the model parameters, that is, the weights of the modeled events.

**Fig. 3.3** An overall architecture for statistical data-driven parsing and evaluation



- The decoding algorithm: That is, an algorithm that can construct and traverse possible analyses given a sentence and pick out the highest scoring one.

The overall parsing architecture is depicted in Fig. 3.3. A treebank enters a **training** phase in which the modeled events are observed and the model parameters are estimated. The estimated parameters are then fed into the **decoding** algorithm, which accepts a new sentence as input. Based on the parameters and the formal constraints on the representation, it outputs the highest scoring analysis.

In order to quantitatively evaluate the performance of the parsing system we need to parse sentences that have not been seen during training, and compare the predictions with the correct analyses. We follow a common practice in machine learning where the annotated data (the treebank) is first split into a training set and a test set that are disjoint. The model parameters are estimated based on the train set, and the decoding algorithm is applied to raw sentences in the test set. The part of the treebank reserved as a test set contains the gold analyses for the parsed sentences, and comparing parse hypotheses with gold analyses allows us to quantify the parser's performance as scores (Fig. 3.3).

This architecture presents a general framework in which parsing systems can be formalized, modeled and implemented. Current approaches to parsing can be roughly grouped into (i) grammar-based approaches (ii) transition-based approaches and (iii) global discriminative approaches.<sup>3</sup> Let us characterize these approaches in turn.

#### • Grammar-Based Approaches

A generative grammar is a formal finite system which consists of rules that can be used to generate syntactic parse trees. In generative grammar-based parsing systems, the modeled events are grammar rules, and that the vector  $f(x, y)$  for a parse tree  $y \in \mathcal{Y}_x$  contains the rules that are used in deriving the tree according to the formal grammar. The  $\lambda$  vector contains rule scores or probabilities which may be estimated by maximizing the likelihood of the trees in syntactically

---

<sup>3</sup>These methods are orthogonal to the kind of formal syntactic representation being used. That is, grammar-based generative models can be used to generate constituency or dependency structures. Transition-based methods can be defined for constructing dependency graphs or constituency trees.

annotated data. The decoding algorithm for a generative system is requires to build and explore the set of generated trees which yield the input sentence. Efficient algorithms for this task are often based on dynamic programming methods that can pack and efficiently traverse an exponential number of trees in polynomial time and space complexity. Parsing models that are based on probabilistic generative grammars underly the most accurate statistical parsers for English to date. The use of such frameworks for parsing Semitic languages is the main case study addressed in this chapter.

- **Transition-Based Approaches**

A transition-based system is a formal machine that defines states and possible transitions between them. Every transition between states dictates an action that can be applied to a (partial) parse. The transition system defines a start state and a finish state, and transitions are defined such that every transition sequence from the initial state to the finish state corresponds to a sequence of actions that constructs a valid parse tree for the input sentence. In a transition-based system, the modeled events represented in  $f(x, y)$  are state-transition pairs, and the weights in  $\lambda$  are the scores for the different transitions in each state. Some transition-based decoding algorithms serve the fastest parsers to date – it is possible to construct a greedy algorithm which selects  $n$  transition between words, one pair-of-words at a time, and predicts a dependency tree in time complexity linear in the length of the input sentence.

- **Global Discriminative Approaches**

In discriminative parsing systems the modeled events are pieces or factors of the parse-tree and the weights represent scores of these factors. Those weights are estimated discriminatively using a globally optimized training procedure. In practice, such parsing systems typically assume a generative component that generates all possible parse trees in  $\mathcal{Y}_x$  for a given sentence  $x$  (usually in a packed representation), and the decoding algorithm explores the different trees and seeks the highest scoring one by aggregating the scores of the factors. Dynamic programming may be useful here too, though the level of factoring and the use of non-local features may undermine the feasibility of developing such methods.

The challenges in parsing Semitic languages transcend different representational and algorithmic choices. In the remainder of Sect. 3.1 we outline the main properties of Semitic grammar and show how they challenge the general-purpose parsing architectures just described. Section 3.2 focuses on a single case-study, generative constituency-based statistical parsing, and presents modeling methods that effectively address the different kinds of challenges outlined below. Section 3.3 presents parsing results for the different parsing approaches, as they were empirically shown to perform on Semitic languages, with notes on available resources.

### 3.1.2 Semitic Languages

The Semitic language family is a group of related Afro-Asian languages that are spoken across the middle-east and North Africa. Semitic languages are spoken by 270 million native speakers nowadays, where the most widely-spread ones are Arabic (206 million), Amharic (27 million) and Hebrew (7 million). Many of the morphological Semitic phenomena, including the rich system of conjugations (e.g., *binyanim*), inflections, and other syntactic constructions (e.g., *idafa/smixut*) are shared across the Proto-Semitic descendants. This section presents the orthographic, morphological and syntactic phenomena which pose challenges to the design of statistical parsing systems of the kinds we described above.

#### Script and Orthography

Most Semitic languages are written from right to left, and they employ an alphabetic system that is based on consonants and omit some or all of the vowels.<sup>4</sup> Many Semitic languages are known for their notorious use of vocalization patterns, which are indicated by means of subscript or superscript diacritics. Diacritics are explicit in educational texts, but they are usually omitted in standard texts.

Take, for instance, the Hebrew form *fmnh*. It may admit at least three readings.<sup>5</sup>

##### (2) Hebrew

- a. shmena ('fat', ADJ.3Fem.Sing)
- b. shimna ('lubricated', VB.3Fem.Sing)
- c. she-mana ('that' + 'counted', VB.3Masc.Sing)

The lack of diacritics does not pose a problem for mature readers of the language because the correct analysis is easily resolvable based on context. In the following, for example, the agreement features of the subject help to select the right analysis.

##### (3) Hebrew

- a. *hxtwlh fmnh*  
the-cat.FemSing gained-weight.FemSing (analysis (2a))
- b. *hild fmnh*  
the-child.MascSing that-counted.MascSing (analysis (2c))

The selected analysis in (3b) reveals further peculiarity of the Semitic writing system. In Semitic languages, many linguistic elements such as determiners, definite articles, subordinators and conjunction markers, do not appear on their own.

---

<sup>4</sup>A notable exception is Maltese, see Chap. 1, this volume.

<sup>5</sup>We use the Hebrew transliteration of Sima'an et al. [96] and the Arabic transliteration of Habash [49]. The linguistic examples are taken from, or inspired by, Tsarfaty [101] and Shilon et al. [94].

They are concatenated as affixes to the next open-class word in the sentence. This means that a space-delimited token may contain different segments carrying their own part-of-speech tags, as in (4)–(5).

## (4) Hebrew

*wkfmhbit*

- a. *w*/CC *kf*/REL *m*/PREP *h*/DET *bit*/NN  
and/CC when/REL from/PREP the/DET house/NN

## (5) Arabic

*llqlm*

- a. *l*/PP *A*//DET *qlm*/NN  
*l*/for *A*//the *qlm*/pen

Given a word-token out of context, there is massive ambiguity as to how it should be vocalized and segmented. This has far reaching consequences for syntactic analysis. Before we can syntactically analyze the sentence, the system should disambiguate the morphological analysis. However, the disambiguation may need cues from syntactic context, which is provided by the parser. Section 3.1.3 discusses the challenge of dealing with this apparent loop.

## Morphology

The morphological component of a grammar describes the internal structure of words. As discussed in Chap. 1, Semitic languages are known for their rich morphology, encompassing at least derivational and inflectional morphemes. Due to extensive morphosyntactic interactions, morphological phenomena may be directly relevant to making correct syntactic predictions.

### Derivational Morphology

In Semitic languages, consonantal roots are the main carrier of meaning. Nouns, Verbs and Adjectives in Semitic languages are derived by combining consonantal roots with a range of morphological templates that combine vocalization patterns and extra consonants. For instance, from the Hebrew root *k.t.b* (roughly “write”) one can derive the nouns *ktb* (*ketav*, “script”) and *mktb* (*mikhtav*, “a letter”), the verbs *lktwb* (*likhtov*, “to write”) *lhktib* (*lehacktiv*, “to dictate”) and *lhtktb* (*lehitkatev*, “to correspond”), and the adjective/participial *ktwb* (*katuv*, “written”). Semitic roots are not unrelated. The Arabic root *k.t.b.* (“write”) can be used to derive the nouns *ktAb* (*kita:b*, ‘book’) and *mktb* (*maktab*, “office”), the verbs *kAtb* (*ka:taba*, “correspond with”), *Akttb* (*iktataba*, “signed, donated”) and the participials *katb* (*ka:tib*, “writing, writer”) and *mktwb* (*maktab*, “written, letter”). Morphological templates have implication for the grammatical structure of the sentence, for instance, they carry

information concerning the event structure and the type of participants involved in the predicate [34, 99]. In this chapter we treat each root + template combination as a holistic lexical element (henceforth, a *stem* or a *lexeme*) carrying its own POS tag, its lexical meaning and its subcategorization frame.

## Inflectional Morphology

Each lexical element in Semitic languages may be realized in different word forms, corresponding to a set of inflectional features that are appropriate for its syntactic context. The word forms that correspond to the Hebrew verbal lexeme *ktb* (“write”), for instance, realize gender, number and person features, as illustrated in (6). This set of forms is called a paradigm [3].

### (6) Hebrew

	Singular			Plural		
	1st	2nd	3rd	1st	2nd	3rd
<b>Past</b>						
Masculine	<i>ktbti</i>	<i>ktbt</i>	<i>ktb</i>	<i>ktbnw</i>	<i>ktbtm</i>	<i>ktbw</i>
Feminine		<i>ktbt</i>	<i>ktbh</i>		<i>ktbtm</i>	
<b>Present</b>						
Masculine		<i>kwtb</i>			<i>kwtbim</i>	
Feminine		<i>kwtbt</i>			<i>kwtbw</i>	
<b>Future</b>						
Masculine	<i>aktwb</i>	<i>tktwb</i>	<i>iktwb</i>	<i>nktwb</i>	<i>tktbw</i>	<i>iktbw</i>
Feminine		<i>tktbu</i>	<i>tktwb</i>		<i>tktwbnh</i>	<i>iktwbnh</i>

The sets of inflectional features that are realized by morphological markers are similar across Semitic languages. In Hebrew and Arabic, verbs are inflected for gender, number and person, Arabic verbs are inflected for case, aspect and mood. Hebrew and Arabic nouns are inflected for gender and number. In addition, nouns and adjective are inflected to mark their so-called state (*smixut/idafa*), determining genitive relations in complex noun compound constructions.

The richness of the paradigms makes it hard to observe all possible realization possibilities of a lexeme in a finite treebank. This leads to a high OOV (out of vocabulary) words rate, a challenge that we address in Sect. 3.2.4.

## Syntax

The Semitic grammar uses rich morphological marking to indicate, in systematic ways, how words combine to form grammatical phrases and sentences. In English, the order of words and phrases is a significant factor determining how the words

should be combined to deliver certain meanings. For instance, the subject must precede the verb and the direct object has to follow it. In Semitic languages, subjects, objects and modifiers are alternatively by marked means of morphology. There are two ways to mark grammatical functions morphologically. One is by *case assignment* on a word or a phrase, indicating its relation to the sentence predicate, and the other is by *agreement*, where the inflectional features on two different words or phrases agree in order to indicate a grammatical relation between them. Due to the rich system of morphological argument marking, the word-order patterns in Semitic clauses are less rigid than in English.

## Word-Order

The most prominent dimension of variation across languages is their *basic word-order*, that is, the order in which the Subject, Verb and Object appear in a canonical sentence [47]. In the Proto-Semitic language, the default word-order has been Verb-Subject-Object (VSO). This order is retained in Classical Arabic and Biblical Hebrew. In Modern Semitic languages including Modern Hebrew and many Arabic dialects this order has changed into Subject-Verb-Object (SVO) default order, similar to the canonical order in English. Ethiopic Semitic languages follow a Subject-Object-Verb (SOV) order. This variation in word-order is not only attested across languages but also within the corpora of particular languages. In naturally occurring Semitic texts we can attest both SVO and VSO constructions, as well as V2 constructions in which a preposed object or modifier triggers subject/predicate inversion (similar to Germanic languages) [95]. Within nominal phrases, we often find relatively fixed word order patterns. In Arabic and Hebrew nominals we find possessed–possessor (NG), and noun–adjective (NA) orders. Modern Ethiopic languages have a possessor–possessed, and adjective–noun order within noun phrases.

## Case-Marking

Semitic languages utilize a case system in which the nominative, accusative and genitive cases are marked. Modern Standard Arabic maintains such case endings in literary and broadcasting contexts. Ethiopian languages preserved the accusative case ending. Hebrew and Amharic use a differential Object marking system, in which objects are marked by the acc marker if and only if they are also marked for definiteness. Such explicit case marking allows to determine the grammatical function of a word or a phrase regardless of its position.

### (7) Hebrew Object Marking [31]

- a. *dni ntn mtnh ldinh*  
Dani gave present to-dina  
“Dani gave a present to Dina”

- b. *dni ntn at hmtnh ldinh*  
 Dani gave ACC DEF-present to-Dina  
 “Dani gave the present to Dina”

(8) Amharic Object Marking [5]

- a. *Lemma wiffa j-aj-al.*  
 Lemma dog 3MascSing-see-AUX(3MascSing)  
 “Lemma sees a dog.”
- b. *Lemma wiffa-w-in j-aj-ew-al.*  
 Lemma dog-DEF-ACC 3MascSing-see-3MascO-AUX(3MascSing)  
 “Lemma sees the dog.”

Semitic nouns and adjectives can be inflected for state, also known as *idafa* in Arabic or *smixut* in Hebrew. This is the morphological marking of genitive case, which can be productively used to create arbitrarily long genitive constructions.

(9) Construct State Nouns in Hebrew

- a. *bit hspr*  
 house-of the-book  
 school, lit: the house of the book
- b. *mnhl bit hspr*  
 principal-of house-of the-book  
 The school principal

(10) The Idafa Construction in Arabic

- a. *mdyr Almdrsp*  
 principle-of the-school  
 The school principal
- b. *Abn mdyr Almdrsp*  
 son-of principle-of the-school  
 The school principal’s son

Definite marking (*nunation* in Arabic) occurs on the single (final) head of such constructions. Due to the possibility of nesting *idafa/smixut* constructions, the definite marker may be arbitrarily distant from the related genitive-marked noun.

## Agreement

The inflectional features on two different elements in the sentence may share their values in order to indicate a grammatical relation between these words or phrases. Semitic languages exhibit patterns of agreement both at the clause level and phrase level. For instance, Semitic subjects agree with predicates on gender, number and person, as in (11). (In certain configurations this is not so – in Arabic VSO sentences, the verb is always singular (12).) In noun phrases in both Hebrew and Arabic the noun agrees with its modifier on gender, number and definiteness.

## (11) Agreement in Hebrew SVO

- a. *hildim ktbw*  
 kid.MascPl write.past.3MascPl  
 “The boys wrote”

## (12) Agreement in Arabic VSO

- a. *ktb Al+AwlAd*  
 write.past.MascSing boy-MascPl.Def  
 “The boys wrote”

## Verbless Sentences

Some Semitic sentences lack a verbal predicate altogether. In such sentences, the predicate is realized by a nominal phrase, an adjectival phrase or a prepositional phrase. This phrase determines the inflectional properties of the predicate.

## (13) Hebrew

- a. *dni mwrh*  
 Danny teacher  
 “Danny is a teacher”
- b. *dni xkm*  
 Danny smart  
 “Danny is smart”

Verbless (nominal) sentences can indicate identity sentences. In such cases, pronominal elements act as copular elements, linking the subject to the nominal predicate. These pronominal elements must agree with the subject on inflectional features.

## (14) Hebrew

- a. *dni hwa mwrh*  
 Danny Pron.MascSing teacher.MascSing  
 “Danny is a teacher”
- b. *dinh hia mwrh*  
 Dina Pron.FemSing teacher.FemSing  
 “Dina is a teacher”

## Clitics

Pronouns indicating grammatical functions such as subject and object may be dropped if the verb is inflected to reflect their pronominal inflectional features.

(15) Hebrew

a. *raitih*

Saw.1Sing.3FemSing

I saw her

(16) Arabic

a. *rAythA*

Saw.1Sing.3FemSing

I saw her

All in all, a Semitic word contains information about the lexical meaning, argument structure, inflectional features, and one or more grammatical relations that are realized by the word. This information provides crucial cues concerning the syntactic structure of a sentence. In parsing, we wish to utilize such cues when we search for the correct syntactic analysis of the sentence (Sect. 3.2.3).

### 3.1.3 The Main Challenges

Whether we are aimed at dependency parsing or constituency parsing, whether our parser relies on a generative grammar, a transition-based system or a graph-based framework, some basic assumptions inherent in the design of parsers for English break down when the system is applied for parsing Semitic languages. One such assumption is the notion of an input word. In English, the input words are assumed to appear as is in the parse tree. An additional assumption made in parsing English is that the position of words largely determines their grammatical functions and how they can be combined. In Semitic languages, non-adjacent words may interact in non-trivial ways due to complex morphological marking. There is also an implicit assumption in the design of statistical parsers for English that it is feasible to derive a comprehensive probabilistic lexicon from treebank data. Due to word-form variation and high ambiguity, Semitic lexicons are hard to bootstrap. We hereby delineate the emerging challenges as the architectural challenge, the modeling challenge, and the lexical challenge.

#### The Architectural Challenge

In English and similar languages, space-delimited word token represents the basic units for syntactic analysis. Due to the rich morphology of words of Semitic languages and their complex orthography (Sect. 3.1.2), every space-delimited word-token may contain multiple units which carry independent grammatical categories. To illustrate, recall the Hebrew word-token *wkf'mhbit* in Example (4). It contains multiple morphological units indicated with their own part-of-speech tags (and/CC when/REL from/PREP the/DT house/NN).

In order to parse a sentence in a Semitic language, the text has to go through morphological analysis which uncovers the sequence of word segments which can be combined into phrases and sentences. Morphological analysis in Semitic languages is however highly ambiguous due to morphological variation, complex orthography, and the omission of diacritics. For example, a Hebrew word like *bclm* may admit different analyses, as illustrated in (17).

(17) *bclm*

- a. *b/IN clm/NN*
- b. *b/IN h/DT clm/NN*
- c. *b/IN h/DT cl/NN fl/POSS hm/PRN*
- d. *bcl/NN fl/POSS hm/PRN*

Such ambiguity can only be resolved in the context at which the word token occurs. Sometimes, local context such as neighboring words is sufficient for disambiguating the morphological analysis. At other times, a word that contains disambiguating cues may be distant from the word that needs to be disambiguated.

Consider the Hebrew word *hneim* in the below example. Here, the existence of an agreeing element helps to pick out the correct morphological analysis of this word.

(18) a. *bclm hneim fl hecim*

in-DEF-shadow DEF-pleasant of the-trees  
“In the pleasant shadow of the trees”

b. *bcl fl hecim hwa at zmninw hneim*

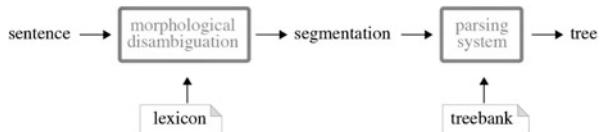
in-DEF-shadow of DEF-trees he.MascSing ACC time-ours made-pleasant.  
1MascSing  
“In the shadow of the trees he made our time pleasant”

In (18a) there is agreement of *hneim* on the definite article with the previous noun, which renders the correct analysis of *hneim* a definite adjective. In (18b), there is agreement with the pronoun “he”, which is the subject of the entire sentence. This agreement helps us understand *hneim* as a verb, inflected to reflect the subject properties.

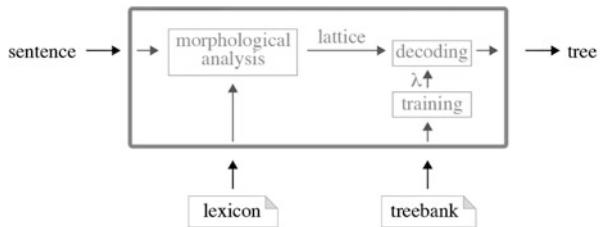
So, in order to assign a syntactic analysis we first need to identify the correct morphological segments, but in order to pick out the correct morphological segmentation in context, we need information concerning the overall syntactic structure. How can we construct an architecture that can deal with this loop? Computationally, there are two main strategies:

- **A Pipeline Architecture.** In a pipeline architecture, we first set up a morphological disambiguation component, and then provide the most probable morphological segmentation as input to a standard parsing model. The parser thus aims to assign a tree to the given sequence of morphological segments.
- **A Joint Architecture.** An alternative way to build the architecture is to select the most probable morphological segmentation and syntactic analysis at once.

**Fig. 3.4** A pipeline architecture for parsing Semitic languages



**Fig. 3.5** A joint architecture for parsing Semitic languages



The two types of architectures for parsing Semitic languages are sketched in Figs. 3.4 and 3.5 respectively. The appealing property of a pipeline approach is its simplicity and modularity. The downside of a pipeline is that errors in the morphological disambiguation component may propagate to the parser and seriously undermine its prediction accuracy. The main advantage of the joint strategy, as advocated by Tsarfaty [100], Cohen and Smith [28], Goldberg and Tsarfaty [42], and Goldberg and Elhadad [39], is that the joint architecture allows us to use syntactic information to disambiguate morphological information, and vice versa. It also helps to avoid error propagation. An apparent downside of this strategy is that it may make the search space over joint morphological and syntactic analyses a lot larger, which may in turn lead to search errors. In Sect. 3.2 we present an efficient lattice-based decoder that cater for a joint solution for syntactic and morphological disambiguation.

Correctly resolving the morphological ambiguity in the input is essential for obtaining a correct parse, but whether this morphological disambiguation should be done before or jointly with the parser is an empirical question.

## The Modeling Challenge

A crucial factor in the modeling of a parsing system is the choice of modeled events. Intuitively, these events reflect regular patterns in the annotated data and allow us to generalize from them. In practice, we have to ensure we pick out relevant regularities that lead to accurate predictions. These considerations require us to take into account the properties of the languages that we aim to parse.

Take the notion of word-order in different languages. There exist different word ordering patterns in different languages, some are more flexible than others. If we are dealing with a language with fixed word order, such as English, using the order of the different elements as model parameters may help us make very accurate predictions, because we will find repetitive patterns in high frequencies. For

languages with free word order, picking word-order patterns as model parameters will not necessarily help us in making precise predictions, since the correctness of a syntactic structure is dependent on other factors, and identical word-order patterns will not capture the same kind of grammatical relations. In such languages we may want to parameterize other features that are likely to exhibit more regular patterns.

When we design a parsing model for a Semitic language, we have to take into account the following facts: (i) word order and morphological information jointly determine the predicate-argument structure of sentences, (ii) word order may be different for similar predicate-argument constructions, and (iii) word combinations may be indicated by matching morphological features of remote words. In Sect. 3.2.3 we survey different ways to parametrize such factors, implicitly or explicitly, in a generative parsing model.

## The Lexical Challenge

The syntactic analysis of sentences requires us to know the correct morphological analysis of words and morphological segments. The morphological analysis of Semitic words typically includes their part-of-speech tags, pronominal clitics, and various inflectional features that are reflected in the form of the words. We refer to this information as the *morphosyntactic representation (MSR)* of the word form in context. These MSRs present the interface between word level information and sentence structure, and the parameters of assigning MSRs to words are called lexical parameters. During parser development in lab settings it is often assumed that the input words are provided to the parser with correctly disambiguated MRSs. (This happens in both constituency-based and dependency-based parser settings.) In realistic scenarios, when parsing unstructured texts (e.g., web data), the MSRs are not known in advance.

One way in which the model can learn how to automatically assign MSRs to words is by observing the relative frequency of different analyses in an annotated corpus. In languages such as English, there is not much variation in the form of words in different syntactic contexts and we can attest enough occurrences for each word to robustly estimate the lexical parameters. This is not the case in morphologically rich languages, and Semitic languages in particular, where a single lexeme may be realized as a paradigm of different word forms that realize additional inflectional features for gender, number and person (the complete paradigm for a single lexeme “worked” in English includes seven forms *ebd*, *ebdti*, *ebdt*, *ebdnw*, *abdtm*, *ebdtm*, *ebdw* in Hebrew).

In languages such as English, the amount of annotated data appears to be sufficient for estimating the lexical parameters, but treebanks for languages that have been understudied have a fairly limited amount of annotated texts, typically not enough for observing a sufficient number of lexical items in different contexts. Even if we are given an external resource such as a complete lexicon for the language, we usually do not have sufficient annotated data to estimate the parameter weights for the MSRs and score the complete trees.

This problem is referred to as high OOV (out of vocabulary) rate, and it is also found in domain adaptation scenarios where a trained parser is applied to texts from a different domain. In Sect. 3.2.4 we survey methods that can be used for scoring the lexical parameters using additional resources, for instance, an external morphological analyzer and additional amounts of unannotated data.

### 3.1.4 Summary and Conclusion

The grammar of Semitic languages is different than that of English, in the sense that their word order is a lot more complex and the syntactic structures are more flexible. These properties challenge existing parsing architectures in various ways. The parsing architecture has to handle complex, ambiguous input words. When constructing and scoring candidate parse trees, the model has to allow for flexible word-order patterns while ensuring the coherence of their morphological marking. From a machine learning point of view, a probabilistic lexicon in Semitic languages is harder to obtain, due to high word form variation and massive ambiguity.

The next section demonstrates how the design of a parsing system can effectively address these challenges. We show how morphological information can be disambiguated jointly with the parse tree, we demonstrate different ways to parameterize a parser such that it can take into account, implicitly or explicitly, complex morphosyntactic interactions, and we also discuss how a parser can exploit additional resources (such as a dictionary and/or unannotated data) in order to handle word-form variation. We further show how to evaluate parsers, taking into account both the morphological and syntactic components of the predicted analyses.

## 3.2 Case Study: Generative Probabilistic Parsing

Generative probabilistic models are the earliest and most commonly studied models for constituency-based parsing. Despite theoretical claims concerning the inadequacy of context-free grammars for analyzing natural language data [93], statistical modeling based on Probabilistic Context-Free Grammars (PCFGs) underlies the implementation of many accurate parsers for English [21, 23, 30, 85].<sup>6</sup> The conceptual simplicity and empirical viability of the framework has led to many adaptations, and several language-independent parsing frameworks have been developed based on it [8, 59, 85]. However, applying these ‘ready-made’ frameworks to parsing Semitic languages requires non-trivial adaptation or reconstruction in order to perform well on Semitic data [28, 32, 42, 45, 63, 100, 102].

---

<sup>6</sup>The best result reported for English constituency parsing on the Penn treebank is now just above 92 F-Score [76].

In this chapter we describe building blocks of generative probabilistic models for parsing and demonstrate how to reconstruct the model in ways that address the challenges we outlined above. Addressing these challenges allows improves parsing results for Hebrew and Arabic significantly. Parsers for Amharic, Syriac or Maltese are relatively simple to derive based on the general framework presented here.

### 3.2.1 Formal Preliminaries

We formally define parsing as a structure prediction task implementing a prediction function  $h : \mathcal{X} \rightarrow \mathcal{Y}$  where  $\mathcal{X}$  is a set of sentences in a language over a vocabulary  $\mathcal{T}$ , and  $\mathcal{Y}$  is a set of linearly-ordered labeled trees with terminals drawn from  $\mathcal{T}$ . Due to syntactic ambiguity, an input sentence  $x \in \mathcal{X}$  may admit multiple analyses which we denote by the set  $\mathcal{Y}_x$ . In a probabilistic model, we aim to find the most probably parse tree given the input sentence:

$$h(x) = \operatorname{argmax}_{y \in \mathcal{Y}_x} p(y|x) \quad (3.5)$$

We spell out (3.5) further by incorporating the definition of conditional probability and by observing that  $p(x)$  is constant with respect to the maximization (3.8).

$$h(x) = \operatorname{argmax}_{y \in \mathcal{Y}_x} p(y|x) \quad (3.6)$$

$$= \operatorname{argmax}_{y \in \mathcal{Y}_x} \frac{p(y, x)}{p(x)} \quad (3.7)$$

$$= \operatorname{argmax}_{y \in \mathcal{Y}_x} p(y, x) \quad (3.8)$$

In languages such as English, each sentence  $x \in \mathcal{X}$  is contained in any tree  $y \in \mathcal{Y}_x$ , because the words stand as terminals in the tree. So we can simplify  $h(x)$  further:

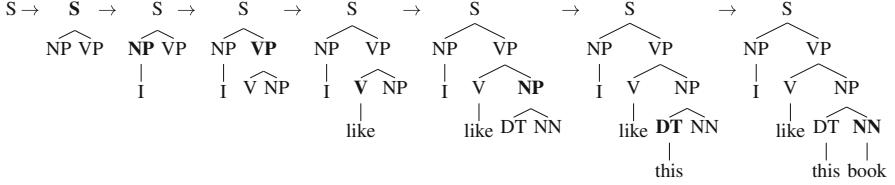
$$h(x) = \operatorname{argmax}_{y \in \mathcal{Y}_x} p(y) \quad (3.9)$$

A parse tree  $y \in \mathcal{Y}_x$  for an input sentence  $x \in \mathcal{X}$  is a complex structure, too complex to be observed directly in the training data. Therefore, a common strategy is to break down the construction of the tree into a finite sequence of decisions  $d_1, \dots, d_n$  such that  $y = d_1 \circ \dots \circ d_n$ . The probability of the tree equals the multiplication of the probabilities of the conditional decisions (3.9).

$$h(x) = \operatorname{argmax}_{y \in \mathcal{Y}_x} p(d_1 \circ \dots \circ d_n) \quad (3.10)$$

$$= \operatorname{argmax}_{y \in \mathcal{Y}_x} p(d_1|\emptyset) \times \dots \times p(d_n|d_1 \dots d_{n-1}) \quad (3.11)$$

$$= \operatorname{argmax}_{y \in \mathcal{Y}_x} \prod_{i=1}^n p(d_i|d_1 \dots d_{i-1}) \quad (3.12)$$



**Fig. 3.6** A context-free derivation of the sentence “I like this book”. For each rule  $\alpha \rightarrow \beta$ , the substitution site  $\alpha$  is in **bold**, and its dominated daughters form the  $\beta$  sequence

The modeled events are the decisions  $d_i$  and the model parameters are the probabilities of applying each decision in a certain conditioning context. The conditioning context is narrowed down to relevant aspects of the tree generation. We indicate it by a function  $\psi$  that selects relevant features of the generation.

$$h(x) = \operatorname{argmax}_{y \in \mathcal{Y}_x} \prod_{i=1}^n p(d_i | \psi(d_1 \circ \dots \circ d_{i-1})) \quad (3.13)$$

## Probabilistic Grammars

A constituency tree can be thought of as having been generated via a formal generative device that we call a *grammar*. Formally, a grammar  $G$  is a tuple

$$G = \langle \mathcal{T}, \mathcal{N}, S, \mathcal{R} \rangle$$

Where  $\mathcal{T}$  is a finite set of terminal symbols,  $\mathcal{N}$  is a finite set of non-terminal symbols,  $S \in \mathcal{N}$  is a designated start symbol, and  $\mathcal{R}$  is a set of grammar rules. A *context free grammar* (CFG) has rules of the form  $\alpha \rightarrow \beta$  where  $\alpha \in \mathcal{N}$  is a non-terminal symbol and  $\beta \in (\mathcal{T} \cup \mathcal{N})^*$  is an ordered sequence of terminals and non-terminal symbols. These rules indicate a substitution of a non-terminal symbol in the left-hand side with the sequence of symbols at the right-hand side of the rule. Such substitution is independent of the context of the non-terminal in the left-hand side, and hence the name: context-free grammars [26].

A context-free grammar can be used to generate syntactic parse trees. Given a CFG we can obtain parse trees by means of sequentially applying rules from  $\mathcal{R}$ . The derivation of the sentence “I like this book” in Fig. 3.1a is graphically depicted in Fig. 3.6. When we fix a particular traversal order of trees, say, BFS, every context-free derivation corresponds to a unique tree and vice versa. That is,  $y = r_1 \circ \dots \circ r_n$ .

We can extend this generative device into a probabilistic grammar that can assign non-zero probability to every sentence in the language generated by the grammar. A probabilistic context-free grammar is a CFG tuple extended with  $p : \mathcal{R} \rightarrow [0, 1]$ , a probability mass function assigning probabilities to rules.

$$PCFG = \langle \mathcal{T}, \mathcal{N}, S, \mathcal{R}, p \rangle$$

The function  $p$  is defined such that  $\forall r \in \mathcal{R} : 0 \leq p(r) \leq 1$ . In order to guarantee assignment of probability mass to all sentences that are generated by the grammar,  $p$  has to be defined such that the probability of all rules with the same symbol at the left-hand side sums up to one.

$$\sum_{\{\beta | \alpha \rightarrow \beta \in R\}} p(\alpha \rightarrow \beta) = 1 \quad (3.14)$$

The probability assigned to a tree by a PCFG is calculated as the probability of its unique derivation  $y = r_1 \circ \dots \circ r_n$ . Due to context-freeness, the application of context-free rules is independent, so we can simply multiply rule probabilities.

$$p(y) = p(r_1 \circ \dots \circ r_n) = p(r_1) \times \dots \times p(r_n) = \prod_{i=1}^n p(r_i) \quad (3.15)$$

Our baseline probabilistic parsing model is defined to be one that finds the most probable parse tree for a given sentence by computing the probability of its derivation according to a given PCFG. The decisions  $d_i$  are CFG rules, and the conditioning context is the label at the substitution site.

$$h(x) = \operatorname{argmax}_{\{y \in \mathcal{Y}_x\}} \prod_{A \rightarrow \alpha \in y} p(A \rightarrow \alpha | A)$$

## Training

In order to score syntactic parse trees generated by the probabilistic grammar we need to know the probabilities of rules of the form  $p(\alpha \rightarrow \beta | \alpha)$ . In a data-driven setting, we extract the grammar rules (the modeled events) and estimate the model parameters (the probability of the rules) from the annotated trees in the treebank.

The trees in the treebank are decomposed into context-free rules, and the probability of each rule may be estimated using relative frequency counts. Formally, if  $Count : \mathcal{R} \rightarrow \mathbb{N}$  indicates corpus counts of the rules occurrences, then the empirical estimate of  $\hat{p}$  is defined as follows.

$$\hat{p}(\alpha \rightarrow \beta | \alpha) = \frac{Count(\alpha \rightarrow \beta)}{\sum_{\gamma} Count(\alpha \rightarrow \gamma)}$$

Probabilistic grammars extracted in this way are called treebank grammars [21]. Every treebank grammar satisfies the condition that  $\hat{p}$  assigns non-zero probabilities for every rule in the grammar, and the sum for all rules with the same left hand side equals to 1. This guarantees that we can assign non-zero probabilities for all the

trees generated by the grammar for sentences drawn from  $\mathcal{T}^*$ . For every sentence in the language generated by the treebank grammar it then holds that:

$$0 \leq \hat{p}(x) \leq 1$$

It can be shown that this method of grammar extraction provides unbiased and consistent statistical estimates. That is, assuming that the corpus is generated by some context-free grammar, if we had an infinite amount of annotated data, the probabilities estimated in this way will converge to the true probabilities in the grammar that generated the corpus.

## Decoding

Given a PCFG  $G = \langle \mathcal{T}, \mathcal{N}, S, \mathcal{R}, p \rangle$  and an input sentence  $x \in \mathcal{X}$ , we need an algorithm that can find the most probable parse for a sentence according to the probabilistic grammar. This phase is called *parsing* or *decoding*.

A naïve way to decode would be to enumerate all possible derivations of a sentence given the rules in  $\mathcal{R}$ , score them according to the probability model  $p$ , and pick out the analysis with the highest probability. However, since the number of analyses for a sentence of length  $n$  is exponential in  $n$ , this would be an inefficient way to decode.<sup>7</sup> In order to decode efficiently, we can store all parse trees in a two-dimensional chart and search for the highest probability parse-tree using dynamic programming methods. This is the idea behind the Cocke-Kasami-Younger (CKY) algorithm for parsing with PCFGs [56].<sup>8</sup>

A pre-condition for using the CKY algorithm is that each rule in the context-free grammar appears in one of the following two forms:

- Syntactic Rules:  $A \rightarrow B C$ ; where  $A, B, C \in \mathcal{N}$
- Lexical Rules:  $A \rightarrow \alpha$ ; where  $A \in \mathcal{N}$  and  $\alpha \in \mathcal{T}$

Such a grammar is said to be in Chomsky Normal Form (CNF) [26]. It can be formally shown that every PCFG can be uniquely converted into its CNF-equivalent by replacing a flat rule of  $n$  daughters with  $n - 1$  binary rules encoding intermediate steps in generating the sequence of daughters. In Fig. 3.7a we show how to assign a symbol for every intermediate step. Moreover, it is possible to convert a CFG into CNF by employing the Markov “limited history” assumption, where the right side of the rules records a limited number of previously generated sisters, as in Fig. 3.7b.

The CKY algorithm assumes a data-structure called a chart which is a two-dimensional array of size  $(n + 1) \times (n + 1)$  for a sentence of length  $n$ . Each cell

<sup>7</sup>The catalan number  $C_n = \prod_{k=2}^n \frac{n+k}{k}$  is the number of full binary trees with  $n + 1$  leaves.

<sup>8</sup>There are several algorithms that use dynamic programming and a two-dimensional chart. They can construct the tree top-down, bottom-up or left-to-right, they can use an agenda or not, and they can be exhaustive or greedy. Here we present an exhaustive bottom-up algorithm, used at the backbone of many state-of-the-art parsers, such as [8, 59, 85].

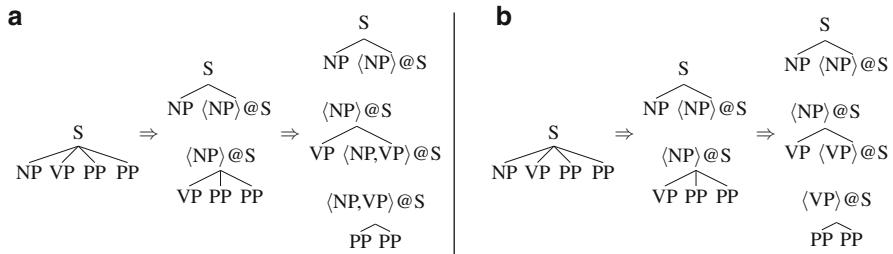


Fig. 3.7 (a) Conversion into Chomsky Normal Form (b) CNF 1st-order Markovization

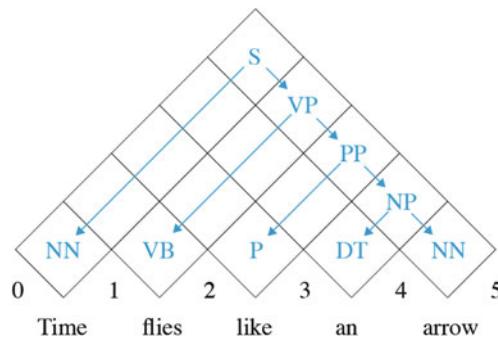


Fig. 3.8 The correct analysis of “Time flies like an arrow” stored in a chart

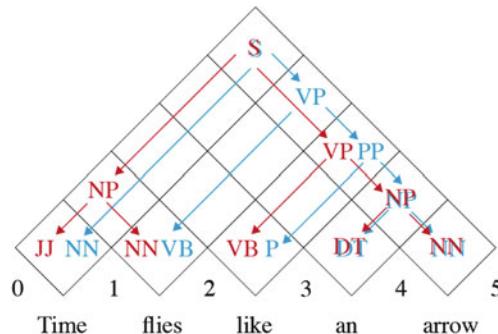


Fig. 3.9 The ambiguous analyses of “Time flies like an arrow” packed in a chart

$[i, j]$  may contain multiple chart-items of the form  $A[i, j]$  where  $A$  is a non-terminal symbol labeling the span between two indices  $0 \leq i < j \leq n$  in the sentence.

We store a tree in a chart by storing its labeled spans in the relevant chart cells and recording the back-pointers to the labeled daughters that created this span, as shown in Fig. 3.8. When we store multiple parse trees in one chart, there is a lot of overlap, such as the shared NP “an arrow” in Fig. 3.9. This overlap is the reason we can store an exponential number of analyses for a sentence of length  $n$ , in

---

**Algorithm 1** The CKY algorithm for chart parsing (with back-pointers)

---

```

1: for  $i = 1 \rightarrow n$  do
2:   for  $L = 1 \rightarrow |\mathcal{N}|$  do
3:      $\delta_L[i - 1, i] \leftarrow p(A_L \rightarrow w_i | A_L)$             $\triangleright$  Initiate pre-terminal probs
4:      $\beta_L[i] \leftarrow \langle w_i \rangle$                           $\triangleright$  Store words
5:   for  $span = 2 \rightarrow n$  do
6:     for  $end = span \rightarrow n$  do
7:        $begin \leftarrow end - span$ 
8:       for  $L = 1 \rightarrow |\mathcal{N}|$  do
9:          $\delta_L(begin, end) \leftarrow \max_{\langle m, J, K \rangle} p(A_L \rightarrow A_J A_K | A_L) \times \delta_J(begin, m) \times \delta_K(m, end)$ 
10:         $\beta_L(begin, end) \leftarrow \operatorname{argmax}_{\langle m, J, K \rangle} p(A_L \rightarrow A_J A_K | A_L) \times \delta_J(begin, m) \times \delta_K(m, end)$ 
11:   return RECONSTRUCT-TREE( $\delta_S[0, n]$ ,  $\beta_S[0, n]$ )            $\triangleright$  Follow back-pointers

```

---

polynomial space  $O(n^2)$ . Each chart cell can potentially store as many items as the number of non-terminals in the grammar. So, in practice, the space complexity of the algorithm is bounded by  $O(n^2 \times |G|)$  where  $|G|$  is the number of non-terminal symbols  $|\mathcal{N}|$ .

Algorithm 1 presents the CKY procedure. Given a PCFG with a set of rules  $\mathcal{R}$ , a sentence of length  $n$  and an empty chart, the algorithm proceeds by filling in the chart with all the analyses that are licensed by the grammar, scoring them, and picking the most probable analysis. In lines 1–4 we fill in the labels that can span 1-length terminals according to the grammar. Let  $L$  be the index of a part-of-speech label  $A_L \in \mathcal{N}$ . For each terminal  $t_i$ , the  $\delta$  values store the rule probabilities for each part-of-speech tag  $A_L \in \mathcal{N}$ .

$$\delta_L([i - 1, i]) = p(A_L \rightarrow t_i | A_L)$$

In lines 5–10 the algorithm considers sequences of length  $1 < span \leq n$ . For each cell  $[i, j]$  it adds a label  $A_L$  into this chart item iff there exists an index  $i < m < j$  and a rule  $A_L \rightarrow A_J A_K \in \mathcal{R}$  such that  $A_J[i, m]$  and  $A_K[m, j]$  also exist in the chart. For every label stored as a chart-item the algorithm stores the accumulated probability  $\delta$  based on the maximal probability of obtaining the daughters:

$$\delta_L(i, j) \leftarrow \max_{\langle m, J, K \rangle} p(A_L \rightarrow A_J A_K | A_L) \times \delta_J(i, m) \times \delta_K(m, j)$$

The algorithm also keeps the back pointers to the combined cells of the most probable chart item using the  $\beta$  values. Once the chart is completely filled, we find the start symbol  $\delta_S[0, n + 1]$  (where  $S$  indexes any designated start symbol) and traverse the back-pointer indicating the most probable chart items that construct the tree rooted in this symbol.

## Evaluation

The performance of a constituency based parsing model is standardly evaluated using the ParseEval measures [10], which calculate the precision and recall of labeled spans that have been correctly predicted. Let us assume a sentence of length  $n$ . Let  $P$  be the set of labeled spans  $\langle i, L, j \rangle$  in the parse-tree and let  $G$  be the set of labeled spans in the gold tree for the sentence. As in [35], we use the  $\hat{C}$  notation to denote only the root part of a tree  $C$ , that is, discarding the terminals and pre-terminal nodes. The notation  $|\hat{C}|$  indicates the number of labeled spans, discarding the root category. The labeled precision (LP), labeled recall (LR) and  $F_1$ -Score (their harmonic means) are calculated as follows:

$$LP = \frac{|\hat{P} \cap \hat{G}|}{|\hat{P}|} \quad (3.16)$$

$$LR = \frac{|\hat{P} \cap \hat{G}|}{|\hat{G}|} \quad (3.17)$$

$$F_1 = \frac{2 \times LP \times LR}{LP + LR} \quad (3.18)$$

### 3.2.2 An Architecture for Parsing Semitic Languages

#### Preliminaries

We continue to assume that our parser implements a structure prediction function  $h : \mathcal{X} \rightarrow \mathcal{Y}$ , where each element  $x \in \mathcal{X}$  is a sequence of space-delimited word tokens  $x = w_1, \dots, w_n$  from a set  $\mathcal{W}$  and each output elements is a constituency tree. When parsing Semitic languages, Semitic words do not uniquely determine the leaves of the parse tree (Sect. 3.1.3). Therefore, we have to alter our prediction function.

Let  $\mathcal{S}$  be a vocabulary containing a finite set of valid morphological segments in the language, and let us continue to assume a finite set of non-terminal categories  $\mathcal{N}$ . Every parse tree  $y \in \mathcal{Y}$  for a word sequence  $x = w_1 \dots w_n$  is a constituency tree over a sequence of segments  $s_1 \dots s_m$ , where  $s_i \in \mathcal{S}$  and possibly  $n \neq m$ . We assume a symbolic lexicon  $\mathcal{L}$ , which is simply a set of morphological segments associated with their corresponding part-of-speech tags, or, in short, a set of *lexemes*.

$$\mathcal{L} = \{(s, l) | s \in \mathcal{S}, l \in \mathcal{N}\}.$$

We demonstrated in Example (4) that a token  $w_i \in \mathcal{W}$  may admit to multiple morphological segmentation possibiities. The set of analyses is constrained by a

language-specific morphological analyzer  $\mathcal{M}$  where  $\mathcal{M}$  is a function from words to sets of sequences of lexemes  $\mathcal{M} : \mathcal{W} \rightarrow \mathcal{P}(\mathcal{L}^*)$ .

The morphological analysis of an input word  $\mathcal{M}(w_i)$  can be represented as a lattice  $L_i$  in which every arc corresponds to a specific lexeme  $\langle s, l \rangle \in \mathcal{L}$ . The morphological analysis of an input sequence  $x = w_1 \dots w_n$  is then a lattice  $L$  obtained through the concatenation of the lattices  $\mathcal{M}(w_1) = L_1, \dots, \mathcal{M}(w_n) = L_n$ . An example for a complete morphological analysis lattice is shown in Fig. 3.11.

Let  $\mathcal{Y}_L$  be the set of trees  $y \in \mathcal{Y}$  that dominate sequences of segments which are contained in the morphological lattice  $\mathcal{M}(x) = L$ . A parser for Semitic languages has to predict both the correct sequence of morphological segments and the parse tree that dominate this sequence.

$$h(x) = \operatorname{argmax}_{\{y \in \mathcal{Y}_L\}} p(y) \quad (3.19)$$

## Joint Probabilistic Modeling

There are two different approaches to implement (3.19), as a pipeline scenario, or as a joint prediction. In a pipeline scenario, the morphological lattice  $L$  is provided as input to a morphological disambiguation component that aims to predict the correct morphological analyses of the input word tokens.

$$\langle s_1^m, l_1^m \rangle^* = \operatorname{argmax}_{\langle s_1^m, l_1^m \rangle \in \mathcal{M}(w_1^n)} p(s_1^m, l_1^m | w_1^n) \quad (3.20)$$

Then, the most probable segmentation  $s_1^m$  enters a standard parsing system which takes that morphological segments found in (3.20) to be the tagged tree terminals.

$$y^* = \operatorname{argmax}_{y \in \{y' : \text{yield}(y') = \langle s_1^m, l_1^m \rangle^*\}} p(y | \langle s_1^m, l_1^m \rangle^*) \quad (3.21)$$

In a joint architecture, the lattice is provided as input to the parser, and the parsing algorithm aims to jointly predict the most probable parse tree and the most probable morphological segmentation. Since the morphological lattice for every sequence of words is unique, we further can simplify:

$$y^* = \operatorname{argmax}_{y \in \{y' : \text{yield}(y') \in \mathcal{M}(x)\}} p(y | \mathcal{M}(w_1^n)) \quad (3.22)$$

If we assume that all the morphological paths in the lattice are equally likely, as in [42], we are in fact selecting both the path and the segmentation based on the probability of the trees only, as in [42, 45], and we can simplify further:

$$y^* = \operatorname{argmax}_{y \in \mathcal{Y}_L} p(y) \quad (3.23)$$

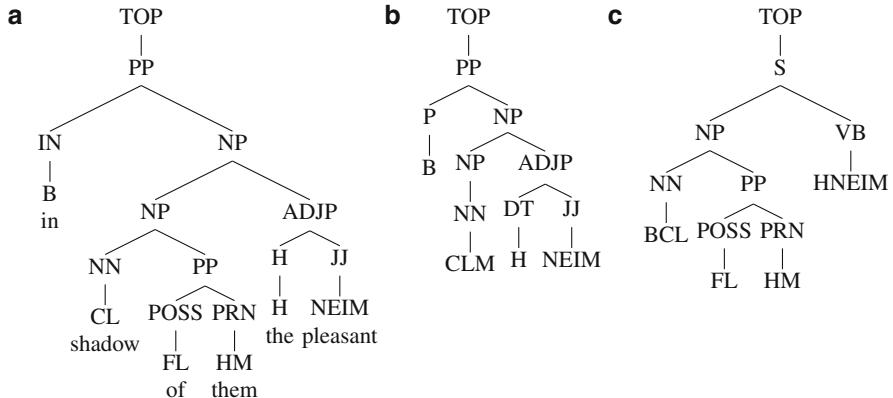


Fig. 3.10 Three possible parses of the phrase BCLM HNEIM. Tree (a) is the gold parse

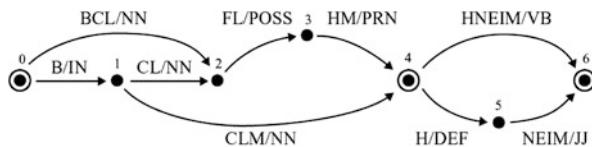


Fig. 3.11 Morphological segmentation possibilities of the phrase BCLM HNEIM. The pairs on each arc  $s/l$  indicate the segment  $s$  and a part-of-speech label  $l$ , as assigned by the lexicon  $\mathcal{L}$

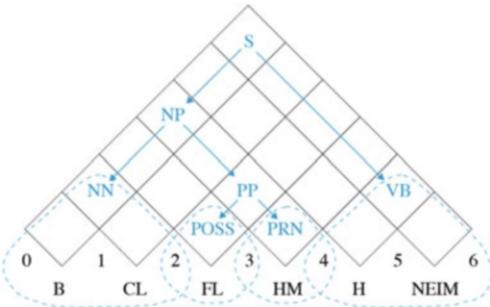
To illustrate this, consider the morphological lattice of the Hebrew phrase BCLM HNEIM, as depicted in Fig. 3.10. All trees in Fig. 3.10 dominate paths in the lattice. The goal of a joint probabilistic model is to select the most probable tree, and this tree will pick out a single path through the lattice.

### Lattice-Based Decoding

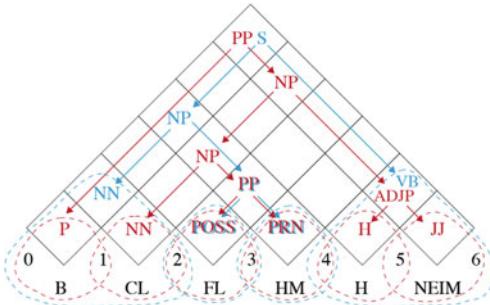
Fortunately, we can still use the CKY decoding algorithm for finding the most probable tree over a lattice, by altering a few of its underlying assumptions [42]. We can no longer assume a chart of length  $n$ , since we do not know up front the number of morphological segments in  $x = w_1^n$ . We can know the size of the morphological lattice, and we know that the the trees in  $\mathcal{Y}_{\mathcal{M}(x)}$  will always be anchored in states in the lattice  $\mathcal{M}(x)$ . So we can chart size is bounded by the number of states in the morphological lattice.  $|\mathcal{M}(x)|$ , and the lattice states define the indices of the array. Some morphological segments in the trees span more than two indices. At initiation time, we seek to fill in the part-of-speech for all the segments in the lattice, even those of which span is greater than 1.

Consider for example the chart in Fig. 3.12. It holds a tree that spans a path in the morphological lattice. Both the NN and the VB elements span more than one chart item, with no dominated daughters of span 1. In Fig. 3.13 the span [4,6] can

**Fig. 3.12** Extended CKY lattice parsing:  
segment-based indexing of  
the phrase BCLM HNEIM.  
*Dotted circles* indicate tagged  
morphological segments



**Fig. 3.13** Extended CKY lattice parsing: two possible analyses of the phrase BCLM HNEIM. *Dotted circles* indicate tagged morphological segments




---

### Algorithm 2 The CKY algorithm for lattice parsing (with back-pointers)

---

```

1: for  $\langle i, A_L, j \rangle \in \text{EDGES}(MA(x))$  do                                 $\triangleright$  traverse the morphological analysis lattice
2:    $\delta_{AL}[i, j] \leftarrow p(A_L \rightarrow s_{i,j})$                                       $\triangleright$  Initiate segments probs
3:    $\beta_{lattice_{AL}}[i, j] \leftarrow \langle s_{i,j} \rangle$                                       $\triangleright$  Store segments
4: for  $span = 2 \rightarrow n$  do                                                  $\triangleright$  Fill in the chart
5:   for  $end = span \rightarrow n$  do
6:      $begin \leftarrow end - span$ 
7:     for  $L = 1 \rightarrow |\mathcal{N}|$  do
8:        $\delta_L(begin, end) \leftarrow \max_{\langle m, J, K \rangle} p(A_L \rightarrow A_J A_K) \times \delta_J(begin, m) \times \delta_K(m, end)$ 
9:        $\beta_L(begin, end) \leftarrow \text{argmax}_{\langle m, J, K \rangle} p(A_L \rightarrow A_J A_K) \times \delta_J(begin, m) \times \delta_K(m, end)$ 
10:  return BUILD-TREE  $\delta_S[0, n]$ ,  $\beta_S[0, n]$                                           $\triangleright$  Follow back-pointers

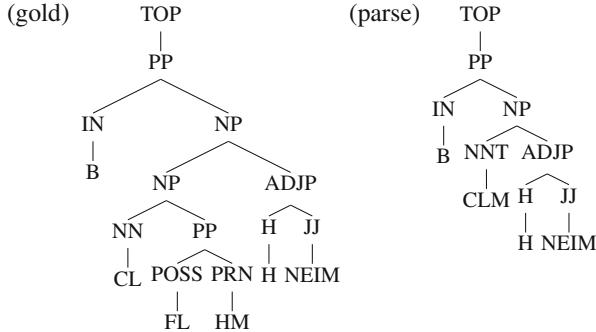
```

---

be covered by a single part of speech tag VB, or a phrase ADJP dominating two spans of length 1. At decoding time, we traverses all possible trees licensed by the grammar over all possible segmentations possibilities encoded in the lattice. The highest probability tree selects a segmentation among the alternatives. The pseudo code for the altered algorithm is given in Algorithm 2. Notably, only the initiation phase (lines 1–3) has to be altered.

## Evaluation

The structure and properties of Semitic languages pose challenges not only for parsing, but also for evaluating parser performance. Standard evaluation metrics for



**Fig. 3.14** A gold tree and a parse hypothesis for evaluation

parsing assume identity between the input word sequence and the terminals in the tree. The ParseEval metrics for evaluating constituency-based parsing [10] assume that every tree can be represented as a set of labeled spans of the form  $\langle i, L, j \rangle$  such that a labeled constituent  $L$  spans word indices  $i < j$ . If  $P$  and  $G$  are sets of such tuples for the gold and parse trees, the ParseEval scores are as follows.<sup>9</sup>

$$LP = \frac{|\hat{P} \cap \hat{G}|}{|\hat{P}|} \quad (3.24)$$

$$LR = \frac{|\hat{P} \cap \hat{G}|}{|\hat{G}|} \quad (3.25)$$

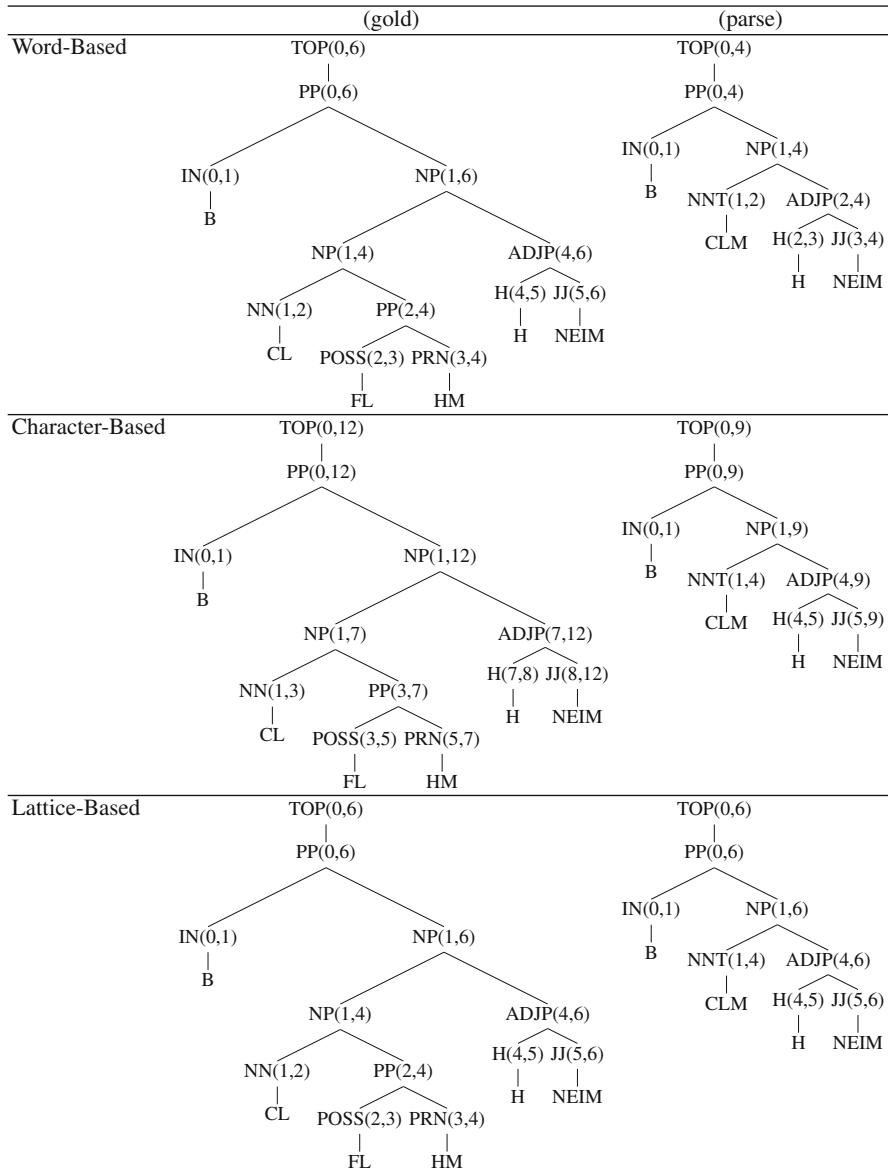
$$F_1 = \frac{2 \times LP \times LR}{LP + LR} \quad (3.26)$$

Applying ParseEval for cases where the parsed tree spans a different yield from that gold tree causes the evaluation procedure to break down. Consider for example the trees in Fig. 3.14, which all correspond to the same original raw sentence BCLM HBEIM with the morphological lattice in Fig. 3.11. The trees contain different segmentation hypotheses. The gold segmentation of BCLM HNEIM is more fine-grained than the coarse-grained segmentation hypothesis. The syntactic trees are very similar. In both of them, the overall PP and the attached NP are identified correctly. Figure 3.15 shows the word-based indexing of the gold tree and parse hypothesis. There are 5 syntactic nodes in the gold tree and 2 syntactic nodes in the parse hypothesis. The intersection of labeled spans is empty. The ParsEval score then calculates as  $F_1(\text{gold}, \text{tree}) = 0$ .

To circumvent this, early studies on parsing Modern Hebrew propose a generalized version of ParseEval, in which tuples of the form  $\langle i, L, j \rangle$  indicate character

---

<sup>9</sup>The standard implementation of ParseEval, Evalb, is available at <http://nlp.cs.nyu.edu/evalb/>.



**Fig. 3.15** Comparing the gold segmentation, tree to the predicted segmentation, tree. The different rows demonstrate different ways of indexing. The lattice-based indices refer to the lattice in Fig. 3.11

indices [42, 100]. This solution does not solve the problem adequately, since many morphological processes go beyond concatenation and the sequence of characters themselves may be misaligned. Figure 3.15 shows the character-based indexing of

the gold tree and parse tree in our example. Because of an incorrect segmentation hypothesis, some phonological forms are not identified, and the sequences of characters are of different lengths. Again, the intersection of labeled spans is empty, and the ParseEval score is again 0.

In both cases, the metrics reduce the structures to sets. We can instead use similarity metrics that measure the differences between the trees themselves, rather than their set reduction. Several studies address this evaluation challenge by correcting the hypothesized segmentation using string edit operations, and then score the parse tree dominating a corrected segment [28, 90]. This method has been used successfully to evaluate parse trees over speech lattices, where the input signal is ambiguous. However, in both implementations, string edit corrections may lead to discarding syntactic nodes that were dominating removed segments, so the overall result may be biased. In our example, changing the parse segmentation into the gold segmentation involves deleting segment CLM and adding segments CL FL HM. The normalized string edit distance is then  $\frac{1+3}{6+4} = 0.4$ . Changing the gold segmentation into the parsed one we would remove CL FL HM and add BCLM, and get the same segmentation error. Note that in the first case the trees are much larger than in the second case, and the normalization of the tree scores are different. These normalization changes distort the empirical results which are based on such string-edit corrections.

To solve this challenge, Tsarfaty et al. [107] use tree-edit distance metrics to score trees over morphological lattices directly.<sup>10</sup> TedEval requires a set  $\mathcal{A}$  of operations such that, for every  $y_1, y_2 \in \mathcal{Y}_x$ , there is some sequence of operations  $\langle a_1, \dots, a_m \rangle$  ( $a_i \in \mathcal{A}$ ) turning  $y_1$  into  $y_2$ . The set  $\mathcal{A}$  contains four edit operations: ADD( $l, i, j$ ) and DEL( $l, i, j$ ) that adds and deletes a nonterminal node with label  $l \in \mathcal{N}$  that dominates the span from state  $i$  to state  $j$  in the lattice  $L$ ; ADD( $\langle s, t \rangle, i, j$ ) and DEL( $\langle s, t \rangle, i, j$ ) that adds and deletes a lexeme, that is, a pair consisting of a preterminal node labeled  $p \in N$  and a terminal node labeled  $s \in T$ . These operations are properly constrained by the lattice  $\mathcal{M}(x)$ , that is, we can only add and delete lexemes that belong to  $\mathcal{L}$ , and we can only add and delete them between states where they are allowed to occur in the lattice.

We define the cost of an operation sequence  $\langle a_1, \dots, a_m \rangle$  as the sum of the costs of all operations in the sequence  $C(\langle a_1, \dots, a_m \rangle) = \sum_{i=1}^m C(a_i)$  according to some cost function  $C : \mathcal{A} \rightarrow \mathbb{R}^+$ . An *edit script*  $ES(y_1, y_2) = \langle a_1, \dots, a_m \rangle$  is a sequence of operations from  $\mathcal{A}$  that turns  $y_1$  into  $y_2$ . The *minimum edit distance*  $MED(y_1, y_2)$  is the minimal cost edit script:

$$MED(y_1, y_2) = \min_{ES(y_1, y_2)} C(ES(y_1, y_2))$$

The error of a predicted structure  $p \in \mathcal{Y}_x$  with respect to a gold structure  $g \in \mathcal{Y}_x$  is defined to be the edit distance  $MED(p, g)$ . The score of a predicted structure  $p$  with

---

<sup>10</sup>The software may be downloaded at <http://stp.lingfil.uu.se/~tsarfaty/unipar/download.html>.

respect to the gold  $g$  is defined by normalizing the distance and subtracting it from a unity:

$$\text{EVAL}(p, g) = 1 - \frac{\text{MED}(p, g)}{\text{N}(p, g)}$$

The normalization factor  $\text{N}(p, g)$  is normally defined as the worst possible MED given the sizes of  $p$  and  $g$ . In the current setting, this corresponds to the case where every all the nodes and lexemes in  $p$  have to be removed and all the nodes and lexemes in  $g$  have to be inserted, so we interpret the measure as the size of the tree ( $|x|$  discards the root node, following common practice).

$$\text{EVAL}(p, g) = 1 - \frac{\text{MED}(p, g)}{|p| + |g|}$$

In our example, can use tree edit distance to turn the parsed hypothesis into the gold tree directly. For that we assume the lattice-based indices as depicted at the bottom of Fig. 3.15. We now have to apply the following operations over the lattice-indexed lexemes and nodes: DEL(NNT/CLM,1,4), ADD(NN/CL,1,2), ADD(POSS/FL,2,3), ADD(PRN/HM,3,4), and ADD(PP,2,4). The number of nodes and lexemes in the gold tree is 11 and in the parsed hypothesis it is 7 (discarding the roots). TedEval yields the following score:

$$\text{EVAL}(p_1, g_1) = 1 - \frac{5}{7 + 11} = \frac{13}{18}$$

The TedEval score, takes into account both nodes and lexemes. It normalizes the metrics according to the size of trees with the corresponding segmentation, and thus provides a less biased score than those that reduce structures into sets. In our example, the score given by TedEval reflects the fact that the system identified correctly two of the three major components in the sentence – the overall PP and a modified NP, and it avoids penalizing syntactic errors over misaligned segmentation.

## Summary and Conclusion

We presented an architecture for parsing Semitic languages in which we assume a morphologically ambiguous input signal and predict a morphological segmentation jointly with the parse tree. We have further shown an adaptation of the CKY decoding algorithm to such architectures, and demonstrated ways for evaluating parsing results over ambiguous input. The architecture we presented here underlies the best performing constituency parsers for Modern Hebrew and Modern Standard Arabic. It provides a general framework for parser development, and the parser developer to use different kinds of probabilistic grammars for disambiguation. In the next section we show how to learn probabilistic grammars that can effectively cope with the rich morphosyntactic interactions reflected in Semitic data.

### 3.2.3 The Syntactic Model

#### PCFG Refinements

The greatest limitation of PCFG-based statistical parsing models is the context-freeness assumption inherent in the formalism [93]. This assumption imposes independence between model parameters – the probabilities of rules that are used to derive the syntactic trees. This independence assumption is both too strong and too weak for natural language data. Too strong, because the context of a node may have implications for the rule that can be applied in this position, and too weak, because a node sequence may have to be permuted when aiming to parse unseen data. This section discusses techniques that effectively address these two types of limitations. We present methods that involve (semi)-manual tuning of model parameters, automatic tuning of model parameters, and a complete remodeling of the trees. We then discuss constrained parsing, and we finally comment on discriminative methods that discard such independence assumptions altogether. This section concentrates on theoretical foundations. The empirical results obtained by different models for Semitic languages are detailed in Sect. 3.3.

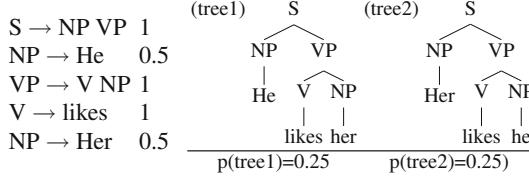
#### History-Based Modeling

In simple PCFG-based models, the parameters are the weights of grammar rules of the form  $A \rightarrow \alpha$ . These parameters may be viewed as instantiating a history-based model, where every decision  $d_i$  takes the form of substituting  $A$  into  $\alpha$ . The function  $\psi$  selects the symbol  $A$  at the substitution site as conditioning context.

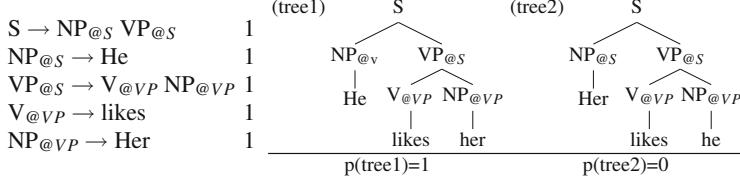
$$h(x) = \operatorname{argmax}_{y \in \mathcal{Y}_x} \prod_{A \rightarrow \alpha \in \pi} p(A \rightarrow \alpha | A) \quad (3.27)$$

The multiplication of such parameters imposes independence assumptions that are too strong for natural language parsing. Consider the sentence “He likes her” as annotated in Fig. 3.16. The probability assigned to the tree according to the PCFG on the left is equal to the probability of the tree for the ungrammatical sentence “Her likes he” on the right. This PCFG model is said to *over-generate*, that is, it assigns probability mass to sentences that should not be allowed by an English grammar.

Johnson [52] observes that it is easy to alter such independence assumptions by relabeling each category label with its parent category. By adding parent information, we enrich the conditioning context for rule application, and this, in turn, creates a relation between the generated nodes and higher levels of the tree. This modeling choice may be seen as a slightly more complex instantiation of the function  $\psi$  with the current label decorated with its immediately dominating parent. (We use the  $X@A$  notation to indicate elements in the syntactic environment of a node  $A$ .)



**Fig. 3.16 Over generalization of a simple PCFG.** Both (tree1) and (tree2) may be generated by the probabilistic grammar on the *left*, and they are assigned probability = 0.25



**Fig. 3.17 Parent Annotation of a simple PCFG.** The tree (tree1) is the only tree generated by the probabilistic grammar on the *left*. The tree (tree2) is assigned probability = 0

---

### Algorithm 3 The Transform-Detransform method for history-based PCFG parsing

---

```

1: trans-trainset  $\leftarrow$  TRANSFORM(original-trainset)                                 $\triangleright$  enrich treebank trees
2: grammar  $\leftarrow$  TRAIN(trans-trainset)                                          $\triangleright$  obtain a probabilistic grammar
3: for  $i = 1 \rightarrow n$  do                                                  $\triangleright$  traverse the test-set
4:    $tree_i \leftarrow$  DECODE(grammar,  $x_i$ )                                          $\triangleright$  parse the input sentence
5:    $y_i \leftarrow$  DE-TRANSFORM( $tree_i$ )                                          $\triangleright$  discard enrichment
6: return  $\{y_i\}_1^n$                                                         $\triangleright$  return prediction

```

---

$$h(x) = \operatorname{argmax}_{x \in \mathcal{Y}_x} \prod_{A \rightarrow \alpha \in x} p(A \rightarrow \alpha | A, \text{parent}@A) \quad (3.28)$$

Consider the annotation of parent category for every node in the trees, as in Fig. 3.17. Here the annotation distinguishes NPs that occur in subject position ( $NP@S$ ) from NPs that occur in object position ( $NP@VP$ ). A model can thus learn to generate the right kind of pronoun at each position of the tree. This helps to differentiate the grammatical sentence on the left from the ungrammatical sentence on the right of Fig. 3.17, which is now assigned zero probability.

In technical terms, Johnson proposes a parsing architecture based on tree transformation, as detailed in Algorithm 3. He first transforms the training trees to their new format, obtains a probabilistic grammar from the transformed trees, uses the treebank grammar to parse the test data, and reverses the transformation of the parse trees prior to evaluation. For English parsing, it turns out that a parsing model obtained in this way improves broad-coverage parsing of newswire texts substantially [52]. Klein and Manning [59] further show that annotating grand-parent information for some of the nodes in the treebank further improves English parsing accuracy.

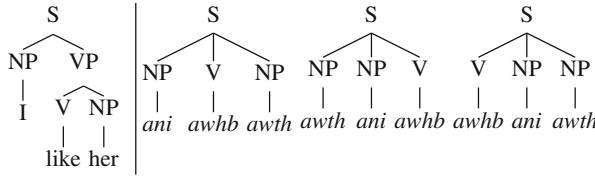


Fig. 3.18 Syntactic analyses in English and Hebrew transitive sentences

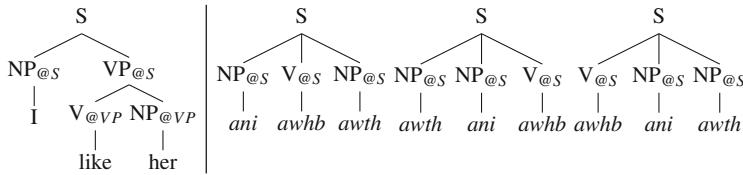


Fig. 3.19 Parent annotation features in English and Hebrew transitive sentences

Would this modeling choice have the same effect when parsing another language? Recall that Semitic languages have a more flexible phrase structure than sentences in English. Thus, the Hebrew version of the phrase for “I like her” could occur as *ani awhb awth*, *awth ani awhb* and *awhb ani awth* (see Fig. 3.18). When applying the parent annotation procedure to the trees in this example (see Fig. 3.19) we do not distinguish the distribution of phrases hanging after S, and thus it is not expected to improve the disambiguation capacity for such structures.

Empirically, parent encoding was shown to improve parsing Semitic languages to a lesser extent than parsing English (see Sect. 3.3). This procedure mainly benefited constructions that exhibit stricter word-order, such as relative clauses and modified noun phrases. In order to parse Semitic languages more accurately, one has to provide a parsing method that can cope with word-order freeness and long flat CFG rules.

### Head-Driven Models

The capacity to generate trees with flexible phrase structures crucially depends on the ability of a grammar to generalize from observed rules to new rules. Assume for instance that during training we observed the parse tree for *ani awhb awth* (I like her) which uses the rule  $S \rightarrow NP\ VP\ NP$ . Also assume that the decoder needs to find the correct parse for the input sentence *awth ani awhb* (her I like) which involves the structure  $S \rightarrow NP\ NP\ VP$ . If we have not seen this rule permutation in the training data, we will not be able to generalize from the former structure to the latter.

This is a problem of *under-generation*, and it leads to a lack of *coverage*. A common way to increase coverage is to break down the generation of standard rules into smaller pieces that can recombine to form rules that were unseen during

training (much like we do in binarization Sect. 3.2.1). To solve this problem for English, Collins and others [22, 30, 70] proposed the *Head-Driven* modeling method. In this modeling method we view a context-free rule of the form  $A \rightarrow \alpha$ , as a complex object where we distinguish a *head* element  $H$  from its left and right sisters.<sup>11</sup>

$$A \rightarrow L_n \dots L_1 H R_1 \dots R_m$$

Head-driven models break down the generation of the daughters' sequence into incremental steps, generating the sisters from the head outwards, one sister at a time.

$$h(x) = \operatorname{argmax}_{x \in \mathcal{Y}_x} \prod_{A \rightarrow \alpha \in x} p(A \rightarrow L_n \dots L_1 H R_1 \dots R_m | A) \quad (3.29)$$

$$\begin{aligned} &= \operatorname{argmax}_{x \in \mathcal{Y}_x} \prod_{A \rightarrow \alpha \in x} p(H | A) \times \prod_{i=1}^n p(L_i | A, H, L_1 \dots L_{i-1}) \\ &\quad \times \prod_{i=1}^m p(R_i | A, H, R_1 \dots R_{i-1}) \end{aligned} \quad (3.30)$$

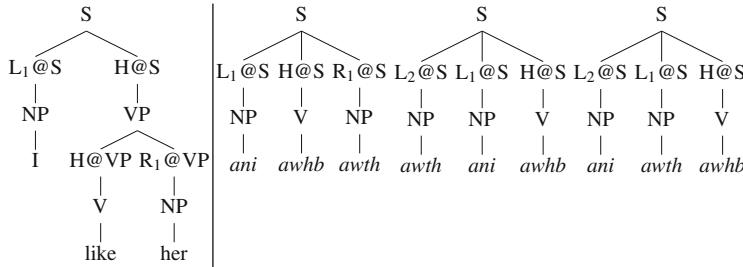
$$\begin{aligned} &= \operatorname{argmax}_{x \in \mathcal{Y}_x} \prod_{A \rightarrow \alpha \in x} p(H | A) \times \prod_{i=1}^n p(L_i | \psi_L(A, H, L_1 \dots L_{i-1})) \\ &\quad \times \prod_{i=1}^m p(R_i | \psi_R(A, H, R_1 \dots R_{i-1})) \end{aligned} \quad (3.31)$$

The model in (3.31) is another instance of history-based modeling. For each node generation we select conditioning context using a history-mapping function  $\psi$  that chooses relevant information from the sisters that we have already generated. For instance, the  $\psi$  functions can select immediately preceding daughters in a Markovian generation process. A 0-order Markovized head-driven process will consider only the parent and the head daughter as conditioning context.

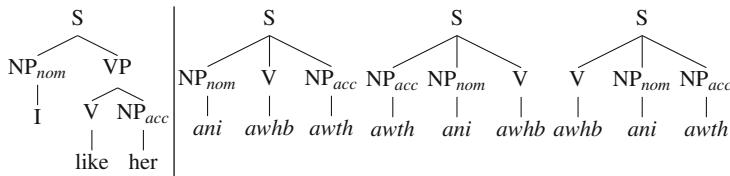
$$h(x) = \operatorname{argmax}_{x \in \mathcal{Y}_x} \prod_{A \rightarrow \alpha \in x} p(H | A) \times \prod_{i=1}^n p(L_i | A, H) \times \prod_{i=1}^m p(R_i | A, H) \quad (3.32)$$

A 1st-order Markovized process records one immediately preceding daughter in addition to the head, a 2nd-order process records two immediately preceding

<sup>11</sup>A head is a linguistic notion coined in the generative linguistic tradition. For instance, in the rule  $VP \rightarrow V NP$ , the verb is considered the head of the phrase. The criteria for selecting heads are subject to debates (see, for instance, the discussion in [109]), here we leave head-selection criteria out of the discussion and assume that deterministic rules for finding heads can be defined.



**Fig. 3.20** Head-outward generation of nodes in English and Hebrew trees



**Fig. 3.21** Morphological features of syntactic nodes in English and Hebrew trees

daughters, and so on. Higher-order Markovian processes are also conceivable but they impose stronger dependencies between the different daughters of a CFG rule, which greatly undermines its generalization capacity.<sup>12</sup>

Head-driven lexicalized parsing<sup>13</sup> has been implemented in a language-independent parsing engine by Bikel [8], and this model was later applied to different languages with partial success. The benefits of this modeling method have not been as significant for Semitic languages parsing as they were for parsing English (see Sect. 3.3). When the word-order is flexible, there is no significant correlation between the position of noun phrases and the form of the different nouns, as can be seen in Fig. 3.20. A more systematic correlation occurs between the form of nouns and their case assignment, as is illustrated in Fig. 3.21. A more helpful enrichment here would then be to encode morphological case, rather than syntactic position.

Parent encoding on top of syntactic categories and Markovization of CFG rules are two instances of the same idea, that of encoding the generation history of a node. The different history-mapping functions  $\phi$  and  $\psi$  correspond to two dimensions that define the parameters' space for parsers [59]: the *vertical dimension*

<sup>12</sup>Additional information may also be added to the  $\psi$  function. Collins [30] includes punctuation and verbal elements in the conditioning context of model 1, and information concerning subcategorization frames (the expected sisters of the verb) in model 2. In all cases, the model continues to assume a strict separation between generating the left and right sisters.

<sup>13</sup>In the original head-driven models, each upper case symbol  $A, H, L_i, R_i$  in fact signals the category of the node and the lexical head information. In unlexicalized parsing, lexical items are omitted, and only the categories remain.

( $v$ ) captures the history of the node’s ancestors (its parent and grandparents), and the *horizontal* dimension ( $h$ ) captures the horizontal ancestors of a node (its sisters) in a head-outward generation process. By varying the value of  $h$  and  $v$  along this two-dimensional grid and following the recipe of Algorithm 3, Klein and Manning [59] showed that both horizontal and vertical history can improve parsing performance of unlexicalized treebank grammars considerably.

### Three-Dimensional Parameterization

The two-dimensional space provides information concerning the position of a node in the tree. In the case of Semitic languages, the presence or absence of a morphological feature may substantially affect the probability of generating certain nodes in these positions. Encoding morphological information on top of syntactic category labels is technically referred to as a *state-split*. In English, for instance, the verbal category VB has splits such as VBS and VBG distinguishing singular verbs and gerunds respectively. Both VBG and VBS share some behavior with the VB category, but they exhibit different behavior in different syntactic contexts.

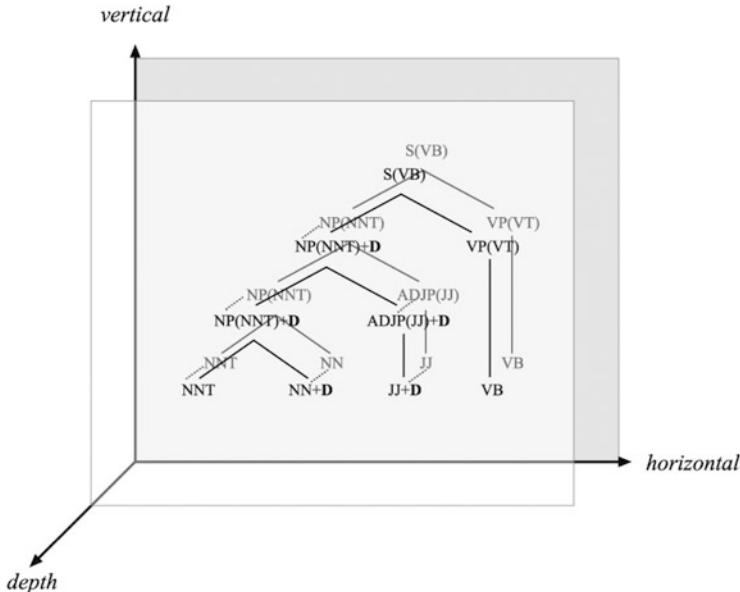
Tsarfaty and Sima’an [102] describe a universe in which tree structures enriched with morphological features reside deeper along a third dimension they refer to as *depth* ( $d$ ). This dimension can be thought of as encoding aspects of morphological analysis orthogonal to the  $v/h$  dimensions. Agreement features, for instance, apply to a set of nodes all at once. Figure 3.22 illustrates an instantiation of  $d$  with a single definiteness feature (**D**). Figure 3.23 selects an NP node from Fig. 3.22 and shows its expansion in three dimensions. Taking a history-based perspective again, we view the function  $\xi$  as selecting morphological aspects of the structure generated so far.

$$h(x) = \operatorname{argmax}_{x \in \mathcal{Y}_x} \prod_{d_i \in x} p(d_i | \phi(d_0 \circ \dots \circ d_{i-1}), \psi(d_0 \circ \dots \circ d_{i-1}), \xi(d_0 \circ \dots \circ d_{i-1})) \quad (3.33)$$

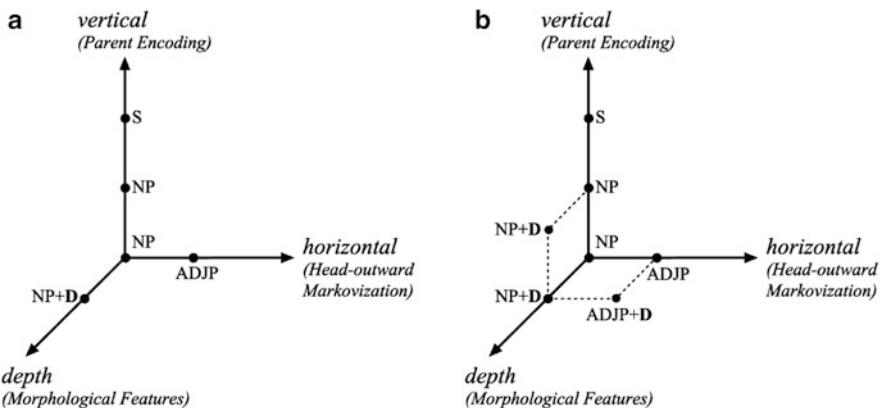
In parsing engines such as [8, 30] it is often the case that  $\xi(d_0 \circ \dots \circ d_k) = \emptyset$ , since English does not benefit much from incorporating morphology. For Semitic languages, where morphological information interacts with syntax more closely, discarding morphological state-splits runs the risk of loosing important information.

There are different ways in which morphological state-splits may be explicitly incorporated into the parsing architecture.

- **Treebank grammars.** This technique involves repeating Algorithm 3 for different combinations of conditioning information, and manually tuning the choice of rule enrichment on a development set, as in Klein and Manning [59]. This is a simple and intuitive way to set a strong baseline. However, it relies on linguistic intuitions, and finding the optimal combination may be labour intensive.



**Fig. 3.22** The three-dimensional parametrization space for treebank-grammar encoding



**Fig. 3.23** (a) Local history of node generation in three dimensions. (b) An orthogonal dimension of correlated splits: looking at a complex NP phrase in a three-dimensional space

- **Probabilistic feature grammars.** Goodman [44] proposes a history-based model in which features are generated one by one, conditioned on previously generated features. The choice of features, their order and the independence between them are determined by a human expert and are tuned on a development set.

- **Coarse-to-fine parsing.** The coarse-to-fine parsing scheme is a popular way to approach decoding when a huge space of split categories is defined [24]. In this approach one learns multiple treebank grammars with increasing depth. The first decoding round uses a coarse-grained treebank grammar, and later passes use more elaborate treebank grammars, but the search is constrained by the most probable coarse-grained structures. This method avoids the need for human interference. However, it runs the risk that good parses will be pruned too early.

### Automatic State-Splits

We have so far exploited the multidimensional space of parameters by explicitly augmenting node labels with information concerning parents, position, morphology, and so on. These techniques require the parser developer to explicitly define the different dimensions, annotate the information in treebank data, and retrain the treebank grammar. This approach is simple and intuitively appealing but it requires investment of human efforts, firstly in adding extra annotations, and secondly in finding the optimal combination of annotated features. Is it possible to search for an optimal set of parameters automatically, without human intervention?

Studies such as [75, 85, 86] explore the possibility of employing general machine learning techniques for finding grammars with performance-improving state-splits. These grammars assume that each non-terminal symbol  $C$  can have different instantiations (splits)  $C_1, C_2, C_3 \dots$  which are used in different contexts. These split-states encode latent information, which is hidden from the point of view of the model. The only split criteria that matter are those that empirically improve parsing performance on the original coarse-grain categories. These state-split grammars are also called latent annotation PCFGs (in short, PCFG-LA). The technique of obtaining accurate PCFG-LA parsers has been mastered and excelled by Petrov et al. [85], delivering a language-independent parsing framework.

The PCFG-LA model runs as follows. In the training phase, the model learns an accurate state-split grammar from a given treebank. It then uses a coarse-to-fine decoding algorithm to efficiently parse unseen data. Training a state-split grammar in the system of Petrov et al. [85] starts off with a completely binarized treebank with coarse-grain phrase labels. It then follows a cycle that repeats the following steps: (i) split (ii) merge, and (iii) smooth.

- At the **split** phase, every existing label is split into two categories. A rule of the form  $A \rightarrow B C$  now has eight instantiations:  $A_1 \rightarrow B_1 C_1, A_1 \rightarrow B_1 C_2, A_1 \rightarrow B_2 C_1, A_1 \rightarrow B_2 C_2, A_2 \rightarrow B_1 C_1, A_2 \rightarrow B_1 C_2, A_2 \rightarrow B_2 C_1, A_2 \rightarrow B_2 C_2$ .

An Expectation Maximization procedure [33] then applies to set the split-rule probabilities. In the Expectation step, one computes the posterior probability of each annotated rule in each position in each training tree. In the Maximization step, these posterior probabilities are used as weighted observations, and we update the rule probabilities to their observed relative frequencies.

- Not all splits are equally useful. Furthermore, too many splits may lead to overfitting. At the **merge** phase, state-splits that are found not to be useful according to an empirical gain criteria are merged back. Merging is followed by an EM procedure that sets the probabilities of the rules in the merged grammar.
- While the grammar considers the label-splits as disjoint states, they still share some of their behavior with the original  $C$  type category. So, at the **smooth** phase some of the probability of the split labels  $C_1, C_2$  etc. is passed back to the original label  $C$ ,

At decoding time, the score of each split rule is calculated by marginalizing out its splits. For efficiency reasons, multi-level coarse-to-fine pruning is used [84].

The PCFG-LA implementation of Petrov et al. [85] is publicly available<sup>14</sup> and has been applied to different languages (English, German, Chinese, French and Arabic) obtaining good results. Applying it out of the box for Semitic language data deserves certain extra considerations:

- Goldberg and Elhadad [38] found that using the PCFG-LA training procedure over fine-grained categories does not yield major improvements. This is because the training procedure cannot merge splits that were not performed by the system. In order to obtain significant improvements, the treebank labels that we start with have to be sufficiently coarse-grain.
- The available PCFG-LA implementations assume a procedure for unknown words handling [85, footnote 3]. This method is tailored for English, where upper case, digits and suffixes are good predictors of syntactic categories. Semitic languages show a different behavior, and adding a specialized unknown words treatment is indispensable [4].
- The default training setup assumes 2nd-order vertical Markovization and 1st-order horizontal Markovization. For languages with more flexible ordering, setting 1st-order vertical Markovization and 0th-order horizontal Markovization is preferred [38]. The generation of daughters is then conditioned on the parent only, and the automatic state-splits implicitly encode ordering preferences of sisters.

## Relational-Realizational Parsing

The previous modeling strategies we discussed assume that the probabilistic grammar defines the probabilities of phrase-structure trees, and hand-coded constraints encode linguistic (hard or soft) constraints and derive their grammatical functions. It is possible to define the probabilistic grammar itself as a set of soft morphosyntactic constraints that ensures a coherent predicate argument-structure by mapping grammatical functions to phrase-structure trees. The predicate-argument

---

<sup>14</sup>The Java implementation can be downloaded from <http://code.google.com/p/berkeleyparser/>. The extended implementation including lattice parsing was made available by Goldberg and Elhadad [39] and can be downloaded at <http://www.cs.bgu.ac.il/~yoavg/software/blatt/>.

structure may be read off directly from the trees, without the need to post-process the data. In addition, the functional information can help to improve the quality of the tree prediction. This is the motivation underlying the Relational-Realizational modeling framework proposed by Tsarfaty [101] and it is applied to modeling Semitic phenomena in Tsarfaty et al. [103–105].

The Relational-Realizational model assumes a grammar called RR-PCFG which is a tuple  $RR = \langle \mathcal{T}, \mathcal{N}, \mathcal{GR}, S \in \mathcal{N}, \mathcal{R} \rangle$  where the PCFG is extended with a finite set of grammatical relations labels  $\mathcal{GR} = \{gr_1, gr_2, \dots\}$  such as subject, predicate, object, modifier etc. ( $\mathcal{GR} \cap \mathcal{N} = \emptyset$ ). The set  $\mathcal{R}$  contains three kinds of rules which alternate the generation of structural and functional notions:

$$\mathcal{R} = \mathcal{R}_{projection} \cup \mathcal{R}_{configuration} \cup \mathcal{R}_{realization}$$

Let us assume that  $C_P \rightarrow C \dots C$  is a context-free production in the original phrase structure tree where each daughter constituent  $C_i \in \mathcal{N}$  has the grammatical relation  $gr_i \in \mathcal{GR}$  to the parent category. The set  $gr_1 \dots gr_n$  is the *relational network* (a.k.a. the *projection* or *subcategorization*) of the parent category. Each grammatical relation  $gr_i @ C_P$  is realized as a syntactic category  $C_i$  and carries morphological marking appropriate to signaling this grammatical relation. The RR grammar articulates the generation of such a construction in three phases:

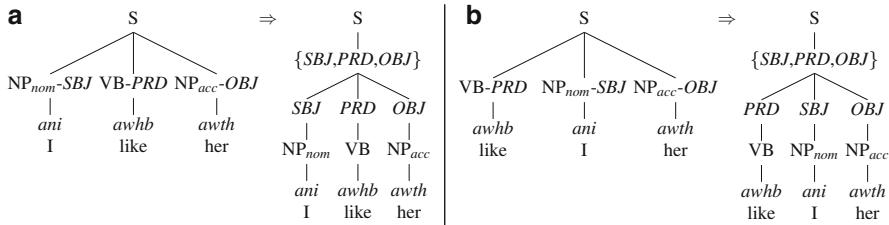
- **Projection:**  $C_P \rightarrow \{gr_i\}_{i=1}^n @ C_P$   
The projection stage generates the function of a constituent as a set of grammatical relations (henceforth, the relational network) between its subconstituents.
- **Configuration:**  $\{gr_i\}_{i=1}^n @ C_P \rightarrow gr_1 @ C_P \dots gr_n @ C_P$   
The configuration stage orders the grammatical relations in the relational network in a linear order in which they occur in the tree.<sup>15</sup>
- **Realization:**  $gr_i @ C_P \rightarrow C_i$   
In realization, every grammatical relation generates the morphosyntactic representation of the child constituent that realizes the already-generated function.

The RR model is a recursive history-based model, where every constituent node triggers a generation cycle, conditioned on this node category. Every time the projection-configuration-realization cycle is applied, the probability of this constituent node is replaced with the probabilities of the three stages, multiplied:

$$\begin{aligned} \mathbf{P}_{RR}(r) = & \\ \textit{Projection} \quad & \mathbf{P}_{projection}(\{gr_i\}_{i=1}^n | C_P) \times \\ \textit{Configuration} \quad & \mathbf{P}_{configuration}(\langle gr_0 : gr_1, g_1, \dots \rangle | \{gr_i\}_{i=1}^n, C_P) \times \\ \textit{Realization} \quad & \prod_{i=1}^n \mathbf{P}_{realization}(C_i | gr_i, C_P) \times \\ & \mathbf{P}_{adjunction}(\langle C_{0_1}, \dots, C_{0_{m_0}} \rangle | gr_0 : gr_1, C_P) \times \\ & \prod_{i=1}^n \mathbf{P}_{adjunction}(\langle C_{i_1}, \dots, C_{i_{m_i}} \rangle | gr_i : gr_{i+1}, C_P) \end{aligned}$$

---

<sup>15</sup>The configuration phase can place realizational slots between the ordered elements signaling periphrastic adjunction and/or punctuation. See further details in the original publication [103].



**Fig. 3.24** Generating canonical and non-canonical configurations using the RR model: S level CFG productions at the LHS of (a) and (b) are different. The RR-CFG representations at the RHS of (a) and (b) share the projection and realization parameters and differ only in their configuration

The first multiplication implements an independence assumption between grammatical relations and configurational positions. The second multiplication implements an independence assumption between grammatical positions and morphological markings of the nodes at that position. These assumptions are appropriate for languages with flexible word order and rich morphology. The later multiplications implement independence between the generation of complements and the generation of adjuncts.<sup>16</sup>

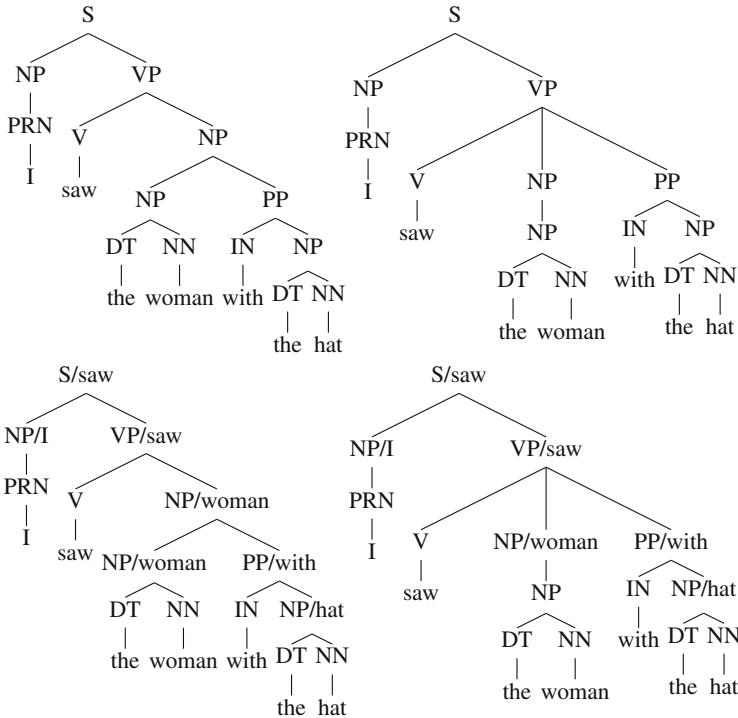
Figure 3.24 shows the RR generation process for the different clauses we discussed before. In their simple context free representation, there is no parameter sharing between the two constructions. In the RR representation, the trees share the *projection* and *realization* parameters, and only differ in the *configuration* parameter, which captures their alternative word-ordering patterns. By making the commonalities and differences between the clauses explicit, the grammar can create new rules for unseen examples.

Training RR grammars confines with the transfrom-detransform method outlined in Algorithm 3. It starts off with a set of phrase structure trees in which every node specifies the phrase-label, grammatical function, and morphological features of that node. The treebank then goes through a transformation which includes: (i) separating grammatical functions from their realization, (ii) separating the position of every relation from its morphological marking. Training can be done in a standard way, using maximum likelihood estimates. Since the resulting grammar assumes independence between the rules, we can use a simple chart parser over sequences or over lattices to parse unseen sentences.

### Lexicalization and Factored Modeling

Lexical tendencies provide cues that are orthogonal to syntactic information. Consider the alternative English trees for the sentence "I saw the woman with

<sup>16</sup>Discussion of the distinction between complements and adjuncts is omitted here, refer to [101] for further details.



**Fig. 3.25** Unlexicalized and lexicalized syntactic analyses of “I saw the woman with the hat”

the hat.” in Fig. 3.25. The analyses on the left and right are syntactically equally plausible. There is a stronger affinity of the lexical items “woman with hat” than that of “seeing with hat”, which makes us pick out the analysis on the left as the correct analysis. In order to capture such tendencies, it is customary to *lexicalize* the parse trees. That is, for each node we add information concerning the lexical head of the dominated constituent, as demonstrated at the bottom of Fig. 3.25. The trigram “woman with hat” may be explicitly read off from the left-hand tree, while “saw with hat” is read off from the right hand tree. If we can capture these lexical affinities in the parsing model, it may help us choose between competing analyses.

Learning treebank grammars from lexicalized trees is hard due to extreme data sparseness. Klein and Manning [58] suggest instead to parse with a factored model of syntactic and lexical information, providing a practical way to model the predicate-argument structure of a sentence as orthogonal to its tree structure.

Formally, the factored *constituency/dependency* model assumes an agenda-based parser that guides the search through the space of structures  $y \in \mathcal{Y}$  looking for the most likely lexicalized tree  $L = \langle y, d \rangle$  with  $y \in \mathcal{Y}$  a phrase-structure tree  $d \in \mathcal{D}$  a dependency structure projected by the phrase structure. Now, if  $Z \rightarrow X Y$  is a CFG rule and  $h = \text{head}(Z) = \text{head}(X)$  and  $h' = \text{head}(Y)$ , the

parser seeks the most probable pair of phrase structure and dependency structure by multiplying the different factors.<sup>17</sup> The word-level dependency information in the model is formally incorporated through the probability distribution  $p(h'|h)$ , and it is completely independent of structural information coming from the PCFG productions  $p(XY|Z)$ .

$$\langle y, d \rangle^* = \arg \max_{(y,d)} p_{factored}(y, d | x) = \prod_{\{Z_h \rightarrow X_h Y_{h'}\}_{i=1}^n} p_{cfg}(XY|Z) \times p_{dep}(h'|h)$$

This model underlies the Stanford Parser<sup>18</sup> which has been applied successfully to parsing English, Chinese, German and Arabic. The application of the model for a new language requires a specification of the head-finding rules (and possibly language-specific unknown-words treatment). At the same time, it is not clear that the model can effectively capture morphological-syntactic interactions, because morphological information captured in the word-forms  $h, h'$  is orthogonal to syntactic configurations. A sound solution for this problem is factored modeling of the lexicon, as described in Green et al. [46] (see also Sect. 3.2.4).

## Constrained Parsing

From an empirical point of view, it is unclear that state-splits learned for non-terminals symbols will constrain the distribution of trees sufficiently. The grammar may still over-generate. In order to enforce grammatical structures it is possible to impose hard-coded linguistic constraints.

- **Parse Selection:** Instead of providing a single output structure, most parsers can provide a set of  $n$  most probable trees. It is possible to devise hand-coded linguistic rules that pick out the first tree that does not violate any constraint. In this way we can discard trees that violate morphological case constraints or agreement. This view of parsing is reminiscent of the Optimality Theory (OT) framework of Prince and Smolensky [87], in which the grammar is composed of a GEN component and a set of violable constraints. According to them, a grammatical structure is one that violates the least amount of constraints.
- **Parse Reranking:** Re-ranking may be viewed as a refinement of the parse selection idea. In reranking frameworks, rather than selecting the first tree that obeys a set of hard constraints, we encode linguistic information as soft constraints. The parses in the n-best list are re-ranked according to the weights of linguistic features of the structure, and the weights are trained using a discriminative framework (see Sect. 3.2.3). The best tree according to the re-ranked model is selected as the output parse-tree [23].

---

<sup>17</sup>The probability model defined by the formula is deficient [60, footnote 7].

<sup>18</sup><http://nlp.stanford.edu/software/lex-parser.shtml>

- **Chart Pruning** The gold tree is not guaranteed to be found in the n-best list. A different way to incorporate constraints is by way of pruning trees during the search. In chart pruning, one observes every rule in the chart and prunes edges that violate linguistic constraints (e.g., morphological agreement). This view of parsing is akin to theoretical frameworks such as Lexical Functional Grammar (LFG) or Head-Driven Phrase-Structure Grammar (HPSG) that view tree construction as a constraint satisfaction problem [88].

## Discriminative Approaches

Encoding non-local features in a history-based model results in fine-grained states which are hard to statistically learn using a limited amount of training data. A different approach to defining probability distributions over structures popularity is based on discriminative methods.

One such approach is based on *Maximum Entropy (MaxEnt)* models [6]. MaxEnt models may be defined over arbitrary graphical representations, and it need not assume any independence between different sub-events in the representation of the tree. All information in MaxEnt models is expressed through *feature functions* that count the number of occurrences of a certain pattern in the representation. Anything that is definable in terms of the representation can serve as a feature that feeds into such functions. The parameters of the model are weights that reflect the importance or the usefulness of the respective features.

The feature-function values are multiplied by their estimated weights to yield the score of the structure defined by means of those features. The structure that maximizes  $p(y)$  is the selected parse, where:

$$p(y) = \frac{1}{Z_\lambda} e^{\sum_i \lambda_i f_i(y)} \quad (3.34)$$

$$Z_\lambda = \sum_{y \in \mathcal{Y}} e^{\sum_i \lambda_i f_i(y)} \quad (3.35)$$

The main challenge in using such models is estimating their parameters from data. There is no practical way of summing over all trees to calculate the normalization constant  $Z_\lambda$ , so estimation procedures that maximize the joint likelihood of the data and structures (MLE) are generally intractable. Johnson et al. [54] suggests to use *maximum conditional-likelihood estimation (MCLE)* instead, where one maximizes the conditional likelihood of the structure given its yield, and the normalizing factor  $Z_\lambda$  is replaced with the following, more manageable,  $Z_\lambda(y)$ .

$$Z_\lambda(y) = \sum_{y' \in \mathcal{Y}_x} e^{\sum_i \lambda_i f_i(y')}$$

This estimation procedure is consistent for the conditional distribution  $p(y|x)$  but not the joint distribution  $p(y, x)$  we have considered so far. This means that

the parser is optimized to *discriminate* between different candidates for a single sentence. There are different ways to incorporate discriminative methods into the parsing model.

- **Discriminative Reranking** [23, 29]: A probabilistic generative model can be used to generate a list of n-best candidates and then re-rank candidates using feature weights. Those feature-weights are learned through a discrimination procedure. These features are selected based on pre-defined feature-schemata and an automatic procedure selects the ones that show the best gains.
- **Discriminative Estimation**: It is possible to limit the application of the discriminative method to the estimation of individual parameters of a generative model, and using a probabilistic model for joint inference [53, 89]. The conditional estimation allows us to incorporate arbitrary features, but since the parameters are employed in a simple generative process, the feature combination has to remain local. Conditional estimation procedures allow for potentially incorporating more information into individual parameters when training on a small amount of data.
- **Discriminative Parsing**: Finkel et al. [36] propose an end-to-end CFG parsing system based on Conditional Random Fields (CRF-CFG) in which the estimated probabilities are normalized globally for undirected representations of complete trees. They use an estimation procedure that maximizes the conditional likelihood instead of the joint likelihood, and enrich their parameters with non-local features. The features they use are selected from feature-schemata and the best features are detected using a small development set. Their parser has been applied only to English, and obtained parsing results on a par with [13, 51].

### 3.2.4 The Lexical Model

#### Combatting Lexical Sparsity

A parser has to assign morphosyntactic categories to the word-forms in the input. These morphosyntactic categories reflect the part-of-speech tag of the word and a set morphological features. Morphosyntactic categories provide the interface between the structure of words (morphology) and the structure of sentences (syntax). In whichever generative probabilistic framework we use for parsing (treebank PCFG, PCFG-LA, RR-PCFG, Factored-PCFG, etc.) we need to estimate the lexical probabilities  $p(w|t)$ , that is, to assign tag probability distributions to word forms.

The probability of lexical rules may be estimated using maximum likelihood estimates, just like the syntactic rules, by counting the relative frequency of word-tag co-occurrences.

$$\hat{p}_{tb}(w|t) = \frac{\#(w, t)}{\sum_w \#(w, t)} = \frac{\#(w, t)}{\#t}$$

Because of the high variation in the morphological form of words, it is hard to inspect all possible morphosyntactic assignments in advance. This is further complicated by the fact that the amount of resources available for parsing Semitic languages is not as extensive as it is for English. Therefore we should estimate the tagging distribution for Out-Of-Vocabulary (OOV) words, and possibly also smooth the probability distribution over rarely occurring ones. To do so, it is customary to fix a rare threshold value  $\beta$  and estimate the different probability distributions separately.

$$p(w|t) = \begin{cases} \hat{p}_{tb}(w|t) & \#w > \beta \\ \hat{p}_{rare}(w|t) & 0 < \#w \leq \beta \\ \hat{p}_{oov}(w|t) & \#w = 0 \end{cases}$$

The simplest way to estimate the distributions for rare and OOV words is by assuming a single distribution of tags for unknown words. That is, one simply assumes an abstract UNKNOWN word and estimates the probability of tagging an UNKNOWN word using the distribution of tags on rare words in the data. Every word type which is under the  $\beta$  threshold is considered a member of a set RARE. We can calculate the OOV distribution as the probability distribution of RARE words.

$$\hat{p}_{oov}(w|t) = \hat{p}_{rare}(w|t) = \frac{\sum_{w \in \text{RARE}} \#(w, t)}{\sum_w \#(w, t)}$$

This estimate embodies a very crude assumption, that the tag assignment to unknown words is not related to surface forms. Obviously, for morphologically rich languages this is not the case, because the form of the word provide cues concerning its syntactic category. Therefore, a refined way to estimate the rare and OOV probability distributions may be using external resources, such external dictionaries and lexica.

## Using an External Lexicon

An external lexicon provides the list of word forms in the language along with their morphological analyses. A morphological analyzer can be thought of as a function mapping word-forms to a set of morphosyntactic signatures. The information provided by an external lexicon can be used for improving the statistical estimates of lexical insertion probabilities for unseen or rare words, in (at least) the following ways.

### Signatures

The basic estimates for the lexical insertion probability given a tag  $p(w|t)$  are estimated according to observation in the training set. Additionally, we can use the

lexicon to analyze input words and to include parameter estimates for unknown word signatures  $p(s|t)$ .

A study of parsing Arabic and French by Green et al. [46], for example, estimates the probability distribution of  $p(t|w)$  and  $p(t|s)$  directly from the training data using relative frequency estimates, and then use Bayes law to get the estimates of  $p(w|t)$  (where  $p_{smooth}$  interpolates the  $p(t|w)$  and  $p(s|t)$  estimates).

$$\begin{aligned}\hat{p}_{lb}(w|t) &= \frac{p(t|w) \times p(w)}{p(t)} & \#w > \beta \\ \hat{p}_{rare}(w|t) &= \frac{p_{smooth}(t|w) \times p(w)}{p(t)} & 0 < \#w \leq \beta \\ \hat{p}_{oov}(w|t) &= \frac{p(t|s) \times p(s)}{p(t)} & \#w = 0\end{aligned}$$

### Factored Lexicon

Another way to combat sparsity is to parse a factored representation of the terminals. Each complex terminal symbol may be factored into a word form, a lemma, grammatical features such as gender, number, and person, a morphological template, and so on. In the factored lexicon, each token has an associated morphosyntactic representation  $m$ , including these various grammatical features. Green et al. [46] generate the word and morphosyntactic tag using a product of independent factors, each of which is conditioned on the part-of-speech tag.

$$p(w; m|t) = p(w|t)p(m|t)$$

The probability  $p(m|t)$  is estimated using the procedure of estimating  $p(w|t)$  specified above. Since the number of  $m, t$  tuples in the training data is a lot smaller than the number of  $w, t$  tuples, many of the probability distributions  $p(m|t)$  may be estimated directly from training data.

### Using Unannotated Data

Given a morphological analyzer for a language, we would like to use morphosyntactic analyses in order to obtain robust probabilistic estimates for rare and unknown words. There are two main challenges that stand in the way of using a morphological analyzer in this way. One challenge is that the morphological analyzer is symbolic, that means that it can provide the list of analyses, but it does not provide the probabilities  $p(w|t)$  for the different analyses. So we need to estimate them from annotated or unannotated data. The other challenge is more subtle. In many cases, the morphological analyzer and the syntactic treebank do not respect the same annotation scheme. In some cases, the morphological analyzer categories are more fine-grained, in other cases the treebank part-of-speech categories are more fine-grained. In any event there is no simple mapping between the two sets of categories. In all cases, the mapping itself may have to be learned directly from the data.

## Layering

The study of Goldberg et al. [43] presents a method for enhancing unlexicalized parsing performance using a wide-coverage lexicon, fuzzy tag-set mapping, and EM-HMM-based lexical probabilities which are obtained using unannotated data. In their setup, the treebank annotation scheme is different from the format of analysis provided by the morphological analyzer. They show that simply retagging the corpus with the latter scheme gives inferior results, since syntactically relevant distinctions are missed. Using treebank tags only provide a good performance on seen words, but precludes the use of the morphological analyzer for analyzing OOV words.

To overcome this, they propose a generative approach in which the emission probability is decomposed. First, the tree-bank tag generates a lexicon category, and then the lexicon category generates the word.

$$p(w|t_{tb}) = p(t_{lex}|t_{tb})p(w|t_{lex})$$

The transfer probabilities  $p(t_{lex}|t_{tb})$  are estimated using maximum likelihood estimates on a layered representation of the treebank, in which every production is extended with the lexical category assigned in the lexicon. The lexicon categories assignment was performed semi-automatically by human experts. The lexical probabilities  $p(w|t_{lex})$  are estimated using an HMM tagger in a semi-supervised manner, using a large set of unannotated data [2]. Goldberg et al. [43] show that using this generative process and estimation of lexical categories improves the performance of a manually-refined treebank PCFG.

## Marginalizing

Manually tagging the corpus with two layers of part-of-speech categories may be labor intensive, and in any event, human annotators themselves do not always reach an agreement on the correct mapping. Goldberg [38] presents an alternative method for assigning lexical probabilities, by marginalizing over all possible lexical tag assignments. Goldberg first assumes an inversion using Bayes law.

$$p(w|t_{tb}) = \frac{p(t_{tb}|w)p(w)}{p(t_{tb})}$$

Then, he provides different ways of estimating  $p(t_{tb}|w)$ .

$$\begin{aligned} \hat{p}_{tb}(t_{tb}|w) &= \frac{\#w,t_{tb}}{\#t_{tb}} & \#w > \beta \\ \hat{p}_{oov}(t_{tb}|w) &= \sum_{t_{lex}} p(t_{lex}|w)p(t_{tb}|t_{lex}) & \#w = 0 \\ \hat{p}_{rare}(t_{tb}|w) &= \frac{\#w \times p_{tb}(t_{tb}|w) + p_{oov}(t_{tb}|w)}{\#w+1} & 0 < \#w \leq \beta \end{aligned}$$

For frequent words, Goldberg [38] uses counts on the treebank. For unseen words, he estimates  $p(t_{lex}|w)$  using pseudocounts taken from an HMM tagger applied to unannotated data [1]. Because the HMM tagger uses an external lexicon, the lexicon tags are different from the treebank tags. The transfer probabilities  $p(t_{tb}|t_{lex})$  are treated as hidden variables and are marginalized out during decoding.

## Clustering

An additional way to utilize unannotated data is by means of unsupervised clustering. First, we use a large unannotated corpus to learn word clusters (e.g., Brown clusters [15]). Then, these clusters can be used as features in a statistical parser. Koo et al. [61] used a cluster-based feature mapping for a discriminative learner. They replaced words with full description of the cluster and pos-tags with shorter description of the hierarchical cluster. A possible way to incorporate these features into a constituency-based generative parser is, for instance, to change the pos-rags and word-form representation into clusters. Candito and others [19, 20] tested the method of replacing inflected forms by clusters of forms in a generative probabilistic parser, and obtained good results. Levinger et al. [64] take a different approach to clustering, where word clusters are based on linguistic similarity. They collect similar-word (SW) sets using hand-coded rules and use the frequencies of words in SW sets in order to differentiate the different analyses of an ambiguous word. They show that their statistical estimates improves performance on a number of tasks. Their method was also useful for initializing semi-supervised methods for tagging [1].

## 3.3 Empirical Results

### 3.3.1 Parsing Modern Standard Arabic

The Penn Arabic Treebank (ATB), a set of annotated corpora for Modern Standard Arabic (MSA), has been developed by the Linguistic Data Consortium (LDC) since 2001.<sup>19</sup> The ATB annotation encodes the rich morphological structure of words and indicates grammatical relations in the form of phrase-structure trees [65]. In addition, the ATB deals with phonemic transliteration of the Arabic script, bidirectionality, and the incorporation of diacritics [66, 69]. The PoS tags in the ATB are complex tags that specify the part-of-speech category, an English gloss, and a list of inflectional features for every space-delimited word. Independent clitics are

---

<sup>19</sup>The project page is available at <http://www.lrcs.upenn.edu/arabic/>.

segmented away and are assigned their own PoS tag. Syntactic trees are built on top of morphologically analyzed and segmented input.

The LDC team made a deliberate effort to rely on the English PTB annotation guidelines in order to speed up the development of the Arabic annotated corpora. The morphological analyzer of Buckwalter [17] was used to propose all possible analyses for surface space-delimited tokens, from which human annotators chose the correct one. Then, the parsing engine of Bikel [7] was used to automatically bootstrap syntactic parse-trees for the morphologically analyzed tokens [65]. This annotation pipeline has been challenged in [67, 68] due to mismatches between the PoS tags assigned by the morphological analyzer and the typical syntactic structures assigned by the parser. These mismatches are found to originate in the inherent complexity of the Semitic grammar, and the ATB guidelines have been revised to reflect such mismatches and treat them in the annotation scheme.

The LDC published different versions of the ATB. ATBp1 contains 166K words from the Agence France Press corpus, segmented, PoS tagged and syntactically analyzed. ATBp2 contains 144K words from *Al-Hayat*, annotated similarly but with added case and mood endings indicated by diacritics. ATBp3 contains 350K words from the newswire text *An-Nahar*, in which the text is morphologically segmented, vocalized, tagged and syntactically analyzed. Because of the discrepancy between treebank words and space-delimited tokens, any morphologically analyzed and vocalized file contains the mappings to the original surface forms.

The first constituency-based parser which was trained to parse Arabic using the ATB is the Head-Driven lexicalized parsing engine of Bikel [7], adapted in [9] to Arabic parsing. The rich morphological structure of Arabic words induces a large set of complex tags which the parser was not equipped to handle (Bikel [9, pp. 79–80]), so the adaptation involved a reduction of the Arabic rich tag set to a set with comparable size to that of English. Bikel [9] only experimented with parsing gold input, assuming that morphological segmentation and tagging are known in advance. The results of applying Bikel’s parsing Engine to parsing Arabic (75.68 F-Score) were lower than the results obtained previously for parsing English on a data set of a comparable size (87.54 F-Score) [63, 66].

Parsing using Bikel’s engine was mostly targeted at finding enhancements of the initial category set in the ATB. In a series of studies the ATB development team suggested fixes for punctuation, deverbal material, finer-grained distinctions between different types of nouns and demonstratives, and other splits based on these experiments [63, 66–69]. Once the PoS tags have been enhanced, the ATB team changed the trees dominating certain nominals to reflect their verbal readings. The parsing results for all of these models leveled at about 79 F-Score for parsing gold segmented and tagged input [63].

Green and Manning [45] study in greater detail the factors affecting Arabic parsing performance using the ATB. They firstly compare the ATB to the Chinese and English treebanks, and report that they are comparable in gross statistical terms, while the ATB demonstrates a higher level of annotation inconsistencies. They then compare the performance of three different parsing models on the standard split of the ATB [25]. Specifically, they compare Bikel’s parsing engine [8] and Petrov’s

PCFG-LA parser [83] with a human-interpretable state-split PCFG developed to address particular sources of ambiguity that they find to be pertinent in the ATB. They show that this PCFG, in conjunction with the Stanford parser (a factored model implementation based on [57]), achieves  $F_1$ 80.67 on gold segmented and tagged input, outperforming the Bikel parser ( $F_1$ 77.99) and outperformed by the PCFG-LA parser ( $F_1$ 84.29) in the same settings. This performance trend persists when the corpus is pre-tagged or self-tagged by the parser (albeit obtaining lower scores). The trend was further confirmed using the leaf-ancestor metrics [92]. They finally compare the performance of the Stanford parser on the ATB for raw words, using a pipeline and a joint scenario. They show that a pipeline containing the MADA tagger followed by the Stanford parser ( $F_1$ 79.17) outperforms joint lattice-based parsing using the stanford parser ( $F_1$ 76.01), an overall loss of 2pt-5pt in F-Scores relative to the artificial gold-input parsing scenario.

Green and Manning [45] provide strong baselines for Arabic parsing and demonstrated the empirical advantages of the PCFG-LA parser [83]. Attia et al. [4] explore the use of simple or complex morphological cues for the handling of rare and unknown words in English, Arabic and French, in order to further improve the performance of the PCFG-LA model. They use their own PCFG-LA implementation and contrast two techniques of handling unknown words: a language-independent method and a language-specific one. The language-independent technique simply learns a distribution for unknown words using rare words. In the language-specific case they map every word to its morphosyntactic signature, and divide the UNKNOWN probability mass across morphosyntactic signatures. Their experiments show that language specific cues were more helpful for Arabic than they were for French and English, ultimately increasing parser performance on the ATB dev set from  $F_1$ 79.53 to  $F_1$ 81.42, for gold segmented input. A study by Dehadri et al. [32] explored strategies for finding the best feature set and PoS tag splits using the same parser. They experimented with a non-iterative best-first search algorithm, an iterative, greedy best-first search algorithm, an iterative, best-first with backtracking search algorithm and simulated annealing. They obtain  $F_1$ 85.03 for vocalized ATB no-tag parsing and  $F_1$ 83.34 for unvocalized ATB no-tag parsing, both of which assuming gold segmented input.

Arabic constituency parsing has also been explored in the context of parsing Arabic Dialects [25]. Arabic dialects are mainly spoken, so they do not have annotated corpora of their own. Chiang et al. [25] try to leverage the lack of data by exploiting the ATB and using explicit knowledge about the relation between MSA and its Levantine Arabic (LA) dialect. They test three approaches of interjecting such knowledge to the parsing architecture: sentence transduction, in which the LA sentence is turned into an MSA sentence and then parsed with an MSA parser, treebank transduction, in which the MSA treebank is turned into an LA treebank and used as training material for an LA parser; and grammar transduction, in which an MSA grammar is turned into an LA grammar which is then used for parsing LA. Grammar transduction obtained the best result, around  $F_1$ 67 on gold-tagged input, more than 17 % error reduction over their baseline.

Advances in dependency parsing [62] and claims for the adequacy of dependency grammars for analyzing free word-order phenomena [78, 98] have jointly led to increased interest in Arabic dependency parsing. Initial Arabic dependency parsing results were published in connection to data sets that were made available in the shared tasks on multilingual dependency parsing [16, 81].<sup>20</sup> The experimental setup in these experiments is artificial in the sense that it assumes that the gold segmentation, part-of-speech tags and morphological features are known prior to parsing. This is an unrealistic assumption, but even in these settings the task allowed the community to gain insights concerning the adequacy of general-purpose parsers for parsing Semitic phenomena. In an overall analysis of the shared-task results, Nivre et al. [81] makes the typological observation that Arabic is one of the most challenging languages to parse, regardless of the parsing method used. The hope that a completely language-independent parser will work sufficiently well on Semitic data has not been met, and various specific studies on Arabic dependency parsing followed.

Marton et al. [72–74] focus on transition-based parsing and present a thorough study of the contribution of the lexical and inflectional features to Arabic dependency parsing. They experiment with two parsers, MaltParser [80] and EasyFirst [37], in three different experimental settings: gold scenarios (where morphological analysis is known in advance), and predicted scenarios (in which morphological analyses are automatically predicted). The data set they used is a converted version of the ATB into dependencies, using a narrowed down set of dependency labels.<sup>21</sup> They report that more informative (fine-grained) tag sets are useful in gold scenarios, but may be detrimental in predicted scenarios. They identify a set of features (definiteness, person, number, gender, undiacritized lemma) that improve parsing performance, even in predicted scenarios. They finally show that some of the deficiency of parsing in predicted scenarios may be mitigated by training the parsers on both gold and predicted tags. Their findings are robust across parsers, with an advantage for EasyFirst (82.6 vs. 81.9 labeled attachment scores in the best settings).<sup>22</sup>

### 3.3.2 Parsing Modern Hebrew

The main resource for the development of statistical parsers for Hebrew is the Modern Hebrew Treebank [96], which was developed at MILA – The knowledge center for processing Hebrew.<sup>23</sup> The treebank contains 6,220 unvocalized sentences

---

<sup>20</sup>The CoNLL-X shared task page for is available at <http://ilk.uvt.nl/conll/>.

<sup>21</sup>The project homepage is available at <http://www.ccls.columbia.edu/project/catib>.

<sup>22</sup>An additional resource for Arabic dependency parsing is the Prague dependency treebank [http://ufal.mff.cuni.cz/padt/PADT\\_1.0/docs/index.html](http://ufal.mff.cuni.cz/padt/PADT_1.0/docs/index.html) which we omit from the present discussion.

<sup>23</sup>The MILA center homepage is at <http://www.mila.cs.technion.ac.il/>.

from the daily newspaper *Ha’aretz*. The sentences were morphologically segmented and syntactically annotated in a semi-automatic process. Each node in the syntactic parse tree contains different sorts of information: a phrase-structure label (S, NP, VP, etc.) grammatical function labels (from a small set containing SBJ, OBJ, COM and CNJ) and a set of morphological features decorating part-of-speech tags. In the second version of the Hebrew treebank, morphological features have been percolated to higher level syntactic categories, and (partial) head information has added to phrase-level nodes, based on hand-coded syntactic rules [48].

Tsarfaty [100] used the Hebrew treebank for creating the first Hebrew NLP pipeline, using a combination of a morphological analyzer, a part-of-speech tagger, a treebank-induced PCFG and a general-purpose chart parser. The best reported result on parsing raw input in this study was  $F_1$ 61 with 91.5 segmentation accuracy. Since the results involved unmatched yields, the scores were reported using character-based ParseEval. Cohen and Smith [28] followed up with an implementation of a joint inference for morphological and syntactic disambiguation using a product of experts. Their syntactic model is a treebank PCFG with first-order vertical Markovization. For the morphological model they contrast a unigram-based model and a discriminative model based on Conditional Random Fields (CRF). For their best model they report 64.4 parsing accuracy and 90.9 segmentation accuracy for the joint model on raw input words. (The numbers are not comparable to other studies, however, because Cohen and Smith used their own evaluation metrics based on string edit distance correction to the tree.)

Goldberg and Tsarfaty [42] present a fully generative solution to the joint morphological and syntactic disambiguation task based on lattice parsing. Their best model included a PCFG with 1st-order vertical Markovization, manually selected category splits and unknown words handling using a Hebrew spell-checker. The baseline for Hebrew parsing was now pushed further to  $F_1$ 66.60 on parsing raw input (68.79 on the metric of Cohen and Smith) with  $F_1$ 94.7 segmentation accuracy. These results made it clear that for a Hebrew treebank so small, additional resources for unknown words treatment may have crucial effects on parsing accuracy. Goldberg et al. [43] experimented with a wide-coverage lexicon, fuzzy tag-set mapping and EM-HMM-based lexical probabilities acquired using a large unannotated corpus, and reported 73.40/73.99 precision/recall on parsing raw words (compared with 76.81/76.49 precision/recall for parsing off of gold segmentation). The components of these different solutions, including a lattice-based decoder, a wide-coverage lexicon and HMM probabilities were incorporated by Goldberg [38] into the PCFG-LA parser of Petrov [83], along with a rule-based filter to the output, obtaining  $F_1$ 76.95 on parsing raw words (and  $F_1$  85.7 for parsing gold segments).

In an orthogonal line of work, Tsarfaty and Sima’an [102] explored the contribution of different parameterization decisions to probabilistic grammars induced from the Hebrew treebank. They experimented with horizontal Markovization, vertical Markovization and morphological state-splits applied to a PCFG-based parsing model. They have shown that when each dimension of parameterization is considered in isolation, the *depth* annotation contributed more to parsing accuracy than

parent annotation, which in turn contributed more than horizontal Markovization. They further show that the contribution of the different dimensions is cumulative, and that using 1st-order vertical Markovization, 1st-order horizontal Markovization and one level of morphological depth, parsing results improve from  $F_1$ 69.24 to  $F_1$ 74.41 on gold segmented input.

The emerging insight is that parameterization decisions which work well for English parsing may miss important generalizations that are relevant for parsing other languages, e.g., languages with rich morphology. Tsarfaty et al. [103–105] followed up on this insight with the development of the Relational-Realizational (RR) model and its application to parsing Hebrew. In the application to Hebrew data, described in detail in Tsarfaty [101], the RR framework is applied to explicitly modeling and directly estimating morphosyntactic patterns such as differential case marking, agreement and clitics. The model obtained  $F_1$ 76.6 score on parsing gold segmented input, and  $F_1$ 84.4 when parsing gold segmented and tagged input.

A recent study by Tsarfaty et al. [107] compares a combination of the RR model with lattice-based PCFG-LA parsing [39] with lattice-based PCFG-LA parsing trained on phrase structure (PS) trees. The different models are evaluated using TedEval, distance-based scores which are tailored for evaluating morphological syntactic disambiguation jointly [107]. In all parsing scenarios (gold, predicted tags, raw), the bare-bone PS model of Goldberg and Elhadad [39] outperform the PS trees extracted from the RR model (TedEval yields 93.39, 86.26 80.67 on trees given by the PS model and 92.45, 85.83, 79.46 on trees extracted from the RR model). An advantage of the RR trees is, however, that they provide a direct representation of predicate-argument structure in the function labels. Evaluation on dependency labels using TedEval shows 88.01, 82.64 and 76.10 accuracy on RR trees, slightly better than obtained by general-purpose dependency parsers on the same task.

A different way to obtain a direct representation of the predicate argument-structure of sentences is by training a statistical parser directly on dependency trees. Goldberg and Elhadad [40] converted the Hebrew treebank into a dependency treebank and compared the results of several general-purpose dependency parsers on the treebank in gold vs. predicted settings.<sup>24</sup> They tested the graph-based MST parser [77] and the transition-based MaltParser [82]. In all scenarios, the 2nd-order MST model outperformed the 1st-order MST, and both outperformed MaltParser. The best result obtained for the Hebrew treebank, a second order MST with limited lexical features, was 84.77 unlabeled attachment scores on the gold input scenario and 76.10 on unlabeled attachment scores on automatically predicted segmentation and tagging.

Goldberg [37] moved on to propose EasyFirst, a non-directional transition-based dependency parser that is aimed at making easier attachments earlier. The feature model of the EasyFirst parser can make reference to partial parses, and thus allows for incorporating features that encode rich morphological interactions. The ability of the EasyFirst parser to prefer different ordering of attachments, along with

---

<sup>24</sup>The treebank data is available at <http://www.cs.bgu.ac.il/~yoavg/data/hebdepth/>.

the ability to incorporate complex features, makes it suited for modeling Semitic phenomena. Goldberg and Elhadad [41] report 84.2 unlabeled attachment scores (UAS) on the gold input and 76.2 UAS on automatically predicted segmentation and tagging. It is still an open empirical question how well the EasyFirst performs on labeled dependencies.

### 3.4 Conclusion and Future Work

This chapter surveyed morphosyntactic phenomena in Semitic languages and presented the challenges they pose to general-purpose statistical parsers. We have seen that the structure of Semitic languages complicates the NLP pipeline, by providing morphologically ambiguous signal as input to the parser. We have further shown how parameterization decisions in generative parsing models may affect the performance of a parser and proposed various methods for adding linguistic constraints and for dealing with sparse data.

We reported parsing results on Modern Hebrew and Modern Standard Arabic. A pre-condition for applying any statistical parsing model to the other Semitic languages is the availability of morphologically and syntactically annotated corpora. As of yet there are no broad-coverage treebanks for these languages. An important direction for further research is therefore the development of such annotated corpora for languages such as Syriac, Amharic or Maltese. Given the similarity of Semitic languages to one another, it may be possible to speed up the development of parsers for these data using transfer learning.

For languages with annotated corpora, the time is ripe for exploring more sophisticated modeling methods, possibly incorporating discriminative estimation and/or semantic resources. It would also be interesting to combine the strengths of the different strategies presented here. For instance, while the constituency parser of Goldberg [38] shows gains from improving the lexical model, the method of Tsarfaty [101] is targeted at improving the syntactic model. Their combination may yield a more accurate parser. Dependency parsing will benefit from investigating joint morphosyntactic disambiguation scenarios, in order to avoid errors propagation through the pipeline.

Finally, a careful treatment of evaluation across languages, parsers and scenarios is critical. We have seen in the previous section that while many of the studies deal with the same challenges, they report numbers on different metrics and on different scales. Upon developing better methods to evaluate the quality of parses across frameworks, possibly within downstream applications, we will be able to point out and further develop models that make better predictions for Semitic data.

**Acknowledgements** The author wishes to thank Joakim Nivre, Imed Zitouni, Ro'i Gibori and two anonymous reviewers for useful comments on earlier drafts. The author further thanks Yoav Goldberg, Spence Green and Shay Cohen for discussion and relevant resources, and Reshef Shilon for discussion of Arabic data. The writing of this chapter was partially funded by the Swedish research council, for which we are grateful.

## References

1. Adler M (2007) Hebrew morphological disambiguation: an unsupervised stochastic word-based approach. PhD thesis, Ben-Gurion University
2. Adler M, Elhadad M (2006) An unsupervised morpheme-based HMM for Hebrew morphological disambiguation. In: Proceedings of COLING-ACL, Sydney
3. Anderson SR (1992) A-morphous morphology. Cambridge University Press, Cambridge/New York
4. Attia M, Foster J, Hogan D, Roux JL, Tounsi L, van Genabith J (2010) Handling unknown words in statistical latent-variable parsing models for Arabic, English and French. In: Workshop on parsing morphologically rich languages, Los Angeles
5. Baker M (2011) On the relationship of object agreement and accusative case: evidence from Amharic. *Linguist Inq* 43(2):255–274.
6. Berger AL, Pietra SAD, Pietra VAD (1996) A maximum-entropy approach to natural language processing. *Comput Linguist* 22(1):39–71
7. Bikel DM (2002) Design of multi-lingual parallel processing statistical parsing engine. In: Proceedings of HLT, San Diego
8. Bikel DM (2004) Intricacies of Collins' parsing model. *Comput Linguist* 4(30):479–511
9. Bikel DM (2004) On the parameter space of generative lexicalized statistical parsing models. PhD thesis, University of Pennsylvania
10. Black E, Lafferty J, Roukos S (1992) Development and evaluation of a broad-coverage probabilistic grammar of English language computer manuals. In: Proceedings of ACL, Newark
11. Bloomfield L (1933) Language. Holt, Rinehart and Winston Inc., New York
12. Bod R (1995) Enriching linguistics with statistics. PhD thesis, University of Amsterdam
13. Bod R (2003) An efficient implementation of a new DOP model. In: Proceedings of EACL, Budapest
14. Bresnan J (2000) Lexical-functional syntax. Blackwell, Oxford
15. Brown PF, deSouza PV, Mercer RL, Pietra VJD, Lai JC (1992) Class-based n-gram models of natural language. *Comput Linguist* 18(4):467–479. <http://dl.acm.org/citation.cfm?id=176313.176316>
16. Buchholz S, Marsi E (2006) CoNLL-X shared task on multilingual dependency parsing. In: Proceedings of CoNLL-X, New York, pp 149–164
17. Buckwalter T (2002) Arabic morphological analyzer version 1.0. Linguistic Data Consortium, Philadelphia.
18. Cahill A, Burke M, O'Donovan R, Riezler S, van Genabith J, Way A (2008) Wide-coverage deep statistical parsing using automatic dependency structure annotation. *Comput Linguist* 34(1):81–124
19. Candito M, Crabbé B (2009) Improving generative statistical parsing with semi-supervised word clustering. In: Proceedings of IWPT, Paris
20. Candito M, Seddah D (2010) Parsing word clusters. In: Proceedings of NAACL/HLT workshop on statistical parsing of morphologically rich languages, Los Angeles
21. Charniak E (1996) Tree-bank grammars. In: AAAI/IAAI, Portland, vol 2, pp 1031–1036
22. Charniak E (1997) Statistical parsing with a context-free grammar and word statistics. In: Proceedings of the fourteenth national conference on artificial intelligence, Providence. AAAI/MIT, Menlo Park, pp 598–603
23. Charniak E, Johnson M (2005) Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In: Proceedings of ACL, Ann Arbor
24. Charniak E, Johnson M, Eisner M, Joseph Austerweil DE, Haxton I, Hill C, Shrivaths R, Moore J, Pozar M, Vu T (2006) Multilevel coarse-to-fine PCFG parsing. In: Proceedings of HLT-NAACL, New York
25. Chiang D, Diab M, Habash N, Rambow O, Shareef S (2006) Parsing Arabic dialects. In: Proceedings of EACL, Trento

26. Chomsky N (1956) Three models for the description of language. *IRE Trans Inf Theory* 2(2):113–123
27. Chomsky N (1957) Syntactic structures. Mouton, The Hague
28. Cohen SB, Smith NA (2007) Joint morphological and syntactic disambiguation. In: Proceedings of EMNLP-CoNLL, Prague, pp 208–217
29. Collins M (2000) Discriminative reranking for natural language parsing. In: Proceedings of ICML, Stanford
30. Collins M (2003) Head-driven statistical models for natural language parsing. *Comput Linguist* 29(4):589–637
31. Danon G (2001) Syntactic definiteness in the grammar of Modern Hebrew. *Linguistics* 6(39):1071–1116
32. Dehdari J, Tounsi L, van Genabith J (2011) Morphological features for parsing morphologically-rich languages: a case of Arabic. In: Proceedings of the second workshop on parsing morphologically rich languages, Dublin
33. Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. *J R Stat Soc Ser B* 39(1):1–38. JSTOR 2984875. MR 0501537
34. Doron E (2003) Agency and voice: the semantics of the Semitic templates. *Nat Lang Semant* 11:1–67
35. Emms M (2008) Tree-distance and some other variants of evalb. In: Proceedings of LREC, Marrakech
36. Finkel JR, Kleeman A, Manning CD (2008) Efficient, feature-based, conditional random field parsing. In: Proceedings of ACL, Columbus
37. Goldberg Y (2010) An efficient algorithm for easy-first non-directional dependency parsing. In: Proceedings of NAACL/HLT, Los Angeles
38. Goldberg Y (2011) Automatic syntactic analysis of Modern Hebrew. PhD thesis, Ben-Gurion University
39. Goldberg Y and Elhadad M (2011) Joint Hebrew Segmentation and Parsing using a PCFG-LA Lattice Parser. In: Proceedings of ACL, Portland
40. Goldberg Y, Elhadad M (2009) Hebrew dependency parsing: initial results. In: Proceedings of IWPT, Paris
41. Goldberg Y, Elhadad M (2010) Easy-first dependency parsing of Modern Hebrew. In: Proceedings of NAACL/HLT workshop on statistical parsing of morphologically rich languages, Los Angeles
42. Goldberg Y, Tsarfaty R (2008) A single framework for joint morphological segmentation and syntactic parsing. In: Proceedings of ACL, Columbus
43. Goldberg Y, Tsarfaty R, Adler M, Elhadad M (2009) Enhancing unlexicalized parsing performance using a wide coverage lexicon, fuzzy tag-set mapping, and EM-HMM-based lexical probabilities. In: Proceedings of EACL, Athens
44. Goodman J (1997) Probabilistic feature grammars. In: Proceedings of IWPT, Boston
45. Green S, Manning CD (2010) Better Arabic parsing: baselines, evaluations, and analysis. In: Proceedings of COLING, Beijing. <http://dl.acm.org/citation.cfm?id=2464109>
46. Green S, de Marneffe MC, Manning CD (2013) Parsing models for identifying multiword expressions. *J Comput Linguist* 39(1):195–227. Special issue on parsing morphologically rich languages
47. Greenberg JH (1963) Some universals of grammar with particular reference to the order of meaningful elements. In: Greenberg JH (ed) Universals of language. MIT, Cambridge, pp 73–113
48. Guthmann N, Krymolowski Y, Milea A, Winter Y (2009) Automatic annotation of morpho-syntactic dependencies in a Modern Hebrew treebank. In: Eynde FV, Frank A, Smedt KD, van Noord G (eds) Proceedings of TLT, Groningen
49. Habash N, Soudi A, Buckwalter T (2007) On Arabic transliteration. *Text Speech Lang Technol* 38:15–22
50. Hockenmaier J, Steedman M (2002) Generative models for statistical parsing with combinatorial categorial grammar. In: Proceedings of ACL, Philadelphia

51. Huang L (2008) Forest reranking: discriminative parsing with non-local features. In: Proceedings of ACL, Columbus
52. Johnson M (1998) PCFG models of linguistic tree representations. *Comput Linguist* 24(4):613–632
53. Johnson M (2001) Joint and conditional estimation of tagging and parsing models. In: Proceedings of ACL, Toulouse
54. Johnson M, Geman S, Canon S, Chi Z, Riezler S (1999) Estimators for stochastic “unification-based” grammars. In: Proceedings of ACL, College Park
55. Jurafsky D, Martin JH (2009) Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd edn. Pearson Education, Upper Saddle River. <http://www.pearsonhighered.com/educator/product/Speech-and-Language-Processing/9780131873216.page>
56. Kasami T (1965) An efficient recognition and syntax-analysis algorithm for context-free languages. Technical report, Scientific report AFCRL-65-758, Air Force Cambridge Research Lab
57. Klein D, Manning CD (2002) Fast exact inference with a factored model for natural language parsing. In: Advances in neural information processing systems 15 (NIPS 2002). MIT, Cambridge.
58. Klein D, Manning CD (2003) A\* parsing: fast exact viterbi parse selection. In: Proceedings of NAACL, Edmonton
59. Klein D, Manning CD (2003) Accurate unlexicalized parsing. In: Proceedings of ACL, Sapporo, pp 423–430
60. Klein D, Manning CD (2003) Factored A\* search for models over sequences and trees. In: Proceedings of IJCAI, Acapulco
61. Koo T, Carreras X, Collins M (2008) Simple semi-supervised dependency parsing. In: Proceedings of ACL/HLT, Columbus
62. Kübler S, McDonald R, Nivre J (2009) Dependency parsing. No. 2 in synthesis lectures on human language technologies. Morgan & Claypool Publishers, San Rafael
63. Kulick S, Gabbard R, Marcus M (2006) Parsing the Arabic Treebank: analysis and improvements. In: Proceedings of TLT, Prague
64. Levinger M, Orman U, Itai A (1995) Learning morpho-lexical probabilistic from an untagged corpus with an application to Hebrew. *Comput Linguist* 21(3):383–404
65. Maamouri M, Bies A, Buckwalter T, Mekki W (2004) The Penn Arabic Treebank: building a large-scale annotated Arabic corpus. In: Proceedings of NEMLAR international conference on Arabic language resources and tools, Cairo
66. Maamouri M, Bies A, Kulick S (2008) Diacritic annotation in the Arabic TreeBank and its impact on parser evaluation. In: Bies A, Maamouri M, Kulick S (eds) Proceedings of the British Computer Society Arabic NLP/MT conference, London
67. Maamouri M, Bies A, Kulick S (2008) Enhanced annotation and parsing of the Arabic Treebank. In: Proceedings of INFOS, Cairo
68. Maamouri M, Bies A, Kulick S (2008) Enhancing the Arabic treebank: a collaborative effort toward new annotation guidelines. In: Proceedings of LREC, Marrakech
69. Maamouri M, Kulick S, Bies A (2008) Diacritic annotation in the Arabic Treebank and its impact on parser evaluation. In: Proceedings of LREC, Marrakech
70. Magerman DM (1995) Statistical decision-tree models for parsing. In: Proceedings of ACL, Cambridge, pp 276–283
71. Manning CD, Schütze H (1999) Foundations of statistical natural language processing. MIT, Cambridge
72. Marton Y, Habash N, Rambow O (2010) Improving Arabic dependency parsing with inflectional and lexical morphological features. In: Proceedings of the first workshop on statistical parsing of morphologically rich languages (SPMRL) at NA-ACL, Los Angeles
73. Marton Y, Habash N, Rambow O (2011) Improving Arabic dependency parsing with lexical and inflectional surface and functional features. In: Proceedings of ACL, Portland

74. Marton Y, Habash N, Rambow O (2013) Improving Arabic dependency parsing with surface and functional morphology features. *Comput Linguist* 39(1):161–194. Special issue on parsing morphologically rich languages.
75. Matuszaki T, Miyao Y, Tsuji J (2005) Probabilistic CFG with latent annotations. In: *Proceedings of ACL*, Ann Arbor
76. McClosky D, Charniak E, Johnson M (2006) Reranking and self-training for parser adaptation. In: *Proceedings of ACL*, Sydney
77. McDonald R, Pereira F, Ribarov K, Hajic J (2005) Non-projective dependency parsing using spanning tree algorithms. In: *HLT '05: proceedings of the conference on human language technology and empirical methods in natural language processing*, Vancouver. Association for Computational Linguistics, Morristown, pp 523–530
78. Mel'čuk I (1988) Dependency syntax: theory and practice. State University of New York Press, Albany
79. Miyao Y, Takashi N, Jun'ichi T (2004) Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the Penn Treebank. In: *Proceedings of IJCNLP*, Hainan Island
80. Nivre J, Hall J, Nilsson J (2006) MaltParser: a data-driven parser-generator for dependency parsing. In: *Proceedings of LREC*, Genoa, pp 2216–2219
81. Nivre J, Hall J, Kübler S, McDonald R, Nilsson J, Riedel S, Yuret D (2007) The CoNLL 2007 shared task on dependency parsing. In: *Proceedings of the CoNLL shared task session of EMNLP-CoNLL 2007*, Prague, pp 915–932
82. Nivre J, Nilsson J, Hall J, Chaney A, Eryigit G, Kübler S, Marinov S, Marsi E (2007) MaltParser: a language-independent system for data-driven dependency parsing. *Nat Lang Eng* 13(1):1–41
83. Petrov S (2009) Coarse-to-fine natural language processing. PhD thesis, University of California at Berkeley
84. Petrov S, Klein D (2007) Improved inference for unlexicalized parsing. In: *Human language technologies 2007: the conference of the North American chapter of the Association for Computational Linguistics; proceedings of the main conference*, Rochester. Association for Computational Linguistics, pp 404–411. <http://www.aclweb.org/anthology/N/N07/N07-1051>
85. Petrov S, Barrett L, Thibaux R, Klein D (2006) Learning accurate, compact, and interpretable tree annotation. In: *Proceedings of ACL*, Sydney
86. Prescher D (2005) Head-driven PCFGs with latent-head statistics. In: *Proceedings of ACL*, Ann Arbor
87. Prince A, Smolensky P (1993) Optimality theory: constraint interaction in generative grammar. Technical report, Rutgers University Center for Cognitive Science and Computer Science Department, University of Colorado at Boulder
88. Pullum GK, Scholz BC (2001) On the distinction between model-theoretic and generative-enumerative syntactic frameworks. In: *Proceedings of logical aspects of computational linguistics (LACL)*, Le Croisic
89. Ratnaparkhi A (1997) A linear observed time statistical parser based on maximum entropy models. In: *Proceedings of EMNLP*, Providence
90. Roark B, Harper M, Charniak E, Dorr B, Johnson M, Kahn JG, Liu Y, Ostendorf M, Hale J, Krasnyanskaya A, Lease M, Shafran I, Snover M, Stewart R, Yung L (2006) SParseval: evaluation metrics for parsing speech. In: *Proceedings of LREC*, Genoa
91. Sag IA, Wasow T (1999) Syntactic theory: a formal introduction. CSLI Publications, Stanford. <http://csli-publications.stanford.edu/site/1575861607.html>
92. Sampson G, Babarczy A (2002) A test of the leaf-ancestor metric for parse accuracy. In: *Proceedings of “Beyond Parseval” LREC workshop*, Las Palmas
93. Shieber SM (1985) Evidence against the context-freeness of natural language. *Linguist Philos* 8:333–343
94. Shilon R, Habash N, Lavie A, Wintner S (2012) Machine translation between Hebrew and Arabic. *Mach Transl* 26(1–2):177–195

95. Shlonsky U (1997) Clause structure and word order in Hebrew and Arabic. Oxford studies in comparative syntax. Oxford University Press, New York/Oxford
96. Sima'an K, Itai A, Winter Y, Altman A, Nativ N (2001) Building a tree-bank for Modern Hebrew text. *Traitemet Automatique des Langues* 42(2):347–380
97. Steedman M (1996) Surface structures and interpretation. No. 30 in linguistic inquiry monograph. MIT, Cambridge
98. Tesnière L (1959) *Élèmenets de Syntaxe Structurale*. Editions Klincksieck, Paris
99. Tsarfaty R (2005) Participants in action: aspectual meanings and thematic relations interplay in the semantics of semitic morphology. In: Zeevat H, ten Cate B (eds) *Proceedings of the sixth international Tbilisi symposium on language, logic and computation*, Batumi
100. Tsarfaty R (2006) Integrated morphological and syntactic disambiguation for Modern Hebrew. In: *Proceeding of ACL-SRW*, Sydney
101. Tsarfaty R (2010) Relational-realizational parsing. PhD thesis, University of Amsterdam
102. Tsarfaty R, Sima'an K (2007) Three-dimensional parametrization for parsing morphologically rich languages. In: *Proceedings of IWPT*, Prague
103. Tsarfaty R, Sima'an K (2008) Relational-realizational parsing. In: *Proceedings of CoLing*, Manchester
104. Tsarfaty R, Sima'an K (2010) Modeling morphosyntactic agreement for constituency-based parsing of Modern Hebrew. In: *Proceedings of NAACL/HLT workshop on statistical parsing of morphologically rich languages*, Los Angeles
105. Tsarfaty R, Sima'an K, Scha R (2009) An alternative to head-driven approaches for parsing a (relatively) free word order language. In: *Proceedings of EMNLP*, Singapore
106. Tsarfaty R, Seddah D, Goldberg Y, Kuebler S, Candito M, Foster J, Versley Y, Rehbein I, Tounsi L (2010) Statistical parsing for morphologically rich language (SPMRL): what, how and whither. In: *Proceedings of the first workshop on statistical parsing of morphologically rich languages (SPMRL)* at NA-ACL, Los Angeles
107. Tsarfaty R, Nivre J, Andersson E (2012) Joint evaluation for morphological segmentation and syntactic parsing. In: *Proceedings of ACL*, Jeju Island
108. Tsarfaty R, Seddah D, Kübler S, Nivre J (2013) Parsing morphologically-rich languages: introduction to the special issue. *Comput Linguist* 39(1):15–22
109. Zwicky AM (1993) Heads, bases, and functors. In: Corbett G, Fraser N, McGlashan S (eds) *Heads in grammatical theory*. Cambridge University Press, New York/Cambridge, pp 292–315

# Chapter 4

## Semantic Processing of Semitic Languages

Mona Diab and Yuval Marton

### 4.1 Introduction

In this chapter, we cover semantic processing in Semitic languages. We will present models of semantic processing over words and their relations in sentences, namely paradigmatic and syntagmatic models. We will contrast the processing of Semitic languages against English, illustrating some of the challenges – and clues – mostly due to the inherent unique characteristics of Semitic languages.

It is a hard task to define what meaning is. Nevertheless, one can still reasonably and usefully define related, easier problems, such as identifying different word senses, or measuring the closeness in meaning of two words (or larger textual units). We generally call these easier problems *semantic processing* tasks, and distinguish two kinds: paradigmatic and syntagmatic semantics. *Paradigmatic semantics*, which studies the lexical meaning of words, i.e., the functional properties that contrast and separate them from other words. Paradigmatic semantics studies the conditions and appropriateness of substituting word (or phrase)  $w_1$  with another,  $w_2$ , in a certain context or contexts. The *context* is usually taken to be the surrounding words in the same sentence, often up to a fixed distance from  $w_1$ . Paradigmatic semantic relations between words include synonymy, antonymy and hyponymy [13]. On the other hand, *syntagmatic semantics* studies the conditions for combining words and creating meaningful structures.

Generally, the most complex component of semantic processing for Semitic languages lies in handling the morpho-semantics of the words, identifying the level

---

M. Diab (✉)

Department of Computer Science, The George Washington University, Washington, DC, 20052,  
USA

e-mail: [mtdiab@gwu.edu](mailto:mtdiab@gwu.edu)

Y. Marton (✉)

Microsoft Corp., 555 110 Ave NE, Bellevue, WA, USA

e-mail: [yumarton@microsoft.com](mailto:yumarton@microsoft.com)

of granularity at which to consider the unit of meaning given a task at hand, and then how to represent these units of meaning in a model in a manner that yields the most benefit. Hence in this chapter we will address both issues of granularity level and representation for the various semantic processing technologies. We will specifically focus on four areas: word sense disambiguation (WSD), paraphrase detection and generation (PDG), multi-word expression detection and classification (MWE-DC), and semantic role labeling (SRL). We will present an overview of some of the latest enabling technologies in these areas, with an eye on how processing Semitic languages impacts choice of algorithm and representation. Although named entity recognition, sentiment analysis, modality, and natural language understanding are all areas that are relevant for semantic processing, we will not consider them in the scope of this chapter. It is worth noting that specifically for Arabic, we are faced with more processing issues due to the pervasive use of dialectal Arabic in current textual genres. For the purposes of this chapter, we will not address the dialectal problem. However it should be noted that identifying the granularity level for processing and its implication on representation choice still holds.

## 4.2 Fundamentals of Semitic Language Meaning Units

Two issues that have significant impact on semantic processing of Semitic languages are (1) the underspecification of short vowels and gemination markers (aka. diacritics) in several of the writing systems (Hebrew, Arabic, Amharic), and (2) the richness of the morphology (word forms), leading to data sparseness, and typically coupled with freer word order (hence semantic role labeling techniques developed for English are less likely to perform well).

### 4.2.1 *Morpho-Semantics: A Primer*

Semitic languages are known for their rich morphology. For a comprehensive discussion of Semitic language morphology, we refer the reader to the chapter on Semitic morphology in this book. For our purposes, we define a word to be a space-delimited token in naturally written raw text. A word in Semitic languages packs more information than a typical word in a language such as English. A word in Semitic languages typically exhibits several morphological mechanisms: derivation, inflection, and agglutination – all described below.

#### Morphemes

New words can be derived from existing words or morphemes. A morpheme is the basic, minimal linguistic unit that bears meaning, regardless of whether it can

stand alone as a word. For example, from a linguistic perspective, the inflectional plural marker suffix *-s*, the lexical prefix *anti-*, or the word *table* are all morphemes; accordingly, the word *tables* is comprised of two morphemes: *table* + *-s*. There are three ways to combine morphemes, where typically one morpheme (called the *base*) is more central before the base (prefix or proclitic; e.g., in Arabic *Al* + ‘the’),<sup>1</sup> after the base (suffix; e.g., in Arabic *+At*, the feminine plural morpheme), or both before and after (circumfix; e.g., in Hebrew *ta + ... + uwA* present tense second person masculine plural morpheme, or *mA + ... + \$* the negation morphemes in some Arabic dialects).

## Derivational Morphology

The derivational mechanism typically takes two morphemes and creates a new word with a part of speech possibly different from that of any of the participating morphemes. For example, *speak* (verb) + *-er* → *speaker* (noun) in English, or *kamwut* ‘quantity’ (noun) + *-i* → *kamwuti* ‘quantitative’ (adj.) in Modern Hebrew. This mechanism can be affixival (merging the morphemes serially) or templatic (interleaving root and pattern).

The *root* is an ordered tuple of consonants, a.k.a. *radicals*, most often a triplet (but could be a pair or a quadruple). For example, the three radicals ك ت ب constitute a root related to writing in both Arabic and Hebrew.

Typically a root is unpronounceable. It is an abstract concept around which words cluster. Roots are not necessarily monosemic, i.e. a root could have multiple meanings. The vowels are added by merging with a pattern, a.k.a. template (see below). Roots and patterns are inherent parts of Semitic languages and unique to them. In some cases, glides such as the Semitic equivalents of *w* or *y* can be part of the root, and they sometimes disappear or are replaced by vowels in certain templates. If the second and third root members are identical consonants, they sometimes undergo gemination or merging, which also affects the form or sound of the resulting word.

In their derivational mechanism, Semitic languages are largely templatic, i.e., many derived words comprise of a root and a pattern. Sometimes an arbitrary element is added to the meaning of the resulting word, for example due to semantic drift in its meaning, with usage over time. The *pattern* is a fixed sequence of vowels and interleaved place-holders for the root’s consonants. For example *1a2a3* conveys a verb in past tense in Hebrew and Arabic, where the digits correspond to the root radicals. A pattern may also contain additional consonants. Like the root, the pattern is not necessarily monosemic. Patterns are typically a relatively small closed class. A root and a pattern are each a morpheme: the root carries a basic meaning of

---

<sup>1</sup>We will be using Buckwalter transliteration throughout this chapter to illustrate the Arabic script in Romanization.

**Table 4.1** Arabic root and template combination for the root ك ت ب k t b, ‘write’

Num	Template	Lemma	Word inflected		Inflection
			form	English gloss	
(1)	1a2a3	katab	katab <u>at</u>	She wrote	Verb, past.3rd.sing.fem
(2)	1A2a3	kAtab	takAtab <u>uw</u>	They corresponded	Verb, past.2nd.pl.masc
(3)	ma1o2a3	makotab	makotab	Office, desk	Noun, masc.sing
(4)	ma1A2i3	makAtib	makAtib	Offices, desks	Noun, masc.pl (broken plural)
(5)	ma1o2a3ap	makotabap	makotabap	Library	Noun, fem.sing
(6)	ma1o2a3ap	makotabap	makotab <u>At</u>	Libraries	Noun, fem.pl (sound plural)

an action or an object (in our example above, ك ت ب k t b is a root related to writing), while the pattern modifies the meaning over the root’s basic meaning, as in conveying a verb/action or a noun/object, mutual or reflexive action, etc. Some examples in Arabic are: كاتب kAtab ‘to correspond’, كتب kat ab ‘to dictate’, and كتاب kitAb ‘book (noun)’ – each having a different pattern, merged with the same root in this case.

## Inflectional Morphology

Semitic languages inflect for various features: number, gender, aspect, grammatical case, mood, definiteness, person, voice, and tense. The set of words sharing meaning but varying over these inflection features is called a *lexeme*. It is typically represented by one of the set members – referred to as the *lemma*. The lemma is used for linguistic purposes as the citation/head form in a dictionary, or as a feature in parsing or semantic processing. The lemma form is conventionally chosen as the masculine singular active perfective verb form for verbs and the masculine singular for nouns. Lemmas are fully diacritized where the short vowels and gemination markers are explicitly represented. The lexeme can inflect with certain combinations of these features depending on the part of speech (noun, verb, adjective, etc.) Table 4.1 illustrates some examples of words.

In many cases the meaning of the resulting word is compositional, i.e., it can be constructed from the meaning of the participating morphemes. In the example in Table 4.1, we note two types of plurals: sound plural as in مكتبات makotabAt, example (4), and broken plural as in مكتب makAtib, example (6). The sound plurals are typically predictable, hence their underlying lemma is the same as the singular form of the word. Conversely the broken plurals are for the most part unpredictable and different from the general (sound) affixival case, hence their underlying lemma is often represented as the broken form. In some lexical resources, we note that in order to render a functional representation for the entries, the singular form is also

**Table 4.2** Arabic surface word *wabiHasnAthm* fully diacritized and analyzed: وَحِسْنَاتِهِمْ  
*wabiHasanAtihim*

Morpheme	Type	Morphological class	POS	Gloss
wa	Proclitic	Agglutinative	Conjunction	‘and’
bi	Proclitic	Agglutinative	Preposition	‘by’
HasanAti	Lexeme/stem	Inflectional	Noun fem. pl.	‘virtues’
Hasanap	Lexeme/lemma	Inflectional/derivational	Noun fem. sing.	‘virtue’
1a2a3ap	Pattern	Derivational	Noun fem. sing.	—
H s n	Root	Derivational	—	‘good’
him	Enclitic	Agglutinative	Masc. pl. poss. pron.	‘their’

represented as a valid lemma, hence for example (4), مَكَاتِبْ *makAtib*, the underlying lemma is linked with the lemma in example (3) مَكَتبْ *makotab*, similar to setting the lemma of the English irregular ‘went’ to be ‘go’.

### Agglutinative Morphology

Lexeme members could further agglutinate to certain closed class words (*clitics*), forming complex surface words in naturally occurring written Semitic texts (where in English, these would typically be space-delimited tokens). These agglutinative clitics may be particles (such as negation particles), prepositions, and grammatical markers of various sorts, such as aspectual and future morphemes, conjunctions, and pronouns, some of which occur as enclitics (suffixes) and others as proclitics (prefixes). It is worth noting that agglutinative morphology is more complex in dialects of Arabic than they are in Modern Standard Arabic. For example, the Arabic word وَحِسْنَاتِهِمْ *wabiHasnAthm*, ‘and by their virtues’, comprises the following morphemes as illustrated in Table 4.2.

Agglutinative morphology is different from derivational and inflectional morphology in that the resulting word retains a complex meaning, as in the example above: conjunction + preposition + the stem + possessive pronoun (equivalent to an English conjunction of a prepositional phrase), and may reach a complexity equivalent to a whole sentence in English.

### Vowel and Diacritic Underspecification

In Semitic languages, especially the major languages, Arabic, Hebrew and Amharic, written words are mostly underspecified for vowels and consonant gemination marking. These markers are known as diacritics. A correlate in English would be rendering the words ‘bear’, ‘bare’, ‘bar’, as ‘br’. In Semitic languages, the expansions could be a multitude of possibilities due to the various derivational forms, in addition to possible inflectional variations that affect the choice of

diacritic. For example, in Arabic, a lemma such as كَتَبَ *katab* ‘to write’, occurs in naturally written text as the undiacritized form كتب *ktb*. Yet, in context *ktb* could be the word كُتِبَ *kutib* ‘was written’, inflected for passive voice, or كَتَبَتْ *kattab* ‘to dictate’, in addition to the form كَتَبَ *katab* ‘to write’, both derivational variants (with no inflection to feminine and/or plural forms).

In Modern Hebrew the ambiguity is even greater than in Arabic, since Hebrew has more vowels, and relatively fewer templates with silent letters (hence fewer written clues). For example, the Hebrew letter sequence *dwd*<sup>2</sup> may stand for *dwod* (uncle), *dwud* (boiler), or *daVid* (David). Modern Hebrew speakers/writers developed a ‘fuller spelling’ (*Ktiv Malé*) scheme, which employs a higher rate of silent letters to indicate vowels, and a doubling of these letters to indicate when they are not silent. In this scheme, ‘David’ would be written as *dwyd* or *dwyd*. Still, although reduced, much ambiguity remains. The most widely used scheme today is a ‘mixed spelling’ scheme, employing some, but not all, of the ‘fuller spelling’ conventions, in an attempt to better balance ambiguity, brevity, and tradition. Another example is the Hebrew letter sequence *SrwT* may stand for *SeruT* (service), *SaroT* (singing, female plural), *CaroT* (ministers, female plural), or *Se + ruT* (that Ruth). In some writing schemes *SeruT* (service) is more commonly written with an additional ‘y’ *SyruT*, presumably to reduce ambiguity with the complementizer ‘S’ (that).

There exist several tools for diacritic restoration (and hence disambiguation) with reasonable success, primarily for Arabic [45, 81, 94, 116] in addition to a system by Sakhr software that claims a diacritization accuracy rate of 97 % [115]. State-of-the-art full diacritization of Arabic, ignoring grammatical case and mood, is in the >95 % accuracy range by the most popular two approaches [116] and [94]. The approach by Zitouni et al. [116] uses a maximum entropy supervised model for diacritic restoration without resorting to any underlying morphological resources such as an analyzer. Contrastively, the approach by Roth et al. [94] which is based on the prior work of [45], is a supervised approach using Support Vector Machines (SVM) that relies on a morphological analyzer which exploits a fully diacritized dictionary to render the full diacritization of words in context. Although both approaches achieve high performance, automatic diacritization still needs significant improvement to show significant positive impact on text processing and NLP applications. For example, in machine translation, a study by Diab et al. [28] shows that partial diacritization for Arabic has positive impact on MT quality,

---

<sup>2</sup>We use the following transliteration for Hebrew letters: *AbgdhwzHtyklmnEpcqrST*. When denoting pronunciation, some letters may denote one of two sounds each. In such cases, they may be transliterated as follows: b→v, k→x, p→f, non-silent w → V (sounds the same as v but denoted uniquely for disambiguation), non-silent y → Y, (h is non-silent unless in final position, so no special denotation is needed), and S→C (sounds the same as s but denoted uniquely for disambiguation). Non-letter (diacritic) vowels are transliterated with the lowercase letters *aeiou* (simplified as well).

however there is still significant room for improvement. The authors show that improving passivization diacritic restoration would have a significant impact on MT performance, however state-of-the-art passivization discovery is in the 68–72 % F-score level [25, 44]. In a study by Zbib et al. [115], the authors show a positive impact on MT performance when using diacritization as a smoothing technique. In Language modeling for automatic speech recognition, [110] show positive impact on performance by using diacritization. These are expected results as restoring diacritics is the first form of disambiguation.

### Form vs. Function Morphology

The morpheme feature forms and their functions do not necessarily have to be congruent in Semitic languages. For example, the Arabic broken plurals of some masculine singular nouns have a feminine form but function as masculine plurals in their agreement, as pointed out in [43]: كِتَابَاتٌ *katabat* ‘writers’ is the plural *masculine broken form* of كَاتِبٍ *kAtib* ‘writer’. In form it ends with ة p which is typically the feminine ending for singular nominals. Likewise, the word for ‘celebrations’ احتفالات *AHtifAlAt* has the feminine ending form At typically associated with sound feminine plural forms. The latter word is the plural form of the masculine noun احتفال *AHtifAl*. Both احتفالات *AHtifAlAt* and كِتَابَاتٌ *katabat* have feminine forms, yet they function as masculine plurals in terms of their agreement functions. This has implications on syntagmatic relations as explored in the realm of semantic role labeling in Sect. 4.6.

## 4.3 Meaning, Semantic Distance, Paraphrasing and Lexicon Generation

It is a hard task to define what meaning is. Nevertheless, one can still reasonably and usefully define related, easier problems, such as measuring the closeness in meaning of two words (or larger textual units). This easier problem is also known as *semantic similarity*, or *semantic distance*.<sup>3</sup> It is essential for many NLP tasks and applications, such as (web) search, question answering, information retrieval and extraction, plagiarism detection, paraphrase detection, and so on.

---

<sup>3</sup>Some researchers distinguish between similarity and relatedness [49]; without getting into this distinction, we use here a general notion of semantic distance measures.

### 4.3.1 Semantic Distance

More formally, a *semantic distance measure* is a function  $(x, y) \rightarrow [0 \dots 1]$ , where  $x$  and  $y$  are two given words in language  $L$ , and the distance typically ranges from 0 (totally unrelated) to 1 (highly synonymous).

Estimating semantic distance has been carried out using a lexical resource, statistical machine translation methods (pivoting), corpus-based distributional methods, or hybrid methods.

#### Resource-Based Measures

WordNet-based measures [49] consider two terms to be close if they occur close to each other in the network (connected by only few arcs [55, 89]), if their definitions share many terms [7, 86], or if they share a lot of information ([61, 91] – which are in fact hybrid methods). WordNet sense information has been criticized to be too fine grained or inadequate for certain NLP tasks [2, 79]. Further, the measures can only be used in languages that have a (sufficiently developed) WordNet.

#### Pivot Measures

Given *parallel texts*, where each sentence is aligned to its translation, create statistical translation models (whose main component is called the *phrase table*), and translate  $x$  to the other language and back [9]. The semantic distance between  $x$  and  $y$  can be estimated by marginalizing their bidirectional translation probabilities  $\sum_j p(x|f_j)p(f_j|y_i)$ , where  $f_j$  are the pivot translations in the other language(s).

#### Distributional Measures

Corpus-based semantic distance measures [109], rely on the *distributional hypothesis* [36, 47]: words tend to have a typical distributional profile: they repeatedly appear next to specific other words in a typical rate of co-occurrence. Moreover, words close in meaning tend to appear in similar contexts (where context is taken to be the surrounding words in some proximity). Therefore, the meaning of word  $x$  can be represented in a vector, where the  $i$ th cell contains a distributional strength-of-association (SoA) measure between  $x$  and the  $i$ th word in the vocabulary of a large monolingual text corpus. Common SoA measures are conditional probabilities [99], mutual information (PMI) [61], TF/IDF-based [38], and the likelihood ratio [31]. Such a vector is often called the *distributional profile* of  $x$ . The semantic distance between  $x$  and  $y$  can then be estimated from the distance between their distributional profiles. Any vector similarity measure may be used, e.g., the cosine function, the

Jaccard coefficient, the Dice coefficient (all proposed by Salton and McGill [97]),  $\alpha$ -skew divergence [16], and the City-Block measure [90].

## Hybrid Methods

Resnik in [91] introduced a hybrid model for calculating “information content” [93]. In order to calculate it for a certain concept in the WordNet hierarchy, one traverses the concept’s subtree, and sums the corpus-based word frequencies of all words under that concept, and all concept nodes in that subtree, recursively. A maximum likelihood log-probability estimation is then calculated by dividing that sum by the total number of word occurrences in the corpus, and taking the negative log. Patwardhan and Pedersen in [86], Mohammad and Hirst in [70] and Marton et al. [68] proposed measures that are not only distributional in nature but also rely on a lexical resource (a mid-size thesaurus). They treat each list of related words (‘concepts’) in the thesaurus as a word sense, and extend these senses to the entire vocabulary of a large corpus by marking known senses in the corpus, and assigning a majority-vote sense to words that are not in the thesaurus. Then, given two words ( $x, y$ ), they build distributional profiles per word sense, measure the distance between each sense of  $x$  to each sense of  $y$ , and return the smallest.

## Paraphrase Generation

Paraphrase generation is a harder task than measuring semantic distance. This is because only one word,  $x$ , is given, and the goal is to provide  $y$ , or a set of  $y$ ’s – and typically also the distance of each  $y$  from  $x$  (or at least, a ranking order of the  $y$ ’s). More formally, a *paraphrase generator* is a function  $x \rightarrow \langle y_i, z_i \rangle$ , where  $x$  is the given word,  $y_i$  are the paraphrase candidates, and  $z_i$  is the semantic distance of  $y_i$  from  $x$  as defined above, such that  $z_i \leq z_{i+1}$  for all  $i$ .

Paraphrases can be generated using statistical machine translation techniques (pivoting), or distributionally. **Pivot:** Given a *phrase table* (as described above), for languages  $L$  and  $L'$ , translate  $x$  in  $L$  to  $L'$  and back to  $y_i$  in  $L$  [9]. Paraphrase candidates  $y_i$  can be ranked by their marginalized bidirectional translation probabilities  $\sum_j p(x|f_j)p(f_j|y_i)$ , where  $f_j$  are the pivot translations in the other language(s)  $L'$ . **Distributional:** Given a large monolingual text corpus, collect the contexts in which  $x$  occurs, and then collect candidates  $y_i$  occurring in same contexts. Rank candidates by the distance of their distributional profiles (see Sect. 4.3.1) [67, 85]. **Other methods:** Paraphrases can also be generated in other ways, for example, using a lexical resource such as WordNet or a thesaurus. However, these resources typically suffer from incomplete coverage, especially for longer-than-word terms (a.k.a. *multi-word expressions*). Last, these methods may be combined, e.g., paraphrasing using a thesaurus together with distributional method [66].

## A Language-Independent Example Algorithm for Paraphrasing

To date, pivoting is a more established paraphrasing method, but its application is limited to languages with large parallel text resources. While it may have limited relevance to Arabic, its current applicability to other Semitic languages is even lower. Therefore, we outline here the more relevant distributional paraphrasing method:

1. Upon receiving phrase  $phr$ , build distributional profile  $DP_{phr}$ .
2. For each occurrence of  $phr$ , keep surrounding (left and right) context  $L\_R$ .
3. For each such context  $L\_R$ , gather all paraphrase candidates  $cand$ , such that  $L \ cand \ R$  occurs in the training corpus; i.e., gather paraphrase candidates occurring in same contexts as  $phr$ .
4. Rank all candidates  $cand$  according to their semantic distance from  $phr$  by building profile  $DP_{cand}$  and measuring profile (vector) distance between  $DP_{cand}$  and  $DP_{phr}$ .
5. Optionally: Filter out every candidate  $cand$  that textually entails  $phr$ . For example, if  $phr$  is *spoken softly*, then *spoken very softly* would be filtered out.
6. Output k-best candidates above a certain similarity score threshold.

However, to date, this has not been applied to a Semitic language.

### 4.3.2 Textual Entailment

Textual entailment (TE) can be defined as follows: given a phrase or a sentence  $x$  and a set of hypotheses  $\{h_i\}$ , predict whether – or the likelihood whether – each  $h_i$  can be reasonably entailed or deduced from  $x$  (with or without larger context). For example,  $x = \text{"she ran quickly"}$  entails  $h_1 = \text{"she moved"}$  ( $x \rightarrow h_1$ ) or even  $h_2 = \text{"she was in a hurry"}$  ( $x \rightarrow h_2$ ). Note that the opposite direction (e.g.,  $x \leftarrow h_1$ ) does not necessarily hold. Paraphrases can be viewed as symmetric TE [39], i.e.,  $x \leftrightarrow h_i$ . TE methods usually rely on hierarchical lexicons (or *taxonomies*) such as WordNet, since the common *is\_a* relation is entailing in nature (for example, if *running* is kind of *moving*, then *running*  $\rightarrow$  *moving*). This relation is hard to deduce from non-annotated text.

For more on paraphrasing and textual entailment, please refer to [4, 39, 65].

### 4.3.3 Lexicon Creation

The task of extracting all paraphrases from a given corpus (and perhaps a seed lexicon) is sometimes called *paraphrase extraction* – or *TE pair extraction* for TE [4]. Here, neither  $x$  nor  $y$  are given, and the desired output is a set of triplets  $\langle x_i, y_i, z_i \rangle$ , where each  $x_i, y_i$  are paraphrases or a TE pair, and  $z_i$  is their

optional semantic distance or entailment confidence score. The lexicon creation is often viewed as a side-effect of paraphrase generation capabilities, traversing the vocabulary. Note that for covering multi-word entries, this task may become exponential in the maximal entry length.

To date, we have not noted the use of semantic distance, paraphrasing, or textual entailment for semitic languages, most probably due to the lack of sufficient resources in terms of raw parallel or monolingual corpora for most Semitic languages (with the exception of Arabic). Accordingly, we do believe there is ample room for algorithmic applications especially where there are fewer dependencies on language-specific resources.

Semantic distance, paraphrasing and lexicon creation all need to take into account the fact that many words (and larger units) may have more than a single sense each. Ignoring this fact is bound to hurt accuracy and usefulness. In the next section we discuss word senses, how to define them, how to detect them – and specific challenges in Semitic languages.

## 4.4 Word Sense Disambiguation and Meaning Induction

Word sense disambiguation or discrimination (WSD) is the task of classifying a token (in context) into one of several predefined classes [80, 111]. WSD has been an active research area for decades. See for example the SemEval and SensEval workshops.<sup>4</sup> For example, given the classes *Financial-Institute* and *River*, and the token *bank* in the context ‘we strolled down the river bank’, output 1 for *Financial-Institute*, or 2 for *River*; in this example the correct sense is sense 2. A more recent variant of WSD is the word sense induction task (WSI), in which the sense classes are not pre-defined, and instead, need to be induced from the data, using unsupervised clustering techniques [98]. The idea is that word senses that have the same meaning cluster together. Pantel and Lin [84] perform word sense induction from raw corpora where they find most similar words to a target word by clustering the words, each cluster corresponds to a sense. There have been several studies and approaches to the problem, all of which use unsupervised clustering for discovering ambiguous words and grouping their senses together. For a nice review the reader can refer to [95]. WSD/WSI is considered one of the hardest tasks in artificial intelligence (AI). It often requires not only linguistic knowledge, but also knowledge of the world (facts). For example, we use world knowledge to decide that the intended sense of ‘bass’ in ‘they got a grilled bass’ is a fish, and not a musical instrument (since we know that typically one would grill fish, not instruments).

Sadly, our world knowledge is not yet fully compiled to machine-readable format. However, recent research efforts aim to extract such knowledge from free

---

<sup>4</sup><http://nlp.cs.swarthmore.edu/semeval/> or <http://www.senseval.org/>

text knowledge bases such as Wikipedia,<sup>5</sup> at least in the form of  $\langle X, \text{relation}, Y \rangle$  tuples, e.g.,  $\langle \text{New York}, \text{in}, \text{United States} \rangle$  [8, 112]. We will not address the issue of world knowledge in the context of this chapter.

WSD and WSI in Semitic languages such as Hebrew or Arabic, pose greater challenges than in English. This is due to the fact that (1) in many cases short vowels are only represented via diacritics that are often omitted in modern writing, and (2) several frequent prepositions, and many types of pronouns (e.g., possessive or prepositional pronouns) are expressed as agglutinated affixes. Hence the biggest challenge for Semitic language semantic processing for WSD is determining the appropriate unit of meaning that is relevant for WSD/WSI. Being an enabling technology, in most cases, this decision should depend on the end application requirements and the availability of preprocessing resources and the amount of data. To date, most approaches have relied on simple tokenization of agglutinative clitics, leaving stems in the form of an inflected lexeme member, with no further tokenization. In one study, however, we note the use of roots [33]. Using the lemma seems a promising level of representation, since it is not as sparse as the stem and the word-form, and yet not as underspecified and generic as the root. Lemma-based approaches are currently being worked on. Tools for full lemmatization (mainly for Arabic) are just becoming available.

#### **4.4.1 WSD Approaches in Semitic Languages**

For Arabic, several approaches have been proposed. In unsupervised disambiguation, one of the earliest efforts to tackle Arabic WSD relies on projection across parallel corpora of WSD classes from English to Arabic [18, 19, 24]. This approach requires the availability of a parallel corpus of Arabic and a language with a dictionary/knowledge base such as WordNet [35]. WordNet [35] is a manually-created hierarchical network of nodes, where each node represents a word sense, and each edge between nodes represents a lexical semantic (paradigmatic) relation such as hypernymy (*is-a*). In this study, the authors use an English–Arabic parallel corpus. Initially senses are assigned to the English side of the parallel corpus by using the fact that several words in English map to the same word in Arabic: *bank*, *shore*, *riverside* correspond to the Arabic word *بَحْرٌ* \$T. The three English words are then disambiguated by maximizing the possible overlap between the chosen word senses. Hence *bank* is assigned the *riverside* sense. Once the senses are assigned to the English side of the parallel corpus, through word alignments, the senses are projected onto the Arabic side of the corpus. The results are evaluated against gold annotated data where the annotations use English WordNet classes assigned to the Arabic words. The performance of the system is almost 65 % F-score. The data is

---

<sup>5</sup><http://www.wikipedia.org/>

not within a specific domain. The Arabic data was simply white space tokenized text with no further preprocessing.

The approach is inspired by earlier work by Dagan and Itai [15] who propose a probabilistic approach to WSD, using parallel corpora for Hebrew and German. They employ a syntactic parser and a bilingual lexicon. They present a statistical model assuming a multinomial model for a single lexical relation and then using a constraint propagation algorithm that accounts simultaneously for all relations in the sentence. They report very high results (over 91 %) but within a narrow domain, hence leveraging the inherent frequency of usage for specific senses of the words in context.

More recently for Arabic, Elmougy et al. [33] use a Naive Bayes approach coupled with an algorithm that pre-processes words to roots. The approach is supervised, where they use 1,600 roots for training. They report accuracies of 76 %. Furthermore, Merhbene [69] and Zouaghi [117] propose an unsupervised system for Arabic WSD using a context-matching algorithm together with a semantics coherence score, indicating the closest fit of the word sense to the context, akin to the Lesk algorithm [57], which has been quite popular in the WSD literature. The basic idea for the Lesk algorithm is to measure the word overlap of the context of a word  $w$  in a given sentence against each sense definition of  $w$  in a dictionary. The study investigates 10 words and they achieve an accuracy of 78 %. One issue with all the work on Arabic WSD is the problem of researchers not using a standard data set to allow for benchmarking.

#### 4.4.2 WSI in Semitic Languages

In general, the approaches are language independent. The main concern is the preprocessing of the data. We note two studies on Hebrew WSI, by Levinson [59], and more recently by Rozovskaya and Sproat [95]. In the former, the author compares two languages, English and Hebrew. He clusters the target words of interest with the 50 most similar words into groups. The results of the clustering are taken to be the number of senses. This study reports comparable results on English and Hebrew. In the latter work, the authors compare English to Hebrew and Russian. They report results of assigning the words to the correct sense cluster with an F-score of 66 %. For Russian they apply morphological lemmatization and show significant improvements (a 17 % error reduction), however they do not replicate the results for Hebrew in this study.

#### Knowledge Base Resources for Arabic WSD

The Arabic WordNet (AWN) [32] was built following the EuroWordNet method of devising a set of base concepts while aiming at maximizing the compatibility with other WordNet databases. Accordingly by design, the AWN is completely mapped

to the English Princeton WN 2.0 [35]. Moreover the AWN has links for its synsets in the Suggested Upper Merged Ontology (SUMO) [82]. This representation serves as an interlingua of sorts. AWN started off with a complete manual population, then later in 2008, [92] semi-automatically extended it. It currently comprises 11,270 synsets (7,961 nominal, 2,538 verbal, 661 adjectival, and 110 adverbial synsets), containing 23,496 Arabic expressions. This number includes 1,142 synsets of named entities.

There exist two Arabic data sets that could be used for evaluation, which are annotated against the AWN: the SemEval 2007 data set proposed by Diab et al. [26]. The data set comprises 888 verb and noun instances: 211 verbs and 677 nouns using the AWN.<sup>6</sup> The second data set is the Arabic Ontonotes sense annotated data,<sup>7</sup> released in 2011. The latter is much larger in scope, and comprises treebanks, co-reference resolution, and sense disambiguated data. The sense annotated data is of the newswire genre. It covers 200,000 words. The effort also includes linking senses to an ontology, Omega.<sup>8</sup> For Arabic, the total number of noun types is 111, and verb types is 150.

### **Knowledge Base Resources for Hebrew WSD**

We note the preliminary work on creating a Hebrew WordNet [83]. They use an alignment technique with the English WordNet. The Hebrew WN comprises a total of 5,261 synsets corresponding to 4,090 nouns, 609 verbs, 779 adjectives, and 151 adverbs.

### **Resources for Amharic WSD**

To date we have not come across WSD for Amharic. There are some efforts to create a WordNet in Amharic. As for WSI methods applied to Amharic, the research community is still creating corpora of significant sizes to be used with these approaches in this language.

## **4.5 Multiword Expression Detection and Classification**

In the spirit of extending the single word unit, we explore the notion of multiword expressions. The meaning (semantics) of linguistic expressions has been thought of as primarily compositional from the lexical meaning of the participating

<sup>6</sup>Licensing for this data set is obtained through the Linguistic Data Consortium (LDC).

<sup>7</sup><http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2011T03>

<sup>8</sup><http://omega.isi.edu>

elements (words, morphemes) and their order or syntactic relations (with only few exceptions), at least from the time of Frege [37]. A multi-word expression (MWE) refers to a longer-than-word unit, or a collocation of words that co-occur together statistically more than chance. A MWE is a cover term for different types of collocations, which vary in their transparency and fixedness. MWEs are pervasive in natural language, especially in web-based texts and speech genres. In fact, we found that in English data where MWEs (all types) are identified as single tokens, MWEs cover 5 % of the tokens in a corpus, however they account for 43 % of the types. Hence processing them and understanding their meaning is of crucial importance to language understanding and essential for the development of robust NLP applications. In fact, the seminal paper [96] refers to this problem as a key issue for the development of high-quality NLP applications. Recently, [10] show that modeling MWEs explicitly in the machine translation pipeline leads to significant gains in performance.

MWEs are classified based on their lexicographic, syntactic and semantic constructions. Six types of MWE have been identified in the literature [21,22,96]: Noun Noun Compounds (NNC) ‘traffic light’, Verb Noun Constructions (VNC) ‘spill the beans’, Verb Particle Construction (VPC) ‘set up’, Light Verb Constructions (LVC) ‘make a decision’, Fixed Constructions (FC) ‘above and beyond’, ‘by and large’, and Named Entities Constructions (NEC) ‘John Taylor Smith’. The different MWEs vary in their level of compositionality and their fixedness; some allow for intervening words and/or morphological inflections. The level of transparency varies by MWE especially for NNC and VNC categories. For example, ‘she kicked the bucket’ could denote either a kicking event (compositional sense) or a death event (non-compositional, idiomatic sense). The level of compositionality depends on the context. Hence several research papers have addressed the problem of token level classification where the instance of the MWE is identified as idiomatic vs. literal especially for VNCs [20–22].

#### **4.5.1 Approaches to Semitic MWE Processing and Resources**

For Semitic languages, most of the work to date has been focusing on the creation of MWE resources for the languages with two exceptions: for Arabic [48] and for Hebrew [3].

#### **Arabic MWE Processing**

In this preliminary work, [48] present a method of pattern matching over instances in the Arabic Giga Word 4.0 (AGW).<sup>9</sup> The authors lemmatize the entire corpus

---

<sup>9</sup><http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2009T30>

and they match all the entries in an extensive hand-built MWE lexicon (also lemmatized) against the corpus tokens. The matching algorithm allows for gaps such as اغْمَضْ فَلَانْ أَيْنِيهُ أَنَّ الْأَمْرَ *AgmD (flAn) Eynyh En (AlAmr)*, ‘(one) disregarded/overlooked/ignored(the issue)’, literally, ‘closed one’s eyes’. In this context, *flAn* ‘one’ and *AlAmr* ‘the issue’ are wild cards that could be filled with any appropriately matched replacement from the corpus. The pattern matching algorithm could also handle inflectional variants such as اغْمَضْتْ فَلَانَةً عَيْنِهَا عَنَ الْأَمْرَ *AgmDt flAnp EynyhA En AlAmr* where the subject is feminine. The pattern matching algorithm performs on all above-mentioned types of MWE constructions. The resulting corpus is an automatically classified corpus comprising 576 LVC, 64,504 VNC, 75,844 VPC, and 340,107 NNC. It is worth noting that several of the MWEs are FC. A sample of the automatically detected MWEs are evaluated manually. The results are 98 % accuracy for VNC, 99 % accuracy for NNC, 77.6 % accuracy for VPC.

## Resources

Most of the resources created to date are for Modern Standard Arabic. The study by Attia et al. [5] focuses on creating a large resource for nominal MWEs (NNC), which are comprised of noun noun compounds such as دوَدَةُ الْقَطْنِ *dwdp AlqTn* ‘cotton worm’, noun adjective constructions (NAC) such as اسْعَافَاتُ اُولَئِكَ *AsEAfAt AwAlyp* ‘first aid’, Named preposition noun constructions such as التَّزَلُّقُ عَلَى الْجَلِيدِ *Altzlq ELY Aljlyd* ‘ice skating’, noun conjunction noun constructions such as الْأَكَالُ وَالْبَنُونُ *AlmAi wAlbnwn* ‘wealth and children’. In their study, they mine these nominal MWEs using three approaches: cross lingual asymmetries, translation based extraction and some collocation based statistics over corpora. They carry out statistical and manual validation of the output. They collect an impressive 34,658 MWEs as well as 45,202 named entities. In the recent study by Hawwari et al. [48], they manually curated and collected 2,005 VNC, 670 VPC, 1,524 NNC types.

## Hebrew MWE Processing

In the study by AlHaj and Wintner [3], the authors present an extensive study of MWE NNC of the type *haHlaTat hawaEadah* ‘the committee decision’, *Eowrex haEytown* ‘the journal editor’, *bateY Howlym* ‘hospitals’. They present a supervised system (using support vector machines) that classifies NNC compounds as MWE or not. They experiment with 463 instances, 205 of which are positive examples. The examples are manually collected and classified by three annotators. They model extensive linguistic features inspired by the lexicographic and syntactic nature of the Hebrew language. They report significant improvement (78 % F-score) over random

baselines as well as non-specifically linguistic baselines (63 % F-score). The focus of this study was solely on NNC MWEs.

## Hebrew Resources

A recent resource for Hebrew MWEs is [108]. In this study, the authors focus on extracting nominal MWEs from parallel corpora exploiting non-corresponding alignments in small parallel corpora coupled with monolingual resources. They evaluate the impact of augmenting the dictionary in an SMT framework with the MWEs. They report significant gains in MT performance when the MWEs are used explicitly in the models.

## 4.6 Predicate–Argument Analysis

Analyzing sentences syntagmatically is believed to give us deeper understanding of natural language. Syntactic parsing has been a subject of active research for several decades. Associating a semantic meaning with the syntactic parse is the next logical step toward developing an even deeper insight into meaning and how it is being used. Shallow approaches to semantic processing are making large strides in the direction of efficiently and effectively deriving tacit semantic information from text. Crucially these approaches go beyond the paradigmatic to the utterance/sentential level. Identifying and defining roles of predicate arguments in a sentence has a lot of potential for and is a significant step toward improving important applications such as document retrieval, machine translation, question answering and information extraction [75].

In shallow semantic analysis, predicate argument structure is analyzed where the arguments are associated with their roles in context. Current solutions address the problem by reducing it to the identification of the predicates and their arguments and then classifying the arguments into a set of argument types. For example, in the English sentence, ‘Mark eats bananas.’, the predicate is ‘eats’ whereas ‘Mark’ is the *agent* role for the predicative event and ‘bananas’ is the *theme* role. The crucial fact about semantic roles is that regardless of the overt syntactic structure variation, the underlying semantic roles remain the same. Hence, for the sentence ‘Bananas were eaten by Mark’, ‘bananas’ is still the *theme* regardless of the syntactic position in the sentence. Likewise for the causative/inchoative alternations for instance: ‘John opened the door’ and ‘the door opened’, though ‘the door’ is the object of the first sentence and the subject of the second, it is the *theme* in both sentences. The semantic role labels may differ depending on the linguistic theory or annotated resource adopted. In FrameNet (FN) [6] for instance, ‘Mark’ is labeled *eater* and ‘bananas’ is the *thing being eaten*, whereas in ProbBank (PB) [52] ‘Mark’ would be labeled *ARG0* and ‘bananas’ *ARG1*.

There have been several attempts at inducing this structure from tacit knowledge. Several works have addressed the problem using unsupervised approaches for verb subcategorization [14, 56, 62, 100, 101], and more recently unsupervised semantic role labeling (SRL) [1, 17, 42, 54, 87, 107], among others. With the advent of faster and more powerful computers, more effective machine learning algorithms, and importantly, large data resources annotated with relevant levels of semantic information,<sup>10</sup> we are seeing a surge in efficient supervised approaches to SRL [11, 12, 40, 41, 46, 71, 88, 106, 113] especially for English, with several notable exceptions for German [34] and Chinese [60, 104, 105]. The systems for the other languages follow the successful models devised for English.

However there is a significant performance drop for approaches in Semitic languages. The problem may be attributed to the lack of NLP tools and corpora for Amharic and to a lesser extent for Hebrew, in level and size comparable to the state-of-the-art for English. But for Arabic, there exist advanced tools for NLP processing as well as large corpora, and yet the performance gap still remains. Other potential factors include the richer morphology and the associated data sparseness and freer word order. Exploring unsupervised approaches for Arabic seems to be a promising route to undertake, especially given the abundant number of corpora for that variety of Semitic languages.

#### 4.6.1 Arabic Annotated Resources

We know of two major Arabic lexical resources. The Arabic Proposition Bank (a.k.a. Arabic Propbank, or APB) [114], which was released as part of the Ontonotes 4.0,<sup>11</sup> and was the basis of the SemEval 2007 Task 18 data set [26]; and the more recent Arabic VerbNet (AVN) [77].

##### **APB**

The APB is a database of verb annotations and their predicate argument structure. It comprises 1,955 Frame Files with 2,446 framesets, where a frame file corresponds to a predicate and a frame set corresponds to the sense of a predicate with a unique argument structure. All the data in the Arabic Propbank is annotated against the Penn Arabic Treebank v3.2. newswire corpus [64]. The APB has frame files as a lexicon of entries and an associated corpus with propbank annotations. The following is an example of how a frameset is rendered in the lexicon for one of the senses of the verb ‘to listen’ *إستمع* /isotamaE/

---

<sup>10</sup>Especially for English, notably the above-mentioned FrameNet (FN) [6] and ProbBank (PB) [52].

<sup>11</sup><http://www.ldc.org/Ontonotes>

- Predicate: {isotamaE **إِسْتَمَعَ**
- Roleset id: f1, to listen
- Arg0: entity listening
- Arg1: thing listened

It should be noted that all the entries in the APB are in lemma form. In the APB corpus, a sentence such as ‘they listened to their demands.’ **إِسْتَمَعُوا إِلَى مَطَالِبِهِمْ** *isotamaE /lY mTAlbhM.*:

- Rel: {isotamaE, **إِسْتَمَعَ**
- Gloss: to listen
- Arg0: -NONE- \*
- Gloss: they
- Arg1: **إِلَى مَطَالِبِهِمْ** */lY mTAlbhM,*
- Gloss: to their demands
- Example Sentence: **إِسْتَمَعُوا إِلَى مَطَالِبِهِمْ** */isotamaE /lY mTAlbhM.*

### Arabic VerbNet

AVN is a lexical resource recently manually created for Arabic predicate argument structure. It was later extended semi-automatically for wide coverage [78]. It emulates the English VerbNet [53] which clusters predicative entries according to the Levin classes [58]. The Levin classes group predicates according to both syntactic and semantic attributes. The grouping is based on shared diathesis alternations. The hypothesis is that predicates that participate in similar diathesis alternations and share syntactic structure share some element of meaning. This hypothesis was put forward by Jackendoff [50, 51]. There are claims that this holds universally. The English VerbNet is an implementation of the Levin classes with extensions to new classes. It comprises 3,769 verbs corresponding to 5,257 senses and 274 primary classes.<sup>12</sup> The AVN is of comparable size to the English VerbNet. It comprises 336 classes and 231 subclasses, corresponding to 7,744 verbs, 7,815 deverbals, 7,770 participles. The number of overall frames is 1,399. Crucially, Mousser [78] introduced the notion of sibling classes as subclasses to handle morphological variants that are productive but bear the same meaning as the original class members, for example a verb such as **عَوِّلَ** *Eawlama* ‘globalize’ as an eventive verb has the sub-class **تَعَوِّلُ** *taEawlom* ‘to be globalized’ which is a stative verb. Interestingly, they found that 65 % of the diatheses alternations from English hold in Arabic.

Prior to this work on AVN, Snider and Diab [102, 103] attempted to induce predicate argument structure using unsupervised clustering methods. They created

---

<sup>12</sup><http://verbs.colorado.edu/~mpalmer/projects/verbnet.html>

partially gold clusters for evaluation from treebanked data and mapped them onto the Levin classes using the English gloss information in the treebank entries. They used frame-semantic categories by grouping different syntactic frames into four possible frame forms, templatic patterns for the verbs, a latent semantic analysis feature (which is basically a dimensionality reduction feature), and finally a subject animacy feature. They achieved a 47 % F-score on the clustering against a random baseline of 26 %.

#### 4.6.2 Systems for Semantic Role Labeling

The typical approach to automatically assigning semantic role labels to predicate arguments in context is by first identifying arguments and their boundary spans and then classifying these discovered arguments into classes. The classes may be derived from PB, FN, VN or any other formalism. Most systems to date have been using PB. For Semitic languages, due to lack of resources until recently, the systems developed use the APB for Arabic. We are not aware of any systems developed for other Semitic languages. The only effort to date for building a FN for Hebrew is described as preliminary steps in work by Petrucci.<sup>13</sup>

Semitic languages pose a significant challenge to SRL for several reasons. From a syntactic perspective, the following facts are relevant for SRL.

- *Pro-drop* Semitic languages tend to drop the subject and mark it morphologically on the verb. For example, أكلوا البرتقال *AklwA AlbrtqAl* ‘[they] ate\_they the orange(s)’ is pro-dropped where the pronoun ‘they’ is absent. The explicit rendering of the sentence would be as follows هم أكلوا البرتقال *hm AklwA AlbrtqAl* ‘they ate\_they the orange(s)’. This is a significant phenomenon for SRL since one of the arguments is missing hence the system has to either be able to make it explicit or mark a trace with the argument class as well as identify the morpheme marker on the predicate. This is especially tricky with third person masculine singular cases since there is no explicit morpheme marker on the predicate.
- *Relative free word order* Semitic languages allow for Verb Subject Object (VSO), Object Verb Subject (OVS), SVO, etc. The canonical word order in Arabic is VSO however in a study by Maamouri et al. [63], we found that the Modern Standard Arabic treebank has an equal distribution of SVO (35 %) and VSO (35 %) and pro-drop (30 %). This poses a significant challenge since it has implications in identifying the arguments and their roles, especially when the arguments are proper nouns with no explicit grammatical

---

<sup>13</sup><http://www.icsi.berkeley.edu/pubs/ai/HFN.pdf>

case marking due to diacritic underspecification especially in the singular cases.<sup>14</sup>

- **Possessive constructs** Possessive constructs (e.g., Arabic *idafa* or Hebrew *smyxuwt*) are especially tricky in Semitic languages as they allow for multiple recursive embeddings in the form of *idafa* chains as well as allowing for multiple modifiers. This is problematic for identifying argument spans. An example is the following: مَلِكُ الْيَمَنِ *mlk Alyaman* ‘king of Yemen’, or مَلِكُ الْيَمَنِ الْمَحْبُّ الْمَجَالِيْمِ *mlk Alyaman Almubajal* ‘The great respected king of Yemen’ where the modifiers could be multiplied, in principle, indefinitely. Another example is إِبْنُ عَمِ الْخَالِ صَاحِبِ الْمَدِيرِ الْجَرِيدَةِ */bn Eam xAl SAHib mdyr Aljrydap* ‘the cousin of the uncle of the chairperson of the newspaper’s friend’. These *idafa* constructs are quite difficult to parse and hence pose a significant challenge to boundary detection for the argument spans.

These syntactic phenomena play a crucial role in SRL systems. The complexity is magnified with the absence of functional diacritics that mark grammatical case, mood, passivization, definiteness, and agreement. We list here some examples.

1. **Grammatical case** For example, رَجُلُ الْبَيْتِ الْكَبِيرِ *rjl Albyt Alkbyr* ‘man<sub>[masc]</sub> the-house<sub>[masc]</sub> the-big<sub>[masc]</sub>’ could mean ‘the big man of the house’ or ‘the man of the big house’. For it to be ‘the big man of the house’, the case marking would be رَجُلُ الْبَيْتِ الْكَبِيرِ رَجُلُ الْبَيْتِ الْكَبِيرِ *rjlu Albyti Alkbyru* ‘man<sub>[masc][nominative]</sub> the-house<sub>[masc][genitive]</sub> the-big<sub>[masc][nominative]</sub>’ where the modifier ‘the-big’ is modifying ‘man’. For it to be ‘the big man of the house’, the case marking should be رَجُلُ الْبَيْتِ الْكَبِيرِ رَجُلُ الْبَيْتِ الْكَبِيرِ *rjlu Albyti Alkbyri* ‘man<sub>[masc][nominative]</sub> the-house<sub>[masc][genitive]</sub> the-big<sub>[masc][genitive]</sub>’. In fact, the language allows for adjectives to function as nouns, the modifier ‘the big’ could be the beginning of a new noun phrase in context and could be accusative, hence out of the scope of the NP. For example in the sentence رَجُلُ الْبَيْتِ الْكَبِيرِ حَمْدُ عَبْدِ السَّلَامِ *rjl Albyt Alkbyr mHmd Ebd AlslAm* ‘the man of the house, the elder Mohamad AbdelSalam’, where *Alkbyr* is an honorific for *mHmd Ebd AlslAm*.

2. **Possessive constructs** *Idafa* (Arabic) and *smyxuwt* (Hebrew) constructs make indefinite nominals syntactically definite hence allowing for agreement, therefore if identified correctly, they lead to better scoping and effectively better boundary detection. For example, رَجُلُ الْبَيْتِ الْكَبِيرِ رَجُلُ الْبَيْتِ الْكَبِيرِ *rjlu Albyti Alkbyru* ‘man<sub>[masc][nominative][definite]</sub> the-house<sub>[masc][genitive]</sub> the-big<sub>[masc][nominative][definite]</sub>’

---

<sup>14</sup>The exception is that Arabic sound masculine plural allows for relative disambiguation since it distinguishes between nominative وَنْ and both accusative and genitive cases بَنْ.

- 3. Passivization** Passive constructions are hard to detect due to underspecified short vowels marking passivization inflection. The best automatic systems are at 68–72 % accuracy [25, 44]. For example, in the sentence قتل عمرو بسلاح قاتل *qtl 'mrw bslAH qAtl* ‘killed Amr with a deadly weapon’, could mean *qatal Emrwā<sub>[Accusative][ARG1]</sub> bslAHīK qAtliK* ‘he killed Amr with a deadly weapon’ or *qutil Emrwū<sub>[Nomative][ARG1]</sub> bslAHīK qAtliK* ‘Amr was killed by a deadly weapon’, or *qatal Emrwū<sub>[Nomative][ARG0]</sub> bslAHīK qAtliK* ‘Amr killed [someone] with a deadly weapon’. Identifying the difference between these three depends on the underspecified vowels.

Passive constructions differ from English in that they cannot have an explicit non-instrument underlying subject, hence only ARG1 and ARG2. ARG0 are not allowed. For example: [عمرُو] [قتل] [بسلاح] [قاتل] [qutil] [*Emrwū<sub>[ARG1]</sub>*] [*bslAHīK qAtliK<sub>[ARG2]</sub>*] ‘[Amr]<sub>[ARG1]</sub> was killed [by a deadly weapon]<sub>[ARG2]</sub>’. Arabic doesn’t allow for constructions such as [عمرُو] [قتل] [بسالمي] [qutil] [*Emrwū<sub>[ARG1]</sub>*] [*bslmY<sub>[ARG0]</sub>*] ‘[Amr]<sub>[ARG1]</sub> was killed [by SalmA]<sub>[ARG0]</sub>’, it will have to be expressed as [عمرُو] [قتل] [يد سالمي] [qutil] [*Emrwū<sub>[ARG1]</sub>*] [*byd slmY<sub>[ARG0]</sub>*] ‘[Amr]<sub>[ARG1]</sub> was killed [by SalmA’s hand]<sub>[ARG2]</sub>’.

- 4. Agreement patterns** Relative free word order combined by agreement patterns between Subject and Verb could be helpful when explicit yet confusing with absence of case and passive marker and pro-drop. VSO configurations only allow for gender agreement between Verb and Subject while SVO configurations allow for gender and number agreement between subject and verb.

Accordingly, these syntactic characteristics are coupled with the challenge of vowel underspecification to yield a significant impediment to efficient SRL for Semitic languages. SRL requires that there exist quite robust tools for syntactic parsing and vowel restoration for such languages as precursors to SRL. Hypothetically, if such robust tools exist, the process of SRL would be rendered quite doable. In fact the following two studies show that using gold features with efficient modeling yields very comparable results to English SRL systems trained with gold features.

There are two pieces of work that developed SRL systems for Arabic. Both systems use supervised tree kernels and they experiment with gold parses. In the 2007 system [23, 29], the authors present a tree kernel SVM based system [71–74, 76] that uses standard features, without explicitly accounting for morphology. They used this system to participate in the SemEval 2007 Task 18 ‘Semantic Processing of Arabic’ where the data set is derived from the APB [27]. They report 94 % F-score on the boundary detection task and 81.43 % F-score on the argument classification task. In this study, the authors use a treebank, the ATB with all its syntactic trees. They use both syntactic and lexical features in modeling the problem. Similar to work in supervised SRL for English, the problem is broken down to two classification problems: argument boundary detection (ABD) and argument classification (AC). For both ABD and AC, the authors use the standard

features of: the *Predicate* which is in the lemma form, the *Path* which is the syntactic path linking the predicate and an argument, e.g. NN?NP?VP?VBX, the *Partial path* Path feature limited to the branching of the argument, *No-direction path* Like Path but without traversal directions, *Phrase type* Syntactic type of the argument node, *Position* Relative position of the argument with respect to the predicate, *Verb subcategorization* Production rule expanding the predicate parent node, *Syntactic Frame* Position of the NPs surrounding the predicate, and finally, *First and last word/POS* First and last words and POS tags of candidate argument phrases.

The features are extracted from the syntactically annotated treebank. The training occurs using a polynomial kernel SVM which represent these feature vectors. During the learning phase, given instances of two predicates of interest, the algorithm converts the predicate instance (the syntactic tree with all its relevant features) into a vector of  $n$  dimensions where the dimensions are the afore-defined features. The content of the cells corresponding to the feature dimensions is set to 0 or 1 depending whether it is present or not. The kernel function calculates the degree of overlap between the two vectors for the two predicate instances using cosine similarity. At decode time (tagging process), given a predicate instance, the algorithm converts it into the vector representation and attempts to find the most similar representation it has learned for the argument distribution for that type of predicate (namely the best fitting semantic frame). The algorithm is applied on an argument type by argument type basis. There are 26 argument roles defined in the APB. More details of the algorithm and its application can be found in [30].

Later in 2008 [30], the same authors extended the model to account for the rich morphological nature of the language by adding explicit leaves to the syntactic trees in the treebank. The feature vectors now also included tree features that go beyond the standard features mentioned above. The morphological information is derived from gold morphological information from the treebanks. The innovation in this work was extending the representation power of the trees allowing for an elegant incorporation of the explicit morphology of the predicates and the arguments into the tree representation. In the previous work, they operate on the syntactic trees without explicitly modeling the morphology; in this study, they extend the leaf nodes from being simple lexical items to include morphological features hence extending the tree representations. Then the modeling is similar to the previous study but with the new extended tree representations that explicitly model for the morphology. Therefore the vector dimensions included the standard features, the partial syntactic trees similar to the Path features without the morphology, and the Path feature with the morphology. They refer to these different feature representation settings as P3 (polynomial degree 3),<sup>15</sup> AST, and EAST. They report significant improvements over their previous work with 97 % for boundary detection (compared to 94 %) and 82.5 % for AC assuming gold ABD (compared to 81.43 %) and 82.17 % F-score (compared to 80.43 %) for overall automatic ABD and AC. It is worth noting that their version of the system that does not model the morphology explicitly yields

---

<sup>15</sup>The authors experiment with P1-P6 and find that P3 yields the best performance.

more modest results, i.e. P3+AST which models only the syntactic structures of the trees explicitly without any explicit morphological modeling, 80.8 % F-score for automatic ABD and AC. This illustrates the extreme relevance of taking the morphology seriously into account for morphologically rich languages.

Compared to English, it is noted in both studies that, similar to what we observe in English SRL, ARG0 shows a high F-score (96.70 %). Conversely, ARG1 seems more difficult to classify in Arabic. The F1 for ARG1 is only 90.57 % compared with 96.70 % for ARG0. This is consistent with our observation that word order poses a challenge for Arabic processing especially where the information about grammatical case is masked in the underspecified diacritic markers.

## 4.7 Conclusion

In this chapter, we presented key issues in meaning representation and processing of Semitic languages, focusing on lexical semantics. We pointed out the unique morphological and syntactic characteristics of Semitic languages, especially in comparison to English, how these differences may affect handling Semitic paradigmatic and syntagmatic semantics tasks, and current challenges in this area. We introduced the reader to central tasks: semantic distance measures, paraphrase detection and generation, word sense disambiguation and induction (WSD/WSI), multi-word expressions (MWE), and predicate argument analysis and semantic role labeling (SRL). Last, we presented examples of well-known algorithms for these tasks.

Most of current technology has been historically – and still is to a large extent – designed with English in mind. However, we already see a shift in this trend, with much work that is Semitic-centric, and more generally, centered on morphologically rich languages (see for example publications in the \*SEM<sup>16</sup> conference as well as the recent 2012 ACL SP-SEM-MRL workshop).<sup>17</sup> We see this shift as a mere beginning, as we believe that there is still much room to do more Semitic-centric (and MRL-centric) semantic research, which in turn, may shed more light on general semantic issues and properties in all human languages.

## References

1. Abend O, Reichart R, Rappoport A (2009) Unsupervised argument identification for semantic role labeling. In: Proceedings of the joint conference of the 47th annual meeting of the ACL and the 4th international joint conference on natural language processing of the AFNLP, Singapore. Association for Computational Linguistics, Suntec, pp 28–36. <http://www.aclweb.org/anthology/P/P09/P09-1004>

---

<sup>16</sup><http://ixa2.si.ehu.es/starsem/>

<sup>17</sup><https://sites.google.com/site/spsemmrl2012/>

2. Agirre E, Lopez de Lacalle Lekuona O (2003) Clustering WordNet word senses. In: Proceedings of the 1st international conference on recent advances in natural language processing (RANLP-2003), Borovets
3. Al-Haj H, Wintner S (2010) Identifying multi-word expressions by leveraging morphological and syntactic idiosyncrasy. In: Proceedings of the 23rd international conference on computational linguistics (Coling 2010), Coling 2010 Organizing Committee, Beijing, pp 10–18. <http://www.aclweb.org/anthology/C10-1002>
4. Androutsopoulos I, Malakasiotis P (2010) A survey of paraphrasing and textual entailment methods. *J Artif Intell Res (JAIR)* 38:135–187
5. Attia M, Toral A, Tounsi L, Pecina P, van Genabith J (2010) Automatic extraction of Arabic multiword expressions. In: Proceedings of the 2010 workshop on multiword expressions: from theory to applications, Coling 2010 Organizing Committee, Beijing, pp 19–27. <http://www.aclweb.org/anthology/W10-3704>
6. Baker CF, Fillmore CJ, Lowe JB (1998) The berkeley FrameNet project. In: COLING-ACL '98: proceedings of the conference, University of Montréal, pp 86–90
7. Banerjee S, Pedersen T (2003) Extended gloss overlaps as a measure of semantic relatedness. In: Proceedings of the eighteenth international joint conference on artificial intelligence (IJCAI-03), Acapulco, pp 805–810
8. Banko M, Cafarella MJ, Soderl S, Broadhead M, Etzioni O (2007) Open information extraction from the web. In: IJCAI, Hyderabad, pp 2670–2676
9. Bannard C, Callison-Burch C (2005) Paraphrasing with bilingual parallel corpora. In: Proceedings of the 43rd annual meeting of the Association for Computational Linguistics (ACL2005), Ann Arbor. Association for Computational Linguistics, pp 597–604
10. Carpuat M, Diab M (2010) Task-based evaluation of multiword expressions: a pilot study in statistical machine translation. In: Human language technologies: the 2010 annual conference of the North American chapter of the Association for Computational Linguistics, Los Angeles. Association for Computational Linguistics, pp 242–245. <http://www.aclweb.org/anthology/N10-1029>
11. Carreras X, Màrquez L (2005) Introduction to the CoNLL-2005 shared task: semantic role labeling. In: Proceedings of the ninth conference on computational natural language learning (CoNLL-2005), Ann Arbor. Association for Computational Linguistics, pp 152–164. <http://www.aclweb.org/anthology/W/W05/W05-0620>
12. Chen J, Rambow O (2003) Use of deep linguistic features for the recognition and labeling of semantic arguments. In: Proceedings of the 2003 conference on empirical methods in natural language processing, Sapporo
13. Cruse DA (1986) Lexical semantics. Cambridge University Press, Cambridge
14. Culò O, Erk K, Pado S, Schulte im Walde S (2008) Comparing and combining semantic verb classifications. *Lang Resour Eval* 42(3):265–291. doi:10.1007/s10579-008-9070-z
15. Dagan I, Itai A (1994) Word sense disambiguation using a second language monolingual corpus. *Comput Linguist* 20. <http://aclweb.org/anthology-new/J/J94/J94-4003.pdf>
16. Dagan I, Lee L, Pereira F (1999) Similarity-based models of cooccurrence probabilities. *Mach Learn* 34(1–3):43–69
17. Das D, Smith NA (2011) Semi-supervised frame-semantic parsing for unknown predicates. In: Proceedings of the 49th annual meeting of the Association for Computational Linguistics: human language technologies, Portland. Association for Computational Linguistics, pp 1435–1444. <http://www.aclweb.org/anthology/P11-1144>
18. Diab M (2003) Word sense disambiguation within a multilingual framework. PhD thesis, University of Maryland, College Park
19. Diab MT (2004) An unsupervised approach for bootstrapping Arabic sense tagging. In: Farghaly A, Megerdoomian K (eds) COLING 2004 computational approaches to Arabic script-based languages, COLING, Geneva, pp 43–50
20. Diab M, Bhutada P (2009) Verb noun construction mwe token classification. In: Proceedings of the workshop on multiword expressions: identification, interpretation, disambiguation and applications, Suntec. Association for Computational Linguistics, pp 17–22. <http://www.aclweb.org/anthology/W/W09/W09-2903>

21. Diab M, Krishna M (2009) Handling sparsity for verb noun MWE token classification. In: Proceedings of the workshop on geometrical models of natural language semantics, Athens. Association for Computational Linguistics, pp 96–103. <http://www.aclweb.org/anthology/W09-0213>
22. Diab M, Krishna M (2009) Unsupervised classification for vnc multiword expressions tokens. In: CICLING, Mexico City
23. Diab M, Moschitti A (2007) Semantic parsing for Modern Standard Arabic. In: Proceedings of recent advances in natural language processing (RANLP), Borovets
24. Diab M, Resnik P (2002) An unsupervised method for word sense tagging using parallel corpora. In: Proceedings of 40th annual meeting of the Association for Computational Linguistics, Philadelphia. Association for Computational Linguistics, pp 255–262, doi:10.3115/1073083.1073126. <http://www.aclweb.org/anthology/P02-1033>
25. Diab M, Hacioglu K, Jurafsky D (2004) Automatic tagging of Arabic text: from raw text to base phrase chunks. In: Susan Dumais DM, Roukos S (eds) HLT-NAACL 2004: Short papers, Boston. Association for Computational Linguistics, pp 149–152
26. Diab M, Alkhaila M, ElKateb S, Fellbaum C, Mansouri A, Palmer M (2007) SemEval-2007 task 18: Arabic semantic labeling. In: Proceedings of the fourth international workshop on semantic evaluations (SemEval-2007), Association for Computational Linguistics, Prague, pp 93–98. <http://www.aclweb.org/anthology/S/S07/S07-1017>
27. Diab M, Alkhaila M, ElKateb S, Fellbaum C, Mansouri A, Palmer M (2007) SemEval-2007 task 18: Arabic semantic labeling. In: Proceedings of the fourth international workshop on semantic evaluations (SemEval-2007), Prague. Association for Computational Linguistics, pp 93–98. <http://www.aclweb.org/anthology/W/W07/W07-2017>
28. Diab M, Ghoneim M, Habash N (2007) Arabic diacritization in the context of statistical machine translation. In: Proceedings of machine translation summit (MT-Summit), Copenhagen
29. Diab M, Moschitti A, Pighin D (2007) Cunit: a semantic role labeling system for Modern Standard Arabic. In: Proceedings of the fourth international workshop on semantic evaluations (SemEval-2007), Prague. Association for computational linguistics, pp 133–136. <http://www.aclweb.org/anthology/W/W07/W07-2026>
30. Diab M, Moschitti A, Pighin D (2008) Semantic role labeling systems for Arabic using kernel methods. In: Proceedings of ACL-08: HLT, Columbus. Association for Computational Linguistics, pp 798–806. <http://www.aclweb.org/anthology/P/P08/P08-1091>
31. Dunning T (1993) Accurate methods for the statistics of surprise and coincidence. *Comput Linguist* 19(1):61–74
32. Elkateb S, Black W, Rodriguez H, Alkhaila M, Vossen P, Pease A, Fellbaum C (2006) Building a wordnet for Arabic. In: Proceedings of the fifth international conference on language resources and evaluation, LREC, Genoa
33. Elmougy S, Hamza T, Noaman HM (2008) Naive Bayes classifier for Arabic word sense disambiguation. In: Proceedings of INFOS 2008, Cairo, pp 27–29
34. Erk K, Pado S (2006) Shalmaneser – a toolchain for shallow semantic parsing. In: Proceedings of the language resources and evaluation conference (LREC), Genoa
35. Fellbaum C (1998) Wordnet: an electronic lexical database. MIT, Cambridge
36. Firth JR (1957) A synopsis of linguistic theory 1930–1955. Studies in linguistic analysis (special volume of the Philological Society). Blackwell, Oxford, pp 1–32
37. Frege G (1892) Über Sinn und Bedeutung. *Zeitschrift für Philosophie und philosophische Kritik* 100:25–50
38. Fung P, Yee LY (1998) An IR approach for translating new words from nonparallel, comparable texts. In: Proceedings of coling – ACL, Montreal, pp 414–420
39. Giampiccolo D, Magnini B, Dagan I, Dolan B (2007) The third Pascal recognizing textual entailment challenge. In: Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing, Prague. Association for Computational Linguistics, pp 1–9. <http://www.aclweb.org/anthology/W/W07/W07-1401>

40. Gildea D, Jurafsky D (2002) Automatic labeling of semantic roles. *Comput Linguist* 28(3):245–288. <http://www.cs.rochester.edu/~gildea/gildea-cl02.pdf>
41. Gildea D, Palmer M (2002) The necessity of parsing for predicate argument recognition. In: Proceedings of the 40th annual conference of the Association for Computational Linguistics (ACL-02), Philadelphia
42. Goldwasser D, Reichart R, Clarke J, Roth D (2011) Confidence driven unsupervised semantic parsing. In: Proceedings of the 49th annual meeting of the Association for Computational Linguistics: human language technologies, Portland. Association for Computational Linguistics, pp 1486–1495. <http://www.aclweb.org/anthology/P11-1149>
43. Habash N (2010) Introduction to Arabic natural language processing. Morgan & Claypool, San Rafael
44. Habash N, Rambow O (2005) Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In: Proceedings of the 43rd annual meeting of the Association for Computational Linguistics (ACL'05), Ann Arbor. Association for Computational Linguistics, pp 573–580. doi:10.3115/1219840.1219911, <http://www.aclweb.org/anthology/P05-1071>
45. Habash N, Rambow O (2007) Arabic diacritization through full morphological tagging. In: Human language technologies 2007: the conference of the North American chapter of the Association for Computational Linguistics; Companion Volume, Short Papers, Rochester. Association for Computational Linguistics, pp 53–56. <http://www.aclweb.org/anthology/N/N07/N07-2014>
46. Haghghi A, Toutanova K, Manning C (2005) A joint model for semantic role labeling. In: Proceedings of the ninth conference on computational natural language learning (CoNLL-2005), Ann Arbor. Association for Computational Linguistics, pp 173–176. <http://www.aclweb.org/anthology/W/W05/W05-0623>
47. Harris ZS (1940) Review of Louis H. Gray, foundations of language (Macmillan, New York 1939). *Language* 16(3):216–231
48. Hawwari A, Bar K, Diab M (2012) Building an Arabic multiword expressions repository. In: Proceedings of the ACL 2012 joint workshop on statistical parsing and semantic processing of morphologically rich languages, Jeju. Association for Computational Linguistics, pp 24–29. <http://www.aclweb.org/anthology/W12-3403>
49. Hirst G, Budanitsky A (2005) Correcting real-word spelling errors by restoring lexical cohesion. *Nat Lang Eng* 11(1):87–111
50. Jackendoff R (1983) Semantics and cognition. MIT, Cambridge
51. Jackendoff R (1990) Semantic structures. MIT, Cambridge
52. Kingsbury P, Palmer M (2003) Propbank: the next level of treebank. In: Proceedings of treebanks and lexical theories, Växjö
53. Kipper K, Korhonen A, Ryant N, Palmer M (2006) Extending VerbNet with novel verb classes. In: Proceedings of the sixth international conference on language resources and evaluation, Genoa
54. Lang J, Lapata M (2011) Unsupervised semantic role induction via split-merge clustering. In: Proceedings of the 49th annual meeting of the Association for Computational Linguistics: human language technologies, Portland. Association for Computational Linguistics, pp 1117–1126. <http://www.aclweb.org/anthology/P11-1112>
55. Lee JH, Kim MH, Lee YJ (1993) Information retrieval based on conceptual distance in IS-A hierarchies. *J Doc* 49(2):188–207
56. Lenci A, McGillivray B, Montemagni S, Pirrelli V (2008) Unsupervised acquisition of verb subcategorization frames from shallow-parsed corpora. In: Proceedings of the sixth international conference on language resources and evaluation (LREC'08), Marrakech. <http://www.lrec-conf.org/proceedings/lrec2008/>
57. Lesk M (1986) Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In: Proceedings of the 5th annual international conference on systems documentation, Toronto, pp 24–26

58. Levin B (1993) English verb classes and alternations: a preliminary investigation. University Of Chicago Press, chicago
59. Levinson D (1999) Corpus-based method for unsupervised word sense disambiguation. In: Proceedings of the workshop on machine learning in human language technology, advanced course on artificial intelligence (ACAI'99), Chania, pp 267–273
60. Li J, Zhou G, Ng HT (2010) Joint syntactic and semantic parsing of Chinese. In: Proceedings of the 48th annual meeting of the Association for Computational Linguistics, Uppsala. Association for Computational Linguistics, pp 1108–1117. <http://www.aclweb.org/anthology/P10-1113>
61. Lin D (1998) An information-theoretic definition of similarity. In: Proceedings of the 15th international conference on machine learning, San Francisco, pp 296–304
62. Lüdeling A, Kytö M (eds) (2009) Corpus linguistics. An international handbook. Handbooks of linguistics and communication science, vol 2. Mouton de Gruyter, Berlin
63. Maamouri M, Bies A, Buckwalter T, Diab M, Habash N, Rambow O, Tabassi D (2006) Developing and using a pilot dialectal Arabic treebank. In: Fifth international conference on language resources and evaluation (LREC2006), Genoa
64. Maamouri M, Bies A, Kulick S (2008) Enhanced annotation and parsing of the Arabic treebank. In: INFOS, Cairo
65. Madnani N, Dorr B (2010) Generating phrasal and sentential paraphrases: a survey of data-driven methods. *Comput Linguist* 36(3):341–387
66. Marton Y (2010) Improved statistical machine translation using monolingual text and a shallow lexical resource for hybrid phrasal paraphrase generation. In: Proceedings of the AMTA, Denver
67. Marton Y, Callison-Burch C, Resnik P (2009) Improved statistical machine translation using monolingually-derived paraphrases. In: Proceedings of EMNLP, Singapore
68. Marton Y, Mohammad S, Resnik P (2009) Estimating semantic distance using soft semantic constraints in knowledge-source/corpus hybrid models. In: Proceedings of EMNLP, Singapore
69. Merhbene L, Zouaghi A, Zrigui M (2010) Ambiguous Arabic words disambiguation. In: SNPD, London, pp 157–164
70. Mohammad S, Hirst G (2006) Distributional measures of concept-distance: a task-oriented evaluation. In: Proceedings of EMNLP, Sydney
71. Moschitti A (2004) A study on convolution kernels for shallow semantic parsing. In: Proceedings of the 42nd conference on Association for Computational Linguistic (ACL-2004), Barcelona
72. Moschitti A (2006) Efficient convolution kernels for dependency and constituent syntactic trees. In: ECML'06, Berlin
73. Moschitti A (2006) Making tree kernels practical for natural language learning. In: Proceedings of 11th conference of the European chapter of the Association for Computational Linguistics (EACL2006), Trento, pp 113–120
74. Moschitti A (2008) Kernel methods, syntax and semantics for relational text categorization. In: Proceedings of the 17th ACM conference on information and knowledge management (CIKM), Napa Valley. ACM
75. Moschitti A, Quarteroni S, Basili R, Manandhar S (2007) Exploiting syntactic and shallow semantic kernels for question answer classification. In: Proceedings of the 45th annual meeting of the Association of Computational Linguistics, Prague. Association for Computational Linguistics, pp 776–783. <http://www.aclweb.org/anthology/P/P07/P07-0098>
76. Moschitti A, Pighin D, Basili R (2008) Tree kernels for semantic role labeling. *Comput Linguist* 34(2):193–224. doi:10.1162/coli.2008.34.2.193, <http://www.mitpressjournals.org/doi/abs/10.1162/coli.2008.34.2.193>, <http://www.mitpressjournals.org/doi/pdf/10.1162/coli.2008.34.2.193>
77. Mousser J (2010) A large coverage verb taxonomy for Arabic. In: Proceedings of the seventh conference on international language resources and evaluation (LREC'10), Valletta

78. Mousser J (2011) Classifying Arabic verbs using sibling classes. In: Proceedings of the ninth international conference on computational semantics (IWCS 2011), Oxford
79. Navigli R (2006) Meaningful clustering of senses helps boost word sense disambiguation performance. In: Proceedings of the 21st international conference on computational linguistics and the 44th annual meeting of the association, Sydney, pp 105–112
80. Navigli R (2009) Word sense disambiguation: a survey. *ACM Comput Surv* 41(2):10:1–10:69. doi:10.1145/1459352.1459355, <http://doi.acm.org/10.1145/1459352.1459355>
81. Nelken R, Shieber SM (2005) Arabic diacritization using weighted finite-state transducers. In: Proceedings of the ACL workshop on computational approaches to Semitic languages, Ann Arbor. Association for Computational Linguistics, pp 79–86. <http://www.aclweb.org/anthology/W/W05/W05-0711>
82. Niles I, Pease A (2001) Origins of the IEEE standard upper ontology. In: Working notes of the IJCAI-2001 workshop on the IEEE standard upper ontology, Seattle
83. Ordan N, Wintner S (2007) Hebrew wordnet: a test case of aligning lexical databases across languages. *Int J Trans Spec Issue Lex Resour Mach Trans* 19(1):39–58
84. Pantel P, Lin D (2002) Discovering word senses from text. In: Proceedings of ACM conference on knowledge discovery and data mining (KDD-02), ACM, Edmonton, pp 613–619. <http://www.patrickpantel.com/download/papers/2002/kdd02.pdf>
85. Pasca M, Dienes P (2005) Aligning needles in a haystack: paraphrase acquisition across the web. In: Proceedings of IJCNLP, Jeju Island, pp 119–130
86. Patwardhan S, Pedersen T (2006) Using WordNet based context vectors to estimate the semantic relatedness of concepts. In: Proceedings of making sense of sense EACL workshop, Trento, pp 1–8
87. Poon H, Domingos P (2009) Unsupervised semantic parsing. In: Proceedings of the 2009 conference on empirical methods in natural language processing, Stroudsburg, EMNLP '09, vol 1. Association for Computational Linguistics, pp 1–10. <http://dl.acm.org/citation.cfm?id=1699510.1699512>
88. Pradhan S, Hacioglu K, Ward W, Martin JH, Jurafsky D (2003) Semantic role parsing: adding semantic structure to unstructured text. In: Proceedings of the international conference on data mining (ICDM-2003), Melbourne
89. Rada R, Mili H, Bicknell E, Blettner M (1989) Development and application of a metric on semantic nets. *IEEE Trans Syst Man Cybern* 19(1):17–30
90. Rapp R (1999) Automatic identification of word translations from unrelated English and German corpora. In: Proceedings of the 37th annual conference of the Association for Computational Linguistics, College Park, pp 519–525
91. Resnik P (1999) Semantic similarity in a taxonomy: an information-based measure and its application to problems of ambiguity in natural language. *J Artif Intell Res (JAIR)* 11:95–130
92. Rodriguez H, Farwell D, Farreres B, Bertran M, Alkhalifa M, Marti M, Black W, Elkateb S, Kirk J, Pease A, Vossen P, Fellbaum C (2008) Arabic wordnet: current state and future extensions. In: Proceedings of global WordNet conference, Szeged
93. Ross S (1976) A first course in probability. Macmillan, New York
94. Roth R, Rambow O, Habash N, Diab M, Rudin C (2008) Arabic morphological tagging, diacritization, and lemmatization using lexeme models and feature ranking. In: Proceedings of ACL-08: HLT, Short Papers, Columbus. Association for Computational Linguistics, pp 117–120. <http://www.aclweb.org/anthology/P/P08/P08-2030>
95. Rozovskaya A, Sproat R (2007) Multilingual word sense discrimination: a comparative cross-linguistic study. In: Proceedings of the workshop on balto-slavonic natural language processing, Prague. Association for Computational Linguistics, pp 82–87. <http://www.aclweb.org/anthology/W/W07/W07-1711>
96. Sag IA, Baldwin T, Bond F, Copestate AA, Flickinger D (2002) Multiword expressions: a pain in the neck for nlp. In: Proceedings of the third international conference on computational linguistics and intelligent text processing, Mexico City. Springer, London, pp 1–15
97. Salton G, McGill MJ (1983) Introduction to modern information retrieval. McGraw-Hill, New York

98. Schuetze H (1998) Automatic word sense discrimination. *Comput Linguist* 24(1):97–123
99. Schuetze H, Pedersen JO (1997) A cooccurrence-based thesaurus and two applications to information retrieval. *Inf Process Manag* 33(3):307–318
100. Schulte im Walde S (2000) Clustering verbs semantically according to their alternation behaviour. In: Proceedings of the 18th international conference on computational linguistics (COLING-00), Saarbrücken, pp 747–753
101. Schulte im Walde S (2009) the induction of verb frames and verb classes from corpora. In: Lüdeling A, Kytö M (eds) An international handbook. Handbooks of linguistics and communication science, vol 2. Mouton de Gruyter, Berlin, chap 44, pp 952–971
102. Snider N, Diab M (2006) Unsupervised induction of Modern Standard Arabic verb classes. In: Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers, New York. Association for Computational Linguistics, pp 153–156. <http://www.aclweb.org/anthology/N/N06/N06-2039>
103. Snider N, Diab M (2006) Unsupervised induction of Modern Standard Arabic verb classes using syntactic frames and lsa. In: Proceedings of the COLING/ACL 2006 main conference poster sessions, Sydney. Association for Computational Linguistics, pp 795–802. <http://www.aclweb.org/anthology/P/P06/P06-2102>
104. Sun H, Jurafsky D (2004) Shallow semantic parsing of Chinese. In: Susan Dumais DM, Roukos S (eds) HLT-NAACL 2004: main proceedings, Boston. Association for Computational Linguistics, pp 249–256
105. Sun W (2010) Semantics-driven shallow parsing for Chinese semantic role labeling. In: Proceedings of the ACL 2010 conference short papers, Uppsala. Association for Computational Linguistics, pp 103–108. <http://www.aclweb.org/anthology/P10-2019>
106. Thompson CA, Levy R, Manning C (2003) A generative model for semantic role labeling. In: 14th European conference on machine learning, Cavtat-Dubrovnik
107. Titov I, Klementev A (2011) A bayesian model for unsupervised semantic parsing. In: Proceedings of the 49th annual meeting of the Association for Computational Linguistics: human language technologies, Portland. Association for Computational Linguistics, pp 1445–1455. <http://www.aclweb.org/anthology/P11-1145>
108. Tsvetkov Y, Wintner S (2010) Extraction of multi-word expressions from small parallel corpora. In: Coling 2010: posters, COLING 2010 Organizing Committee, Beijing, pp 1256–1264. <http://www.aclweb.org/anthology/C10-2144>
109. Turney PD, Pantel P (2010) From frequency to meaning: vector space models of semantics. *J Artif Intell Res* 37:141–188
110. Vergyri D, Kirchhoff K (2004) Automatic diacritization of Arabic for acoustic modeling in speech recognition. In: Farghaly A, Megerdoomian K (eds) COLING 2004 computational approaches to arabic script-based languages, Geneva. COLING, pp 66–73
111. Weaver W (1949) Translation. In: Locke W, Booth A (eds) Machine translation of languages: fourteen essays. MIT, Cambridge
112. Wu F, Weld DS (2010) Open information extraction using wikipedia. In: Proceedings of the 48th annual meeting of the Association for Computational Linguistics, ACL '10, Stroudsburg. Association for Computational Linguistics, pp 118–127. <http://dl.acm.org/citation.cfm?id=1858681.1858694>
113. Xue N, Palmer M (2004) Calibrating features for semantic role labeling. In: Lin D, Wu D (eds) Proceedings of EMNLP 2004, Barcelona. Association for Computational Linguistics, pp 88–94
114. Zaghouani W, Diab M, Mansouri A, Pradhan S, Palmer M (2010) The revised Arabic propbank. In: Proceedings of the fourth linguistic annotation workshop, Uppsala. Association for Computational Linguistics, pp 222–226. <http://www.aclweb.org/anthology/W10-1836>
115. Zbib R, Matsoukas S, Schwartz R, Makhoul J (2010) Decision trees for lexical smoothing in statistical machine translation. In: Proceedings of the joint fifth workshop on statistical machine translation and MetricsMATR, Uppsala. Association for Computational Linguistics, pp 428–437. <http://www.aclweb.org/anthology/W10-1763>

116. Zitouni I, Sorensen JS, Sarikaya R (2006) Maximum entropy based restoration of Arabic diacritics. In: Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the Association for Computational Linguistics, Sydney. Association for Computational Linguistics, pp 577–584. doi:10.3115/1220175.1220248. <http://www.aclweb.org/anthology/P06-1073>
117. Zouaghi A, Merhbene L, Zrigui M (2010) Combination of information retrieval methods with lesk algorithm for arabic word sense disambiguation. Artif Intell Rev 1–13. <http://dx.doi.org/10.1007/s10462-011-9249-3>

# Chapter 5

## Language Modeling

Ilana Heintz

### 5.1 Introduction

The goal of language modeling is to accurately predict the next word in a sequence of words produced in a natural language. The history, or context, that is used to make that prediction might be long or short, knowledge-rich or knowledge-poor. We may base a prediction only on a single preceding word, or potentially using knowledge of all words from the start of the passage preceding the word in question. Knowledge-rich models can incorporate information about morphology, syntax or semantics to inform the prediction of the next word, whereas knowledge-poor models will rely solely on the words as they appear in the text, without pre-processing or normalization of any kind. This chapter describes knowledge-rich and knowledge-poor language models and how each might be useful in modeling Semitic languages.

Every language modeling technique and application must handle the issue of data sparsity: with limited amounts of data available on which to train a model, many parameters will be poorly estimated. In a model that estimates probabilities for two-word sequences (bigrams), it is unclear whether a given bigram has a count of zero because it is never a valid sequence in the language, or if it was only by chance not included in the subset of language expressed in the training data. As the length of the modeled sequences grows more complex, this sparsity issue also grows. Of all possible combinations of 4-grams in a language, very few are likely to appear at all in a given text, and even fewer will repeat often enough to provide reliable frequency statistics. The same is true of additional information (e.g., morphological or syntactic tags) added to each term in the sequence; the probability of encountering a given feature combination decreases as the complexity of the features increases. Therefore, if the goal of language modeling is to predict the

---

I. Heintz (✉)

Speech, Language and Multimedia, Raytheon BBN Technologies, Cambridge, MA, USA  
e-mail: [iheintz@bbn.com](mailto:iheintz@bbn.com)

next word, the challenge is to find appropriate, reliable estimates of word sequence probabilities to enable the prediction. Approaches to this challenge are three-fold: smoothing techniques are used to offset zero-probability sequences and spread probability mass across a model; enhanced modeling techniques that incorporate machine learning or complex algorithms are used to create models that can best incorporate additional linguistic information; and particularly for Semitic language modeling, morphological information is extracted and provided to the models in place of or in addition to lexical information.

The data sparsity issue is particularly intense for Semitic languages due to their morphological richness, as is described earlier in this volume. The copious use of affixational morphology results in a great increase in vocabulary size. For example, the English present-tense verb “run” has the correct agreement as the first- and second-person singular, and first-, second-, and third-person plural forms. The same word form is also a singular noun. In the Semitic languages, different affixes would be used to create varying word forms indicating the correct person, number, and gender information, and the root-and-pattern morphology would be employed to derive a different form for the noun, as well as different stems to account for changes in tense or aspect. In order to build reliable language models, this increase in vocabulary size must be countered by collecting more training data, by applying morphological stemming techniques, or by employing sophisticated modeling techniques that better handle sparse data.

Language models are used in a variety of natural language processing applications. In automatic speech recognition, language models can be used to help choose the best sequence of words from an acoustic lattice. Similarly in machine translation, language models help to indicate the best path through a lattice output by a translation model. Natural-sounding phrases and sentences are produced by natural language generation with the aid of language models, as well. Discussions of Semitic language modeling occur predominantly in the automatic speech recognition literature, with some mention also in studies on statistical machine translation. These studies will be cited throughout this chapter.

This chapter proceeds by describing the utility of the perplexity evaluation of language models in Sect. 5.2. This is followed by an introduction to n-gram language modeling in Sect. 5.3, and an explanation of smoothing methods in Sect. 5.4. Understanding smoothing is necessary to understanding variations on n-gram language models that are described in Sect. 5.5. Section 5.6 comprises a review of how morphological information is incorporated into a number of Semitic language models. Section 5.7 summarizes and concludes the chapter with suggestions for the most useful approaches to Semitic language modeling.

## 5.2 Evaluating Language Models with Perplexity

Outside of any particular application, we can compare language models by comparing the likelihood that they assign to a previously unseen sequence of words. For a given language model (LM), we might wish to calculate:

$$P_{LM}(w_1 \dots w_n) \quad (5.1)$$

This probability is an element of the cross-entropy of the text:

$$H(P_{LM}) = -\frac{1}{n} \log P_{LM}(w_1 \dots w_n) \quad (5.2)$$

As discussed in [36] among many other texts, cross-entropy is a measurement of the amount of uncertainty in a probability distribution. We usually calculate the probability of a sequence of terms as the sum of the probability of each term given its preceding context:

$$H(P_{LM}) = -\frac{1}{n} \sum_{i=1}^n \log P_{LM}(w_i | w_1 \dots w_{i-1}) \quad (5.3)$$

It is customary to measure the *perplexity* of a test set given the language model, which gives the number of bits required to encode that text using the LM:

$$PP = 2^{H(P_{LM})} \quad (5.4)$$

A language model that encodes a test set most efficiently will do so with the *smallest* number of bits. Therefore, we optimize a language model by minimizing the perplexity of a given test set. Note that in modeling Semitic languages, the vocabulary of the train and test set is often changed to account for morphological properties of the language; in these cases, a variation on perplexity must be used to compare models fairly. Section 5.3 discusses these changes to the evaluation.

Equations (5.3) and (5.4) assume that we use the entire preceding context to calculate the probability of each term. Because that calculation is not feasible, we replace the  $P_{LM}(w_i | w_1 \dots w_{i-1})$  term with an approximation, such as the probability of only the previous  $N$  terms, or of the previous  $N$  terms and appropriate features.

Perplexity is an often-cited measure of the strength of a language model. It is a measure of that model's complexity, and serves to quantify the entropy of the model in relation to an unseen text. Perplexity is both useful and limited because it is a measurement of a model that is intended for use in a larger system. A language model is normally a means to an end: speech recognition (ASR), machine translation (MT), natural language generation (NLG), and other natural language processing tasks all use language models in one or more steps of processing. Building complete systems for these tasks is usually more difficult and time-consuming than building the language model alone. Even with a stable system, testing whether a new language model improves that system's results can take a long time and consume computational resources. In contrast, perplexity is simple to calculate and easy to compare cross-model. Language models can be directly compared by perplexity without building acoustic models, translation models, or other required

system components. However, the measurement tells us little if anything about how well the model will perform *in context*. Perplexity indicates how likely the correct word is to follow a sequence of known words, but that likelihood usually ignores features like acoustic confusability, semantic confusability, and syntactic confusability, which will be the main hurdles in the tasks of ASR, MT, and NLG respectively. Perplexity can overestimate the difficulty of choosing the right word in some contexts, while underestimating the difficulty in others. If a particular difficult context shows up frequently, then improvements in perplexity may turn out to be misleading.

Most of the Semitic NLP studies cited in this chapter apply language models to the speech recognition problem. The challenge considered in this application is acoustic confusability, which is amplified in Modern Standard Arabic (MSA) by the concatenative morphology of affixes. Two words that differ only in a short prefix or suffix will be acoustically confusable, and there are many such word pairs in MSA. A language model that incorporates syntactic or morphological information may give differing probabilities to these words, and the reduction of perplexity may indicate such differences. But, we do not know whether the acoustic confusability problem is overcome unless the use of the language model in an ASR task results in a reduction of word error rate.

In summary, due to its ease of calculation and relationship to the well-understood property of cross-entropy, perplexity evaluations are often cited as indicators of language modeling utility. In these cases, care must be taken to assure that models are directly comparable in terms of their vocabulary and the vocabulary of the test set. Incorporating the language models into a complete NLP system and evaluating *in situ*, when possible, is a truer indicator of the models' utility.

### 5.3 N-Gram Language Modeling

A very common, basic form of language modeling is *n-gram language modeling*. A probability is assigned to each unit of language based on its frequency in a training corpus. A “unit of language” refers to some sequence of tokens, usually words. The order  $n$  of the language model can be single words (unigrams), or sequences of two, three, four, or more words (bigrams, trigrams, 4-grams, etc.). The simplest example of an n-gram model is a maximum likelihood model over unigrams, where each word is assigned a probability based on its count in a training corpus:  $P(w) = \frac{\text{count}(w)}{\sum_w \text{count}(w)}$ . A maximum likelihood bigram model takes into account the previous word:  $P(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}w_i)}{\sum_w \text{count}(w_{i-1}w)}$ . In theory, increased context leads to greater accuracy. A perfect theoretical model would base the prediction of each word on all of the preceding context:  $P(w_i | w_1 \dots w_{i-1})$ . In practice, long n-grams do not repeat often enough to provide reliable statistics, so we estimate an imperfect n-gram model by restricting the context to only a few preceding words.

Calculating the perplexity of a test text given a bigram model only requires the specification of Eq. (5.4) to the bigram case:

$$PP_{bigram}(w_1 \dots w_n) = 2^{-\frac{1}{n} \sum_{i=2}^n \log P(w_i | w_{i-1})} \quad (5.5)$$

Similar formulas are derived for models with larger-order n-grams.

In using n-gram language modeling for Semitic languages, the definition of what constitutes a word or token in the model must be carefully considered. Often, normalization and tokenization tools are applied to the text before the language model is calculated. Normalization refers to reducing orthographic variation, such as changing the various forms of Arabic hamza to a single form. Tokenization methods such as stemming are used to reduce the size of the vocabulary by separating clitics and morphological affixes from stems. Reducing the size of the vocabulary through normalization and tokenization has the effect of increasing the frequency of each token, which in turn increases the reliability of the model. Normalization and tokenization also reduce the number of out-of-vocabulary tokens - those tokens that appear in the test text but not in the training text, and therefore may receive a zero probability. If a language model uses tokens that are smaller than words, then a post-processing step must be performed after decoding to return the tokens to their original state. This is called de-tokenization. El Kholy and Habash [20] and Al-Haj and Lavie [3] are recent studies that compare various types of normalization, tokenization, and the reverse processes in an English-Arabic machine translation application. Both studies find that a coarse level of tokenization, such as using surface-level segmentation as opposed to a fine-grained morphological segmentation, is sufficient to create useful language models, and also results in more accurate de-tokenization.

These studies and others cited below change the vocabulary of the language model by using morphemes or multiple words as the modeling units. Changing the vocabulary in this way results in a situation where comparing perplexity values is not mathematically sound; the conditions may have changed in such a way as to bias the calculation, and the perplexity results are no longer fully interpretable. Kirchhoff et al. [33] introduces a variation on the perplexity calculation that overcomes this problem by using a consistent normalization factor across models:

$$ModPP_{bigram}(w_1 \dots w_n) = 2^{-\frac{1}{n} \sum_{i=2}^m \log P(w_i | w_{i-1})} \quad (5.6)$$

In this formulation, the normalization factor  $\frac{1}{n}$  always counts the number of *words* in the test set. The number of tokens  $m$  may change from model to model based on tokenization methods, but the number of words  $n$  remains the same for a given test text. The average negative log probability of the test text is calculated by removing the exponent, as in [63] and [27]. These variations on the perplexity formula allow for a more reliable comparison of models that use differing vocabularies.

## 5.4 Smoothing: Discounting, Backoff, and Interpolation

Section 5.3 refers to the use of normalization and tokenization to counter the problem of out-of-vocabulary (OOV) terms, and the use of smaller contexts to increase the reliability of n-gram counts. This section discusses mathematical approaches to solving the problems of OOV terms and unreliable counts. These concepts are a necessary precursor to understanding the more complex types of LMs discussed in Sect. 5.5.

Smoothing refers to three related concepts: discounting, backoff, and interpolation. All three will be discussed in this section, which is modeled on the more complete discussion in [14]. **Discounting** refers to the movement of probability mass from frequent, well-modeled parameters to infrequent parameters with less evidence. The probabilities of all parameters in an LM must always sum to one, therefore discounting also implies normalization. **Backoff** is the term used to describe the use of lower-order n-gram probabilities when the evidence for higher-order n-grams is lacking. Higher-order (trigram, 4-gram) n-grams are preferred because they provide greater context for the predicted word, resulting in more accurate predictions. But this is only true when the higher-order n-grams exist in training text, and usually only if they exist with repetition. Because lower-order (unigram, bigram) parameters are more likely to repeat in a text, their estimates are more reliable, even if they provide less context. Backoff is used to counter the greater context of the high-order n-grams with the greater reliability of the low-order n-grams. Backoff is often done in a step-wise fashion: we use a lower-order probability where the higher-order probability is zero. **Interpolation** also takes into account higher-order and lower-order parameters, but in doing so creates a smoother distribution of probability mass than simple backoff. With interpolation, all of the higher-order n-gram probabilities are tempered by lower-order n-gram probabilities, regardless of their frequency.

### 5.4.1 Discounting

The simplest formulation for determining the probability of each n-gram requires no discounting; it is given by the maximum likelihood (ML) model:

$$P_{ML}(w_i | w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^i)}{\sum_{w_i} c(w_{i-n+1}^i)} \quad (5.7)$$

Such an estimate results in zero-value probabilities for any unseen n-grams, an undesirable outcome, given that we always expect new words to occur in a text different from that used to build the model.

This outcome can be avoided by simply adding 1, or another constant, to the count of every n-gram. In a test condition, the unseen n-gram will be assigned

exactly this constant as its count. This has been empirically shown to be ineffective [22]. A more sophisticated technique is Good-Turing discounting, which involves calculating the count-of-counts: for every count  $r$ , how many n-grams appear  $r$  times? We calculate  $r^*$ , the Good-Turing discounting factor, by smoothing the count-of-counts:

$$r^* = (r + 1) \frac{N_{r+1}}{N_r} \quad (5.8)$$

where  $r$  is the frequency of an n-gram type (its token count) and  $N_r$  is the number of n-grams (types) with that frequency. For an n-gram  $w_i \dots w_{i-n+1}$ , we count its occurrences  $r$ , then estimate its probability using its discounted count  $r^*$ :

$$P_{GT}(w_{i-n+1}^i) = \frac{r^*}{N} \quad (5.9)$$

The probability mass that is taken away via discounting is “given back” in the form of probability mass on unseen events: the total probability attributed to unseen objects is  $\frac{N_1}{N}$ , the maximum likelihood of all singleton n-grams [23].

To understand Eq. (5.8), imagine a corpus of  $N$  tokens. We remove one token from the corpus to create a single-token ‘held-out set’, then count the occurrences of that type in the remaining ‘training’ set. We find that token in the training corpus one fewer times than its actual occurrence in the full corpus. If we repeat the ‘held-out’ experiment for each token, we find that any token that appeared  $r$  times in the training set has a true count of  $r + 1$ . Now consider a held-out token that does not appear in the training set. From the perspective of the training set, it is an unknown, but we know that it in fact has a singleton count in the true corpus. This is why we attribute approximately a singleton count to all unknown tokens. We find the total count of all singletons:  $(0 + 1)N_1$ , then divide that count by the number of unknowns  $N_0$  (Eq. 5.8), assigning to each unknown token an equal portion of the total singleton count. We divide again by the token count of the training corpus to derive the appropriate probability mass for each unseen token (Eq. 5.9). The same logic is true when we consider the discounted value of singletons, twice-appearing types, and so on.

To calculate the Good-Turing discount, the training data must be diverse enough to preclude any zero counts  $r$ ; in particular, there must be tokens with low  $r$  counts, including  $r = 1$  (which is to say, we do not prune singleton tokens before estimating the Good-Turing probabilities). At the upper ranges of  $r$ , where zero values of  $N_r$  are likely to naturally occur, the values of  $N_r$  are interpolated using simple linear regression. This step removes the zero values, and the estimates reflect a trend of fewer items, rather than leveling off at  $N_r = 1$ . In general, this method of calculating  $r^*$  results in estimated counts that are slightly less than the actual count for low values of  $r$ , and close to the actual count for high values of  $r$ . Less probability mass is taken from high-frequency words (and occasionally probability mass is added to them via interpolation, due to the variability of  $N_r$  for high values of  $r$ ) because

they are more trustworthy. We are more skeptical of the repetition of low-frequency words, therefore we take more probability mass from them to apply to unseen words.

With this estimated value of  $r^*$ , Eq. (5.9) can be used to calculate the estimated probability of each item in the training data. This discounting method is often used to estimate n-gram probabilities in conjunction with one of the backoff or interpolation methods discussed next.

### 5.4.2 Combining Discounting with Backoff

A very common form of combining backoff with discounting is known as Katz backoff, introduced in [30]. In this algorithm, frequent n-grams are not discounted; they are trustworthy and their estimates are not changed. Low-frequency n-grams, already considered unreliable, lose some of their probability mass to unseen n-grams. This discounting is done with a variation of the Good-Turing discount factor. Furthermore, backoff weights are used to ensure that the total probability mass of the model remains equal to 1. Formally, where  $c(x)$  indicates the count of  $x$  and  $k$  is a frequency cutoff constant:

$$p_{Katz}(w_i \dots w_{i-n+1}) = \begin{cases} p_{ML}(w_{i-n+1}^i), & \text{if } c(w_{i-n+1}^i) > k \\ d_r(w_{i-n+1}^i), & \text{if } 0 < c(w_{i-n+1}^i) \leq k \\ \alpha(w_{i-n+1}^{i-1}) p_{ML}(w_{i-n+2}^i), & \text{if } c(w_{i-n+1}^i) = 0 \end{cases} \quad (5.10)$$

The discount factor  $d_r$  is a variation of the Good-Turing discount factor; the discount is modified so that the total probability mass of the model remains constant, despite the choice to *not* discount high-frequency n-grams. The backoff weight  $\alpha$  is related to the discounting factor; in the case that we use backoff,  $\alpha$  applies the probability mass gained from the discounting factor evenly across the n-grams that could result from the backoff n-gram. The exact calculation of  $d_r$  and  $\alpha$  are specified in [30] and [14]. What is important to note is the close coordination of backing off in the third term with the discounting in the second term – together they are used to eliminate zero counts and obtain optimal reliability in the language model.

### 5.4.3 Interpolation

In addition to providing an algorithm for combining a well-motivated discounting factor with the benefits of backoff, the Katz algorithm also takes advantage of *interpolation*. In the last step, the probability of an n-gram is calculated iteratively by accounting for all relevant lower-order n-grams. Jelinek-Mercer smoothing [29]

also takes advantage of this recursive interpolation, but does not require calculating the Good-Turing discounting factor or backoff weights. It simply says that, for any n-gram, its probability should be estimated by taking into account the probability of the whole context, as well as the probability of all shorter contexts. This is again meant to balance the accuracy gained from the greater context of a high-order n-gram with the greater reliability of the more frequently seen low-order n-grams. Recursive interpolation is formulated as follows, where  $P_{ML}$  refers to the maximum likelihood probability [10, 14]:

$$P_{interp}(w_i | w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} P_{ML}(w_i | w_{i-n+1}^{i-1}) + (1 - \lambda_{w_{i-n+1}^{i-1}}) P_{interp}(w_i | w_{i-n+2}^{i-1}) \quad (5.11)$$

The interpolation parameters  $\lambda$  determine how much each order n-gram will contribute to the total probability. The values of each  $\lambda_{w_{i-n+1}^{i-1}}$  can be learned via the Baum-Welch algorithm, and may be calculated over word classes. Rosenfeld [52] states that these parameters need not be specified exactly; their variance within 5 % of the best value will create little difference in the perplexity of a held-out set.

Witten-Bell smoothing [6, 67] is another recursive algorithm that interpolates lower-order and higher-order n-gram counts. The key insight is the use of the count of unique word types  $w_i$  that follow a given prefix:  $|w_i : c(w_{i-n+1}^i) > 0|$ . This value is used to determine the values of  $\lambda$  in Eq. (5.11). To determine a particular  $\lambda_{w_{i-n+1}^{i-1}}$ , we divide the *type* count above by the *token* count of all n-grams with that prefix:

$$1 - \lambda_{w_{i-n+1}^{i-1}} = \frac{|w_i : c(w_{i-n+1}^i) > 0|}{|w_i : c(w_{i-n+1}^i) > 0| + \sum_{w_i} c(w_{i-n+1}^i)} \quad (5.12)$$

This calculation is intended to answer the question, “How likely are we to see a new unigram following this prefix?” The answer is used in determining the amount of smoothing at each step of interpolation.

Absolute discounting is another form of interpolation. Ney and Essen [44] and Ney et al. [45] show that using the same discounting factor for all n-grams, regardless of order or frequency, can be effective if that discounting factor is properly set. For instance, an empirical study [45] shows the following estimate to be an effective constant for absolute discounting:

$$Discount = \frac{n_1}{n_1 + 2n_2} \quad (5.13)$$

where  $n_1$  and  $n_2$  indicate the number of singleton and two-count n-grams in the corpus, respectively. Again, the algorithm is recursive so that all lower-order n-grams contribute probability mass to an n-gram. Absolute discounting led to the widely-used Kneser-Ney discounting [35]. In this algorithm, the concern is over how many prefixes a given unigram follows. Put another way, how likely are we to find a given word following a new prefix? For a given unigram, its probability mass calculated over all prefixes will be equal to its maximum likelihood probability:

$$\sum_{w_{i-1}} p_{KN}(w_{i-1} w_i) = \frac{c(w_i)}{\sum_{w_i} c(w_i)} \quad (5.14)$$

This constraint is incorporated into a model that includes an absolute discounting factor and recursion over lower-order backoff models. The modified version proposed by Chen and Goodman [14] is a fully interpolated model – all higher-order n-gram probabilities are discounted and interpolated with lower-order probabilities. The formula for modified Kneser-Ney smoothing, which takes into account both the intuition of [35] regarding known n-gram prefixes and the intuition of [29] regarding interpolation, is given in [14], along with its derivation and further motivation.

In their careful examination of many variations of the smoothing algorithms mentioned, [14] find that their modified Kneser-Ney algorithm is the most successful at reducing the perplexity of a test set. This is due to its use of an absolute discounting factor *and* interpolation of lower-order models over all frequencies of n-grams.

These smoothing methods are not particular to one language or another. They are each designed to help overcome the sparse data problem, to give a non-zero probability to unseen n-grams, and to allow the probabilities of the component n-grams to be distributed more evenly across all parameters of the model. They are used in combination with varying types of language models that incorporate information about semantics and syntax, as described in Sect. 5.5. Several of the models described in Sect. 5.5 also attempt to overcome some limitations of these smoothing algorithms.

## 5.5 Extensions to N-Gram Language Modeling

While the most widely-used language model is the simple n-gram model, there are many variations on the theme. These variations incorporate longer-distance dependencies, syntactic information, and semantic information. The following sections will briefly describe the seminal research for each type of model and its existing or potential application to Semitic natural language processing.

### 5.5.1 Skip N-Grams and FlexGrams

The use of n-gram modeling is pervasive in natural language processing because of its simplicity and surprising effectiveness. Despite the utility of a word’s immediate history in predicting the probability of that word, there are many cases where a slightly different history – not necessarily a longer history, but a different one – may be more effective. For instance, consider the phrase “ravenous lions, voracious tigers, and famished bears.” The appearance of the word *bears* is best predicted by its two predecessors *lions* and *tigers*, but a trigram model will only be privy to the

context *and famished*. One solution to this issue is a skip-gram model, described in [24]. First, the usual collection of n-grams is defined. Then, the parameters are expanded by collecting alternative histories for each word: all bigrams and trigrams within the sentence that skip over  $k$  tokens preceding it. The number of n-grams for a given sentence greatly increases as the number of skips allowed is increased. An evaluation that centers on the coverage, rather than the perplexity, of a test text is useful in this application. Indeed, coverage increases as the number of skips  $k$  increases. When the domain of the train and test are mismatched, e.g., broadcast news as training data and conversational speech in the test set, the coverage does not improve as significantly. This shows that the skip trigrams are not over-generating contexts to the point of false positives. A second evaluation shows that the use of skip-grams can be as effective, or more so, as increasing the size of the corpus. This result may be especially relevant for modeling low-resource Semitic languages such as Amharic, Maltese, and Syriac.

Similarly, [69] describe the use of *flexgrams* to overcome the limited context of n-grams. In this method, the history may arise from anywhere in the sentence preceding the predicted word. This method is combined with morphological decomposition to better model the Turkish language. A reduction in perplexity is shown with the use of flexgrams and morphological information. Therefore, this may be a method that is also applicable to the morphologically rich Semitic languages.

### 5.5.2 Variable-Length Language Models

When the required storage space and computation time of a language model need to be minimized, variable-length language models are useful. To create the optimal model, only the most useful parameters are kept. This can be achieved in a number of ways, including but not limited to the following techniques:

1. Discard longer n-grams if they do not contribute to the model. Kneser [34] shows that longer n-grams can be discarded if they do not meet a frequency threshold, e.g. singletons n-grams, or if they do not reduce the distance between the current LM and the optimal (full) LM. Kneser [34] provides an algorithm for optimizing the pruning function in a computationally tractable way. Seymore and Rosenfeld [57] compare the log probability of the full and reduced model having removed each n-gram in turn, pruning those that do not increase the probability by a threshold amount. Similarly, the method in [62] prunes the LM by measuring the relative entropy of the pruned and original models, where the perplexity of the original training set is used to express entropy. Any n-gram that raises the perplexity by less than a threshold is removed, and backoff weights of the remaining n-grams are re-calculated. Siivola [61] also perform re-calculation of backoff weights and discount parameters after n-grams are removed through pruning, always retaining a model that is properly smoothed according to Kneser-Ney smoothing. This pruning algorithm has been tested

successfully on morphologically rich languages such as Finnish, especially when morphemes are used as the building units for the language model [60]. The method was not as successful for Egyptian Arabic data, however this may have been due to the small size of the corpus. The consistent result in these studies is that the outcomes of applying pruned models vary only slightly from outcomes using larger models, so that time and storage savings are achieved without sacrificing too much accuracy, either in perplexity or application results. Despite the growing use of Kneser-Ney smoothing, [13] show that when aggressive pruning is performed, for instance pruning an LM to a mere 0.1 % of its original size, Katz smoothing is a better choice.

2. Niesler and Woodland [47] take the opposite approach by adding more context to shorter n-grams if they do contribute to the model. In [51], the variable-length n-grams are modeled as a type of Probabilistic Suffix Tree. The branch of a tree can grow only if the resulting model retains a Kullback-Liebler distance less than some threshold from the optimal, full model. Otherwise, the smaller (fewer parameters) model is used. This approach is also used in [61], where a consistent modification of backoff and smoothing parameters are applied to ensure that as the model grows, it has proper Kneser-Ney smoothing at every step. Again, similar task results can be achieved using a more compact language model that excludes low-impact n-grams.
3. A different approach is that of [18], which changes the very construction of an n-gram. Rather than assuming that each unit of an n-gram is a single word token, instead the text is first segmented using the EM algorithm to choose the most informative segments. Collocations, for instance, can be grouped as a single token. This reduces the number of unigrams, creating a smaller model. Perplexity must be calculated over a text that has been segmented in the same fashion.

The benefits of each of these methods is that they provide a more theoretically appropriate model than a fixed-length n-gram model, as words in a sequence are dependent on different length histories. Also, the training algorithms are such that only as many parameters as are useful to the model are used, and the model size can be tuned on held-out data. This is a storage savings for models that would otherwise have size exponential in the order of the n-gram. This is perhaps less appealing when dealing with languages that lack in data, but more appealing for languages with explosive vocabulary growth, such as Modern Standard Arabic. The intuitions that inspired variable length n-gram modeling are surely applicable to language modeling in the Semitic language domain. Despite the lack of literature that applies these models to Arabic or other Semitic languages, it is an area that will likely produce gains in language model accuracy and utility for those languages. In particular, the third approach could be applied after morphological segmentation to reduce the OOV rate and supply the most reliable statistics to the model. The techniques mentioned above could also be applied to morpheme-based n-gram models. Even so, class-based modeling, described in the next section, may be more useful than variable-length modeling for handling large vocabulary

applications; [47] combines class-based modeling with variable-length modeling to positive effect.

### 5.5.3 Class-Based Language Models

One way to reduce the sparsity issue in n-gram modeling is to assign each word to a class. The class may be based on semantic or syntactic principles. If an unknown word can be successfully assigned to a class, then it is easy to apply the statistics of distribution associated with that class to the unknown word. Brown et al. [10] introduced class-based models with the following premise: “If we can successfully assign words to classes, it may be possible to make more reasonable predictions for histories that we have not previously seen by assuming that they are similar to other histories that we have seen.” Where  $c_i$  represents the class that word  $w_i$  is mapped into:

$$p(w_i | w_{i-n+1}^{i-1}) = P(w_i | c_i)P(c_i | c_{i-n+1}^{i-1}) \quad (5.15)$$

Alternatively, one might sum over the classes  $c_i$  if a word is seen as having a distribution over classes. Rather than depend on the token history of each word, we instead calculate the probability of the word given its class and the class history of that word. The intention is to have histories that have been seen more often and are therefore more reliable. There are many ways that the word classes can be derived. In [10], a point-wise mutual information-based measure is used to group words into semantic classes. These classes and the resulting language model can be used in interpolation with word-based models.

As for Semitic language modeling literature, [70] use class-based modeling for an Arabic language application. Classes are used in back-off: when  $w_{i-n+1}^i$  is unknown, the model backs off to  $w_{i-n+2}^i c(w_{i-n+1})$ . Alternatively, the authors create a hierarchy of these back-off possibilities, training a language model for each level of the tree, and linearly interpolating the models. This allows the model to incorporate statistics from the most accurate n-grams and the most reliable class-based statistics.

In [32], class-based language models are developed based on Arabic morphological classes and using word clustering algorithms. These class-based models are combined in a log-linear fashion with stem-based and root-based morpheme models and with two word-based models. The parameters of the log-linear model are derived automatically, and the analysis shows that the morpheme-class model is among the more influential of the group, obtaining a high weight for interpolation. In an ASR application, both of the class-based models bring down the word error rate when combined with the word model, as compared to the word model alone. This shows that interpolating class-based models with more fine-grained information can lead to lower perplexity and improved ASR scores.

The derivation of the class-based models and a more careful comparison to other models, including factored language models, is explored in [33], a study on Egyptian Colloquial Arabic speech recognition. Here, class-based models are defined by morphological components: stems, morphs, roots, and patterns. The models associated with each class are defined and calculated separately and then interpolated. Kirchhoff et al. [33] uses a slightly different formulation of the class-based language model than [10]:

$$P(w_i | w_{i-1}) = P(w_i | c_i) P(c_i | c_{i-1}) P(c_{i-1} | w_{i-1}) \quad (5.16)$$

The class-based models fare well in the ensuing analysis, especially when combined with stream models. Stream models replace each word with a specific morphological component and derive an n-gram model over that morpheme stream. This shows that for Egyptian Colloquial Arabic, very high-level and very low-level information can be successfully combined to improve ASR scores.

### 5.5.4 Factored Language Models

Kirchhoff et al. [33] introduce Factored Language Models (FLMs) with an eye towards Arabic and other Semitic languages. FLMs take into account as many varied kinds of morphological, class, or semantic information as is available for each word. This is an extended form of n-gram modeling where the backoff procedure is more complex than the standard model. Typically, for a trigram model, in the case that a particular trigram is not among the parameters, the model will back off to the most appropriate bigram by ignoring the most distant word and applying a backoff weight [30]. In an FLM, rather than drop the most distant word, instead we consider its class, part-of-speech tag, or other feature for which a reliable weight has been derived. FLMs are formulated as follows for a trigram model, similarly for higher-order or bigram models:

$$p(f_1^{1:K}, f_2^{1:K}, \dots, f_N^{1:K}) \approx \prod_{i=3}^N p(f_i^{1:K} | f_{i-1}^{1:K} f_{i-2}^{1:K}) \quad (5.17)$$

where  $1 : K$  represent the  $K$  features annotating each of the  $N$  words. Modeling is done over  $f$ , a group of factors, rather than a word  $w$ . When an n-gram of complete factors is not specified in the model, backoff can proceed along many routes, dropping any of the factors for which there is an appropriate backoff weight in the model. Many different backoff *paths* can be taken, and different discounting models integrated along those paths. For instance, one might first drop the most specific information in a bundle, the word itself, and use Kneser-Ney discounting to do so. If the n-gram is still not found, one might drop a syntactic tag from the bundle, and use Good-Turing discounting at this juncture. The choice of backoff

can be determined a priori if faith in the features and linguistic knowledge is sufficient to choose a reliable path. However, given the number of possible paths, it is recommended to use an automatic method of choosing or calculating the path. In [33], the authors use generalized parallel backoff, which takes into account all of the possible backoff paths via averaging, multiplication, or a smooth probability distribution.

Alternatively, backoff paths can be chosen via genetic algorithms, as introduced for use with FLMs in [19]. These are the preferred method, as they are able to take into account many if not all of the possible backoff paths, choosing the best in a motivated fashion: the genetic algorithms are trained using perplexity of development data as the optimization criterion. The automatically chosen paths produce a lower perplexity on a test set than n-gram models, hand-chosen backoff models, and random models (but are not compared to generalized parallel backoff).

The benefit of FLMs is the ability to use many morphological and syntactic characteristics of each word; these properties are plentiful in Semitic languages when counting affixes, stems, roots, and other possible morphological categories. The same features that give Semitic languages their ever-expanding vocabularies and intricate complexity are those that can be used to make a more informed language model, without suffering the sparsity problem. On the other hand, one must have tools to produce each of these features, and such tools (especially root finders) are not always available for the lesser-studied languages and dialects. It is sometimes possible to adapt tools from one language to another if there is a tolerance for error; it is unclear whether Factored Language Models are robust to such tagging errors. Given the genetic algorithms, it may be the case that they are robust, as the unreliable features can be dropped early in the backoff path if they do not positively influence the perplexity of the development set (or other training criterion).

### 5.5.5 Neural Network Language Models

Schwenk [55] describes a method of obtaining language model probabilities in a continuous space using neural networks. An input layer, two hidden layers, and an output layer comprise the neural network. The input layer accepts 4-grams extracted from a corpus. The first hidden layer projects this vocabulary of possibly hundreds of thousands of terms into a smaller space of only 50–300 dimensions. The second hidden layer is typical of neural network algorithms: its weights are trained using non-linear back-propagation. The output nodes represent each word in the vocabulary. If the input is a set of words representing a three-word history, then the output layer contains the probability that each vocabulary word follows that history. In training, the optimization criterion is the maximization of the log-likelihood of development data (or minimization of the perplexity, equivalently). These outputs are interpretable as posterior probabilities when the softmax normalization algorithm is applied.

Training neural networks is more time-consuming than estimating a simpler Katz or Witten-Bell language model. To speed training, the output layer may be reduced to only the most frequent words. Tuning the probabilities of these frequent words via neural networks is useful simply because they will appear often in the training data; reducing word error rate on these nodes will more drastically reduce the overall word error rate than will reducing word error rate on rare words. To obtain the necessary coverage of the vocabulary, the neural network models are interpolated with traditional backoff language models.

The estimation of language model probabilities is done in a *continuous* or *distributed* space when using neural networks, as opposed to the discrete calculations performed for traditional back-off models. Continuous probability estimation is better understood, less ad-hoc, and presumably more accurate than discrete models. Neural network models expand linearly with the size of the vocabulary, rather than exponentially as is typical of traditional backoff modeling.

Schwenk [55] describes additional methods to speed training, such as allowing multiple examples to be given as input at each training epoch, and randomly sampling the data from multiple corpora in order to achieve the best adaptation for different language styles. Training over large corpora can also be achieved without great computational demands by using this randomized sampling method. Larger n-grams can be modeled with little effect on training time, however the data sparsity that is attendant with such models is not diminished by using the neural network method.

These methods work well for lattice rescoring with state-of-the-art speech recognition systems in English, French, and Spanish on texts ranging in style: broadcast news, meetings, and conversational data.

Emami et al. [21] apply this method to Arabic data. A variety of morphological data is included in the input layer to create a richer representation of the context, without multiplying the number of parameters and requiring a more complex training technique, as is the case with Factored Language Models. Maximum entropy part-of-speech (POS) tagging, maximum entropy diacritic restoration, and segmentation of words into multiple affixes and stem using a cascaded weighted finite state transducer are the tools used to provide morphologically informed features. These properties are modeled as features of the word histories that are the input to the neural network algorithm. The features are simply concatenated to represent each word as a collection of features. This increases the model size only linearly with the number of features, due to the projection of the input space to a more manageable dimensionality in the first hidden layer of the neural network. Perplexity decreases as more informative features are added to the input, and when the models of varying input types are interpolated. To calculate perplexities, the small-vocabulary neural network model is interpolated with a traditional backoff model with the full vocabulary; the backoff model probabilities are used whenever the neural network model does not have coverage of a word.

Despite the reductions in perplexity, [21] do not show an improvement in word error rate when morphologically-rich neural network language models are used to rescore ASR lattices. This may well have been due to errors in the POS

tagging, segmenting, or diacritic restoration of the recognized word lattices. The work is expanded in [38], where a slightly different set of features are employed: along with segmentation and POS tags, shallow-parse chunk labels and full-parse headwords both preceding and following the predicted word are used as features. The probabilities produced by the neural network language model are used to rescore an N-best list of ASR outputs in place of lattice rescoring. In this study, word error rate did decrease by significant amounts on both broadcast news and conversational data, especially when the more complex syntactic features – previous and following exposed headwords – were included in the neural network language model.

Therefore, the incorporation of morphological and syntactic features into a continuous-space language model has been shown to be beneficial to Semitic natural language processing, in particular in the speech domain.

### 5.5.6 Syntactic or Structured Language Models

Chelba and Jelinek [12] describe a structured language model (SLM) in which a left-to-right parsing mechanism is used to build a model that can be applied in the first pass of speech decoding. The challenge in training and testing is to find the best equivalence class for the history of the current word; the n-gram history is often not sufficient or is misleading, so a syntactically-informed history is used to replace it.

The SLM builds the syntactic structure incrementally; the part-of-speech tags are used to predict the next word as well as that word's tag, the structure of that portion of the parse, and the non-terminal label on the headword. There are multiple dependencies between these parameters, all of which are learned from data in the Penn Treebank.

The model that results is similar to a skip-gram language model, where the number of words skipped in the history of a word depends on the context. Alternatively, the model can be described as a kind of linear interpolation of varied-order n-gram models. Many of the predictions made by this model are long-distance predictions that a trigram model does not capture.

As in other studies, the language models are tested in an ASR application and are found to bring down word error rate a small amount. The computational load of training and applying the SLM, however, are reported to be significant. The use of this model in conjunction with a complementary model that focuses on topical or semantic modeling may be rewarding. Section 5.5.8 discusses a study that does incorporate both syntactic and semantic knowledge.

The incorporation of parse trees into language modeling is, clearly, dependent on the availability of parse trees in that language. At present, this resource and its attendant automatic parsers are available for Modern Standard Arabic, but are in short supply for other Semitic languages or dialects of Arabic. When the reliability of automatic parses is of sufficient quality to merit their use in other tools, then structured language modeling will undoubtedly be useful for Semitic languages.

### 5.5.7 Tree-Based Language Models

Bahl et al. [5] introduce tree-based language models in the context of speech recognition. In this computation-heavy algorithm, binary trees are constructed such that at each node a yes/no question is asked. Each question regards whether some predictor variable, such as the previous word,  $n$ th previous word, previous headword, etc., belongs to a given class of words. The classes are also syntactically defined, including nouns, verbs, and plurals. At the leaves of this tree are sets of predicted words and their probability distribution. An automatic set-building algorithm generates a good, if not optimal, model. The probabilities on the leaves are estimated by taking the average entropy of all of the nodes leading to that leaf; that is, the average conditional entropy of the sets of words defined by the predictors at each node.

The resulting models have lower perplexity than trigram models and would seem to be good complements to them in a log-linear or other combination scheme.

This type of model has not been widely applied, perhaps due to the intensive computing described in the 1989 article. With more modern computing resources, such a model would be less onerous to produce, and could handle a larger vocabulary than 10K words. Future research could show that tree-based language models form a useful complement to the other language model types described in this chapter. The types of features used are perhaps more easily produced than those used in, e.g., FLMs, therefore this might be an appropriate technique to use with resource-poor languages like Maltese and Syriac. Also, that the model improves results on a small-data application may point to its utility for the same languages.

### 5.5.8 Maximum-Entropy Language Models

Maximum Entropy (MaxEnt) models are used to incorporate varying kinds of information in a single language model. They are designed to preserve all uncertainty except where the data explicitly provides evidence for a choice. As described in [8] and elsewhere, the MaxEnt formula takes the form:

$$P_A(y|x) = \frac{1}{Z(x)} \exp\left(\sum_{i=1}^F \lambda_i f_i(x, y)\right) \quad (5.18)$$

The values of  $\lambda_i \in \Lambda$  for each feature  $i$  are calculated over the training set, where every data point  $x$  is described by  $F$  features and a class label  $y$ . Often (but not always), the features are binary, so we can describe a certain feature  $f_i$  as *firing* when the feature is true and variable  $x$  belongs to class  $y$ . The general nature of this algorithm allows the features to encompass many kinds of information. The normalizing factor  $Z$  assures that the resulting probabilities  $(x, y)$  are a proper probability distribution:

$$Z(x) = \sum_{y \in Y} \exp\left(\sum_{i=1}^F \lambda_i f_i(x, y)\right) \quad (5.19)$$

Rosenfeld [52] describes the benefits of using MaxEnt models as well as a step-by-step derivation of the basic training algorithm. In that study, maximum entropy modeling is used to interpolate long-distance functions such as trigger words and skip n-grams together with a baseline trigram model. Mutual information calculations are used to develop trigger models – pairs of words in which each word is informative about the other word’s presence in a document. Skip n-grams form a good counterpart to traditional n-grams, incorporating more knowledge about a word’s history without necessarily incurring the same data sparsity effects of a higher-order model. These kinds of models can be linearly interpolated, which has the benefit of ease of calculation. But, the theoretical grounds of each model are lost; while each model is initially parameterized at a local level, the weights in the interpolation algorithm (trained via EM algorithm) are parameterized on a global level. MaxEnt learning can be used to combine these models without suffering this loss of theoretical grounding. The constraints of each model are relaxed in a way that allows them to interact, benefitting each other and strengthening the predictions. With maximum entropy modeling, the probability of a given bigram will change based on its context and on how functions from the various models handle that context. Experiments show that combining language models via MaxEnt is superior to linear interpolation. Not all of the long-distance models are as useful as hoped, but the interactions are interesting and in many cases complementary. The effect of the trigger model in particular is shown to match its theoretical benefit. The long-distance models are ideal for adaptation from one domain to another, provided that the adaptation training data is appropriate and that local-context models are also incorporated.

Rosenfeld [52] describes training the weights  $\lambda_i$  using the Generalized Iterative Scaling algorithm. Other, more effective models include Improved Iterative Scaling, as introduced in [8], and gradient ascent. Maximum entropy modeling is used in many tasks aside from language modeling; a review and comparison of these and other MaxEnt training techniques is given in [41].

Sethy et al. [56] and Chen et al. [15] describe a model that uses the MaxEnt algorithm together with class-based modeling to improve word error rates in ASR. The introduction of class histories helps to make the model more compact and helps to reduce the data sparsity problem that pervades n-gram modeling. Importantly, [15] show improved scores on an Arabic ASR task using this method.

Sarikaya et al. [54] also use Maximum Entropy modeling to combine varying features in a language model for an Iraqi Arabic ASR task. Word tokens are segmented into morphemes and decorated with part-of-speech and other morphological information. Maximum entropy modeling takes these features, designed in a tree structure, as features to train a morphologically-informed language model. Joint word and morpheme models are used to rescore word-based n-best lists and succeed in reducing word error rate. The same technique, combining morphological with

lexical information in a maximum entropy language model, is also used to get improved results in a statistical machine translation task between English and Iraqi Arabic in [53]. These studies show that maximum entropy modeling is a fruitful arena for combining different kinds of linguistic information, including the kinds of morphological information that have been shown to be useful to Semitic language modeling.

Khudanpur and Wu [31] describe a method for combining semantic and syntactic information in a maximum entropy language model. Semantic information is incorporated by building a topic-dependent language model; in contrast to other studies where a separate LM is built for each topic and interpolated with a full-vocabulary LM, [31] use maximum entropy modeling to create a single, fully-interpolated model with fewer parameters. Furthermore, they derive syntactic information with a trained parser, and incorporate the top  $N$  parses and probabilities into the same language model, again using a maximum entropy technique. The topic and parse data add more global information to the model, allowing it to make predictions with a history greater than only the two previous words. The combined model is successful in lowering word error rates on an English conversational speech ASR task. The resources required to replicate this are significant: to build the topic-based language model, [31] use a corpus with manually labeled topics (per conversation), and a well-trained parser is required for the syntactic information. In performing this task on Semitic data for which fewer resources are available, one might consider using an alternative method for assigning topics to training data, e.g. Bayesian topic modeling, and a simpler parsing methodology that incorporates long-range dependencies.

### 5.5.9 Discriminative Language Models

Discriminative language models are an alternative to n-gram models introduced by Roark et al. [50]. The discriminative model tries to solve the equation:

$$F(x) = \operatorname{argmax}_{y \in GEN(x)} \Phi(x, y) \cdot \bar{\alpha} \quad (5.20)$$

In this equation,  $x$  represents the input to a model; for instance, these inputs may be an acoustic signal of speech. The function  $GEN(x)$  produces all of the possible outputs given this input, that is, all of the possible sequences of words given that acoustic input. The function  $\Phi(x, y)$  is a set of feature values over  $x$  and  $y$ , and  $\bar{\alpha}$  is a set of weights chosen so that, when the function produces the maximum value,  $y$  is the correct sequence of words spoken in the acoustic signal. The dual concerns of discriminative language modeling are finding an appropriate and effective feature vector  $\Phi$  and an algorithm for training the weights  $\bar{\alpha}$ . As regards feature vectors, the technique used in [50] takes advantage of application-specific information to build the most effective models. Regarding the training of  $\bar{\alpha}$ , the authors show

how two discriminative models, the perceptron algorithm and the global conditional log-linear model (GCLM) can be successfully applied.

The steps of the process are:

1. Build a baseline recognizer with an acoustic model and n-gram language model that produces output in lattice format.
2. Use these baseline lattices to derive features for discriminative LM training, and as the training data for training the weights  $\bar{\alpha}$ .
3. Use the new language model parameters to refine the decoder output in a second pass over the lattices.

Therefore, the set of outputs  $GEN(x)$  on which the discriminative model is trained are the sequences in the baseline lattices. There are two kinds of features that are derived from these lattices: the log probability of the best path  $y$  (either the correct path if it exists in the lattice, otherwise the minimum error path is found), and the frequency of each n-gram in  $y$ . Rather than estimate probabilities for all n-grams, only those n-grams found in the best path  $y$  are considered. This produces great savings in training and storage for the discriminative language model.

The perceptron algorithm increases or decreases the weights of  $\bar{\alpha}$  at each iteration whenever the predicted output is incorrect, eventually guiding the algorithm to the maximum number of correct answers. In order for the training to converge, the data must be separable. This means that for each incorrect answer, its score is at least  $\delta$  different than the score for the correct answer. The number of training iterations is dependent on the value of  $\delta$ . To apply the perceptron algorithm for training, the weights on all paths in the baseline lattice are initialized to zero and are adjusted so that  $y$  receives the greatest weight. The final weights on the lattice become the parameters  $\bar{\alpha}$ . These parameters are used both for second-pass decoding, and as the feature set and starting point for the global conditional log-linear model (GCLM) training algorithm.

GCLMs, in contrast, are exponential, and assign a conditional probability to each member of  $GEN(x)$ .

$$p_{\bar{\alpha}}(y|x) = \frac{1}{Z(x, \bar{\alpha})} \exp(\Phi(x, y) \cdot \bar{\alpha}) \quad (5.21)$$

The conditional distribution over the members of  $GEN(x)$  are used to calculate the log probability of the training data, given parameters  $\bar{\alpha}$ . In GCLM, the weights on the language model represent probability distributions over the strings. The same features as mentioned above are used for training. The parameters of the model are estimated from the training data and applied as a language model in a second pass of decoding.

Both the perceptron and GCLM models are trained as weighted finite state automata (WFSA). The  $\bar{\alpha}$  parameters are derived from the WFSA. The number of features used in training can be kept to a minimum by only considering those n-grams that appear in baseline output, which also results in only the most crucial

parameters being included in the model. The resulting language model can be directly combined with the baseline lattice for re-scoring.

Presumably, the same approach could be taken with machine translation and its lattices. The refined language models will then be configured to best discriminate between semantically, rather than acoustically, confusable terms.

Kuo et al. [37] introduce a third method of training the discriminative LM, the minimum Bayes risk calculation. In this case, the  $\bar{\alpha}$  parameters of the model are trained by calculating the number of errors contained in each hypothesis. The loss associated with each hypothesis is compared to the same loss without a given feature included; if the loss is worse without the feature, then  $\alpha_i$  associated with feature  $i$  is positively updated, otherwise it is negatively updated. Kuo et al. [37] show that this algorithm produces improved word error rate on ASR for Modern Standard Arabic, especially when morphological features are used to reduce the out-of-vocabulary rate. Additionally, the authors describe the Bayes risk metric, which allows them to train the discriminative LM in an unsupervised way. Instead of comparing each hypothesis to a reference transcription, each hypothesis is compared to every other hypothesis, and the sum of the differences represents the Bayes' risk metric:

$$L(y|x_i) = \sum_{y' \in GEN(x_i)} L(y, y') p(y'|x_i) \quad (5.22)$$

where  $L(y, y')$  represents the number of errors in hypothesis  $y$  as opposed to  $y'$ , and  $p(y|x_i)$  is the posterior probability of hypothesis  $y$ . The hypotheses and probabilities are given by a baseline recognizer, rather than by hand transcription. The hypothesis that minimizes this equation,  $y$ , is considered the gold standard for perceptron and GCLM training. This allows the use of more training data for which hand transcriptions are not available, and as one expects, the additional training data improves WER scores.

Arisoy et al. [4] revisits discriminative language modeling with a Turkish ASR application. The rich morphology of Turkish serves as a good example of how discriminative LMs, in particular those that include morphological as well as whole-word features, might work with similarly rich Semitic languages. In this study, the authors experiment with features derived from word n-grams, word-based syntactic features, morph n-grams (derived via Morfessor [17]), syntactic features estimated over the morph sequences, and topic-based features estimated using a *tf-idf* vector methodology. The results show that while morph-based features and syntactic features (both word- and morph-based) in the discriminative LM framework improve ASR results, the semantically-informed topic features and long-distance morph relation features are not useful. It is also the case that unigram features are better than higher-order n-gram features in the discriminative LM. The discriminative modeling and morph-based features help overcome the OOV problem and data sparsity that occur in Turkish, and the lessons may well be applied to building language models for Arabic and other Semitic languages for ASR or other NLP applications.

### 5.5.10 LSA Language Models

Bellegarda [7] describes a method for integrating long-range semantic information into the normally short-range context of n-gram models. This method first involves calculating the Singular Value Decomposition (SVD) of a word-by-document matrix derived from a large set of training data that is semantically similar to the test data. The decomposed and truncated matrices derived via SVD can be used to represent the words and documents in a clustered semantic space. In other words, it becomes possible to compare each word to other words in the training set based on their distribution among documents. The decomposition effectively takes into account all word pairs and all document pairs in deriving the similarity of all words and documents. A new document can be clustered into a semantic grouping of similar documents from the training set by projecting it into the SVD space. In terms of n-gram modeling, we can model the probability of each word in a document given the semantic history of the entire preceding document. This semantic language model prediction is interpolated with typical n-gram modeling in the following way:

$$P(w_i | H_{i-1}^{(n+l)}) = \frac{p(w_i | w_{i-n+1}^{i-1}) \frac{p(w_i | \tilde{d}_{i-1})}{p(w_i)}}{\sum_{w_i \in V} p(w_i | w_{i-n+1}^{i-1}) \frac{p(w_i | \tilde{d}_{i-1})}{p(w_i)}} \quad (5.23)$$

where  $w_i$  is the current word,  $p(w_i | \tilde{d}_{i-1})$  is the LSA model probability for the word given the semantic history, and  $V$  is the vocabulary.  $H_{i-1}^{(n+l)}$  represents the integration of the n-gram probability  $n$  with an LSA probability  $l$  for the history of the document up until word  $w_i$ . This calculation provides a natural interpolation of long- and short-term context. The component  $\tilde{d}_{i-1}$  can be smoothed by referring to the word or document clusters rather than to the vectors of the individual words seen in the document. Further tuning can be applied to determine how much of the previous context should be taken into account for each word.

The experiments in [7] show good reductions in perplexity and word error rate when incorporating the LSA history into the standard bigram and trigram model. However, some caveats apply. The semantic type of the training data used to develop the LSA model must be tightly coupled with that of the test data, otherwise the effectiveness of the extra modeling is greatly diminished. Luckily, not a very large amount of data is necessary to construct an effective model. Also, while the LSA modeling is very useful as concerns content words, those that control the broad semantic topics and ideas of a document, it is not effective at all in discriminating between function words that appear in all documents. As these tend to be the words most likely to be mis-recognized in an ASR context, the benefit of LSA may be limited for that task. However, while function word mistakes may be common, mistakes in content words may be more harmful to actual understanding, so that the mistakes that are avoided by using an LSA-interpolated model may outweigh the function word mistakes in some evaluations. Furthermore, the long-distance semantic system can also be interpolated with a near-term, syntactically trained language model.

The Semitic NLP literature is missing reference to LSA language modeling. This may be because a major obstacle in Semitic NLP is often the precise word choice, i.e. choosing the right syntactic word form from among many semantically identical ones, for which LSA modeling is not helpful. However, in natural language generation, LSA modeling may help to provide some variation in phrase creation, and it could also be useful in English-to-Semitic machine translation.

### 5.5.11 Bayesian Language Models

Two related goals motivate the language modeling work of [66]: to incorporate a corpus-based topic model into an n-gram language model, and to do so using a Bayesian algorithm. The large-context topic modeling serves to complement the smaller-context n-gram modeling. The Bayesian paradigm enforces a method in which all assumptions about the data are formulated mathematically in order to be incorporated into the model. The work of [66] is based largely on that of [40]; the discussion below draws from both studies.

The path to achieving both goals is found in Dirichlet modeling, which is used as a replacement for the simpler kinds of smoothing and backoff described in Sect. 5.4. A Dirichlet model provides a prior model of the data. Assume a matrix  $\Phi$  that describes word transition probabilities over a training corpus. Each row  $w'$  of  $\Phi$  represents the probabilities  $P(w|w')$ . These rows make up the parameters of the model  $Q$ . However, the parameters of  $Q$  are usually poorly estimated because of the sparse data problem that affects all language modeling attempts. The real parameters of the data are unknown; we can easily imagine that there are multiple possible parameters of  $Q$ . The *distribution* of the possible parameters of  $Q$  is estimated by the Dirichlet model. The parameters of the Dirichlet model – a hierarchical, exponential model – are the *hyperparameters* of the corpus data. The crucial parameters of the Dirichlet model are usually notated as  $\alpha$  and  $m$ ;  $\alpha$  describes the peakiness of the distribution, while  $m$  describes its mean. In most of Chap. 3 of her dissertation, Wallach replaces these with  $\beta$  and  $n$ , respectively, and that is the notation that will be used in this summary. The focus here is the *use* of the Dirichlet model in language and topic modeling; for a further explanation of its properties and estimation, please see [66] and [40].

Typical bigram modeling starts with the same transition matrix  $\Phi$ , using a formulation like Eq. (5.24) to predict the probability of a test corpus given the training data:

$$P(\mathbf{w}|\Phi) = \prod_w \prod_{w'} \phi_{w|w'}^{N_{w|w'}} \quad (5.24)$$

The term  $\phi_{w|w'}$  describes the probability of the bigram  $w'w$  in the training data, while the term  $N_{w|w'}$  is an exponent of  $\phi$  describing the bigram's count in test data. Smoothing and backoff algorithms such as those described in Sect. 5.4 are

usually used to enhance this formulation to account for the sparse data problem. Those methods are replaced here with a Dirichlet prior, which counters the sparse data problem by estimating the probability of our training data  $\Phi$  over the hyperparameters  $\beta n$  of the Dirichlet model:

$$P(\Phi|\beta n) = \prod_{w'} Dir(\phi_{w'}|\beta n) \quad (5.25)$$

When we substitute Eq. (5.25) into Eq. (5.24), we can estimate  $P(\mathbf{w}|\beta n)$ , the probability of the test corpus given the Dirichlet hyperparameters:

$$P(\mathbf{w}|\beta n) = \prod_{w'} \frac{\prod_w \Gamma(N_{w|w'} + \beta n_w)}{\Gamma(N_{w'} + \beta)} \cdot \frac{\Gamma(\beta)}{\prod_w \Gamma(\beta n_w)} \quad (5.26)$$

$\Gamma$  is a function used in the estimation of the Dirichlet model. At the test stage, we can predict the probability of word  $w$  given a previous word  $w'$  and the estimated model using the following formulation:

$$P(w|w', \mathbf{w}, \beta n) = \frac{N_{w|w'} + \beta n_w}{N_{w'} + \beta} \quad (5.27)$$

$$= \lambda_{w'} n_w + (1 - \lambda_{w'}) \frac{N_{w|w'}}{N_{w'}} \quad (5.28)$$

Equation (5.28) is meant to evoke a typical interpolation model:  $\frac{N_{w|w'}}{N_{w'}}$  is a bigram probability; we back off to a unigram probability described by  $n_w$ , and these are interpolated by parameter  $\lambda_{w'}$ . In the Dirichlet formulation, the concentration (peakiness) hyperparameter  $\beta$  takes the place of EM-estimated  $\lambda$ , and the  $n_w$  hyperparameter takes the place of Good-Turing or another kind of smoothing discounting for  $w$ . The result is a formulation of the smoothing parameters of the bigram  $w'w$  through the calculation of a Dirichlet prior rather than through EM estimation.

Similar intuitions are used to apply the Dirichlet prior to topic modeling. This is known as latent Dirichlet allocation, and is also described in [9]. Earlier we described a matrix  $\Phi$  in which the rows represent the probabilities of transition from word  $w'$  to each word  $w$ . Now we instead define  $\Phi$  as the probabilities of transitioning from topic  $t$  to words  $w$ . The set of topic distributions are notated as  $\mathbf{z}$ . Furthermore, we assume that a given document will include a subset of possible topics, so there is a second transition matrix  $\Theta$  from documents to topics. The estimation of each of these two matrices is again based on sparse data, and therefore it is useful to estimate a Dirichlet prior describing their distributions. We estimate hyperparameters  $\beta$  and  $n$  to describe the distribution of possible matrices  $\Phi$ , and separate hyperparameters  $\alpha$  and  $m$  to describe the distribution of possible matrices  $\Theta$ .

To estimate the probability of a word using a topic-based model, we apply the parameters  $\beta$  and  $n$ :

$$P(w|t, \mathbf{w}, \mathbf{z}, \beta n) = \frac{N_{w|t} + \beta n_w}{N_{*|t} + \beta} \quad (5.29)$$

$$= \lambda_t n_w + (1 - \lambda_t) \frac{N_{w|t}}{N_t} \quad (5.30)$$

The term  $N_{*|t}$  refers to the distribution of all words over that topic  $t$ . Similarly, to calculate the probability of a given topic given a document model, we apply the hyperparameters  $\alpha$  and  $m$ :

$$P(t|d, \mathbf{w}, \mathbf{z}, \alpha m) = \frac{N_{t|d} + \alpha m_t}{N_{*|d} + \alpha} \quad (5.31)$$

$$= \lambda_d m_t + (1 - \lambda_d) \frac{N_{t|d}}{N_d} \quad (5.32)$$

The models resulting from latent Dirichlet allocation have properties in common with those resulting from latent semantic analysis, described in Sect. 5.5.10.

Given a topic model and a bigram model, the next challenge is to find a method to combine their predictions. When using LSA to incorporate topical information (Sect. 5.5.10), the topic models are combined with n-gram models using linear interpolation. In contrast, to retain and extend the Bayesian properties of the component n-gram and topic models described here, [66] calculates the joint probability of a word given both its n-gram context and its topic context by combining Eqs. (5.27) and (5.29). Stepping back to the initial model of the training corpus, the transition probability matrix  $\Phi$  will now have rows defined as the probability of transition from word  $w'$  and topic  $t$  to word  $w$ :  $P(w|w't)$ . The Dirichlet hyperparameters describe the probability distribution of this jointly estimated matrix  $\Phi$ . If the contexts  $w't$  are modeled as strictly joint, then the estimation is formulated as:

$$P(\Phi|\beta n) = \prod_{w'} \prod_t Dir(\phi_{w't}|\beta n) \quad (5.33)$$

The hyperparameters  $\beta$  and  $n$  are fully tied, shared between the contexts  $w'$  and  $t$ . Alternatively, the contexts  $w'$  might be modeled separately for each topic:

$$P(\Phi|\{\beta n_t\}_{t=1}^T) = \prod_{w'} \prod_t Dir(\phi_{w't}|\beta n_t) \quad (5.34)$$

Here the hyperparameters capture topic-specific similarities between words. Lastly, the distributions of  $t$  could be modeled separately over each context  $w'$ :

$$P(\Phi|\{\beta n_{w'}\}_{w'=1}^W) = \prod_{w'} \prod_t Dir(\phi_{w't}|\beta n_{w'}) \quad (5.35)$$

In this last case, the parameters capture information about common, topic-independent bigrams.

The choice of prior as well as the choice of estimation procedure affect the outcome, as better choices result in more interpretable topics and more informative models. For jointly estimating word and topic contexts, [66] shows Eq. (5.34) to be most effective; it is more effective than Dirichlet-estimated bigram or topic models alone, and more effective than using either of the other priors (Eqs. 5.33 and 5.35) to estimate the joint model.

The models are evaluated by a measure related to perplexity, differing from the usual formulation due to the use of topic models and the alternative method of estimating the models. The results over a small training and test set show that the proposed algorithms are effective in reducing the information rate of the models. Furthermore, the words comprising topic models are interpretable, showing that the method is as useful as LSA in incorporating semantic properties of the corpus into the language model. The differences in evaluation technique, however, preclude the comparison of the Bayesian topic-based models in [66] to typical language modeling methods. Ostensibly, this method of calculating the models and of testing them is task-independent, but no task-based evaluation is given. The questions of whether this technique is amenable to use in speech recognition or machine translation in terms of storage and processing requirements, and whether the information theoretic gains extend to gains in task results, remain for further research.

As for tasks within the Semitic language realm, the usual questions apply: should the word contexts be determined over unique tokens, or should stemming be applied first? More generally, will using this technique for estimating smoothing parameters be useful in solving the sparse data problem that most affects Semitic languages – the growth of vocabulary due to morphological richness? Perhaps the incorporation of a third kind of context, one that models morphological properties, will result in the most useful model for Semitic language applications.

## 5.6 Modeling Semitic Languages

The specifics of morphological and grammatical principles of Semitic languages are covered elsewhere in this volume. There are multiple morphological functions, including both root-and-pattern and affixational morphology, that result in an abundance of unique word forms. As for handling this rich morphology in automatic processing, there are two overall devices: tokenization, which reduces the number of unique words by separating clitics from the main semantic information, and stemming, which reduces the number of unique word forms by arranging words into classes by their common properties, ignoring the less semantically salient properties that make them different. Tokenization can often be applied through simple scripts that look for specific alphabetic sequences at word boundaries. An information-theoretic approach to tokenization can also be taken when affix and stem dictionaries are unavailable (e.g. Morfessor, [17]). There are multiple

approaches to stemming, ranging from language-independent statistical methods of deriving morphemes to linguistically complex methods such as the multi-featured, context-sensitive classes derived through the use of parsers and part-of-speech taggers for Factored Language Models [33]. This section will review studies that incorporate tokenization, stemming, or both in order to improve language models for Semitic natural language processing.

### 5.6.1 Arabic

Vergyri et al. [65] and Kirchhoff et al. [33] both describe the utility of incorporating morpheme information into language models for Arabic speech recognition. Both of these studies use Egyptian Colloquial Arabic data and Factored Language Models (FLMs), described above in Sect. 5.5.4. Automatic means are used to derive morphemes as well as other kinds of morphological and syntactic information. These studies focus on how FLMs and associated technologies such as generalized parallel backoff can be used to successfully incorporate this knowledge into a useful language model, eventually reducing word error rate on an ASR task. One obstacle that must be overcome is the combination of recognized morphemes into words without generating spurious words, or words not in the original vocabulary. Another aspect of this technology is the decision of when to incorporate the morpheme models; if the model is too complex to be effectively incorporated into first-pass decoding, it can still be beneficial to use morpheme and enhanced models in a second-pass decoding of word-based lattices. The experiments in [33] also effectively separate the aspects of morpheme-based language modeling from the FLM technique in a useful way; it is clear in this study that the use of the more complex FLMs indeed improves upon a more simple incorporation of morphological knowledge into the language model.

The use of morphemes in Arabic language modeling is also discussed in [21], discussed in Sect. 5.5.5 above. Morphemes derived via a finite state algorithm, part of speech tags, and automatically-derived short vowel information are used as input to a neural network language modeling tool. While the gain in word error rate is negligible in this study, the follow-up study [38] does show that accurately derived morphological information can be used to successfully lower word error rate when incorporated into neural network language models.

When working with a dialectal form of Arabic, there is a tension between using morphological information derived from the more data- and resource-replete Modern Standard Arabic and creating new tools to work with the particular dialect, for which there may not be adequate text for training new models. Afify et al. [2] combine tools and knowledge from both MSA and Iraqi Arabic in creating morpheme-based language models for the Iraqi dialect. First, a set of affixes are pre-defined based on linguistic knowledge of the dialect. The affixes are segmented from the words based on orthographic information. These segmentations are then refined by applying knowledge from the Buckwalter Arabic Morphological Analyzer [11],

which encodes information about legal stems in MSA. Heuristics such as setting a minimum stem length and checking stem accuracy against the stem dictionary limit the number of segmentations of each word. The resulting segmentations are used to derive both the language model and pronunciation dictionaries. The morpheme models result in a lower out-of-vocabulary rate and a lower ASR word error rate, especially when interpolation is used to smooth the word and morpheme language models. The interpolation allows the model to take advantage of the longer context of word models and the greater token coverage of morpheme models.

Xiang et al. [68] use a similar method of segmentation constrained by defined MSA stems to produce morpheme-based language models of Arabic. When automatically derived phonetic information is incorporated into both these language models and corresponding acoustic models, word error rates drop. An important observation of this study is that words must be segmented carefully; doing so without constraint results in models with greater acoustic confusability. Incorporating phonetic knowledge into the segmentation algorithm aids in producing models that are better aligned with the challenges encountered in ASR. Constrained segmentation of Arabic words into morphemes is also explored in [39] and [16]. In these studies, the statistically-derived morpheme segmentations are constrained by pre-defined affix and stem lists. The benefit of using these morphemes in language modeling for ASR may be limited to small-vocabulary tasks [16], but in some cases may also be useful in large-vocabulary tasks [46].

Heintz [27] uses finite state machines to identify all of the possible stems in a word according to a set of pattern templates. Together with a set of pre-defined affixes, the segmentation into morphemes produces a text over which useful language models can be estimated. However, there is no significant reduction in word error rate in either Modern Standard Arabic or Levantine Arabic speech recognition experiments. The use of simpler techniques such as Morfessor [17] or affix-splitting for morpheme creation is recommended.

Marton et al. [42] discusses the use of tokenization of an Arabic language model for translation from English to Arabic. (The main thesis of this study regards paraphrasing the source language, English, but this does not affect the language modeling of the target language, Arabic.) Tokenization and normalization are performed using MADA, a morphological analysis tool trained to work with Modern Standard Arabic data. They note that the process of *de-tokenization*, returning the morphemes to word status by correctly stringing them back together, is a non-trivial and important step in processing, and discuss a successful technique.

### 5.6.2 Amharic

In [1], bigram language models are incorporated into an Amharic large-vocabulary automatic speech recognition system. The system is straightforward and without complications of morphological processing. The use of only whole words results in a rather weak model due to data sparsity – both perplexity and the number of

singleton words are large. The author points to the use of syllable modeling as a useful prospect for segmenting Amharic text; the use of syllables in the acoustic modeling experiments, tested separately from language modeling, is shown to be beneficial.

Tachbelie and Menzel [63] build on the work for Amharic NLP begun in [1]. In this study, the authors use the Morfessor algorithm [17] to segment the Amharic train and test texts, greatly reducing the out-of-vocabulary rate. A comparison of word and morpheme language models via log probability (see Sect. 5.2) shows that the word-based model provides a better encoding of the test text. Still, reducing the out-of-vocabulary rate should be useful for application of the model in an NLP task, and the language-independent Morfessor algorithm is sufficient to provide adequate segmentation for this purpose. The authors test their hypotheses further in [64]. Here the speech recognition system developed in [1] is used to test several morpheme-based language models in Amharic. A manual segmentation of words into morphemes is compared to the Morfessor-derived morphs. The morpheme models are applied by expanding word bigram lattices into morpheme-based lattices and finding the new best path. Linguistically (manually) derived morphemes, and trigrams in particular, are found to produce better results in this second-pass decoding. Factored language models are also explored, with word, POS, prefix, root, pattern and suffix features. Pre-determined and genetic algorithm backoff models are applied in this study. FLMs, especially those that include POS information and use genetic algorithms to find the best backoff path, produce better decoding results than the simpler methods described above.

Pellegrini and Lamel [48] also study the application of NLP techniques to Amharic. Words are decomposed using the Harris algorithm, which looks for morpheme boundaries at places where the number of possible subsequent letters is large. Splitting words at these affix boundaries effectively reduces the size of the vocabulary. Furthermore, ASR experiments using re-trained morpheme acoustic models are successful in reducing the word error rate, both when morphemes are counted as ‘words’ and when the morphemes are recombined into words. However, it is found in later studies that this result is restricted to this experiment with its small 2-h acoustic model training set. In [49], the word decompounding is performed with Morfessor, modified to include a Harris probability for word boundaries. The effect of this is to favor short morphs, which correspond better to the kind of concatenative morphemes (affixes) that are found in Semitic languages. Furthermore, the algorithm is enhanced with information about phonetic distinctive features. A constraint is added to each model regarding acoustic confusability, derived from acoustic alignment tables. This biases the algorithm to avoid creating monosyllables that are acoustically confusable. All of the morpheme models reduce the vocabulary size of the training set as compared to word models. In applying these models in an ASR task, the lowest word error rate occurs with the most complex model, which uses both morphological and phonetic constraints in its segmentation. This study uses a 35-h training set, and finds that the use of a phonetic constraint is required when deriving morphemes for language modeling to reduce word error rate in Amharic ASR.

### 5.6.3 Hebrew

Netzer et al. [43] use a trigram-model trained on 27 million Hebrew words for the task of predicting likely words following a given context. The task envisioned in this study is mobile device input, therefore no acoustic or translation models are necessary. A morphological and syntactic model are incorporated to account for rich Hebrew morphology, but in this case, results are worsened. The application is such that only a partial sentence is used in computing and applying morphological and syntactic information at test time, which likely accounts for the unexpected results. It is important to have sufficient context when applying morphological and syntactic models to modify or enhance both training and test texts.

Shilon et al. [58] is a study of Hebrew-Arabic machine translation. There is great difficulty in building such a system, as there is no sufficient corpus of parallel language data. Target-language LMs can be built on the available single-corpus data, but it is also essential to incorporate linguistic rules in the decoding phase that account for the morphological and syntactic differences that exist between Hebrew and Arabic. A second difficulty is the fact that the lattices output by the decoder tend to be very large, both for Hebrew and Arabic as the target language. Again, this is due to the morphological richness of both languages, and the lack of parallel corpora to help align and filter the output. In general, the language models are not able to sufficiently assign accurate probabilities to all of the sequences in the lattices. This study points to the amount of work left to be realized, both in building and applying resources like parallel corpora and knowledge-rich language models, for the successful application of statistical machine translation in Arabic and Hebrew.

### 5.6.4 Maltese

The morphological characteristics of Maltese are somewhat different than those of the other Semitic languages discussed in this book. This impacts the kinds of techniques that might be used in language modeling for Maltese.

As stated earlier in the book, Maltese is written with Latin text. Unlike Modern Standard Arabic or Hebrew, all vowels are included in the orthographic form. If vowel variation were to occur within stems as it does in the derivational morphology of Hebrew and Arabic, this would produce a great increase in the number of word forms in Maltese. However, it is the case that vowel harmony is not used for morphological purposes, so vocabulary size should not increase dramatically due to the inclusion of vowels in the orthography.

The problem of vocabulary expansion in Modern Standard Arabic and other Semitic languages comes from the combinations of many affixes and clitics used to encode morphological information such as gender and number agreement, case markings, object pronouns, tense, and aspect. Maltese also uses affixes for some of these purposes, so that the use of a stemmer may reduce the out-of-vocabulary rate.

However, Maltese uses affixational morphology somewhat less frequently than is seen in other Semitic languages. For instance, there is no case marking on nouns, and some tense and aspect markers are separate particles, meaning that the amount of vocabulary expansion will not be as great for Maltese. Therefore, larger training sets may compensate for the growth in vocabulary due to affixational morphology more quickly than is the case in MSA. If adequate training data is not available, stemming techniques are likely to be useful in generating the most reliable language models.

It is also the case that Maltese has relatively free word order. The ordering of verb, subject, and object changes based on the topic of the sentence. This free word order means that exact n-grams are less likely to be repeated throughout a corpus. A modeling technique like skip n-grams or flexgrams (Sect. 5.5.1) may be especially useful in Maltese for capturing the history of a word without relying too heavily on word order.

### **5.6.5 Syriac**

It is likely that the most pressing issue in language modeling for Syriac will be the lack of training data. This issue will be exacerbated by the existence of several forms of orthography. Therefore, a crucial step will be the normalization of the orthography in order to take advantage of all available data.

As in MSA and Hebrew, many vowels are not normally marked in the orthography, reducing the number of word forms that must be predicted. Affixational morphology is used on verbs in Syriac to encode number, gender, tense, and object pronouns, as well as possessive pronouns on nouns. Given the probable lack of training data, stemming will be useful to decrease the out-of-vocabulary rate. The regularity of the affixes and clitics may mean that a surface segmenter such as Morfessor [17] will be adequate to provide useful morphemes, without resorting to a deep morphological analysis.

### **5.6.6 Other Morphologically Rich Languages**

Languages such as Turkish and Finnish also exhibit high degrees of morphological complexity resulting in a quickly expanding vocabulary and high out-of-vocabulary rate [25, 28]. Many of the techniques for dealing with the resulting data sparsity in Semitic languages can be used to address the same problem in these other languages, and vice versa. For instance, Sect. 5.5.9 mentions the work of [4] in which morpheme and lexical information is used in a language model to improve the results of a Turkish ASR task. Hacioglu et al. [26] also describe a morpheme-splitting algorithm for Turkish, where the morphemes rather than words are again used to improve ASR. Siivila et al. [59] discuss the use of stemming in building

language models for Finnish ASR. Hirsimäki et al. [28] introduced the Morfessor algorithm for segmenting words into statistically-derived “morphs” by applying it to Finnish data. These and other studies on morphologically rich languages can certainly inform natural language processing work in Semitic languages.

## 5.7 Summary

In this chapter we have explored basic n-gram modeling and many variations on that technique that allow the incorporation of semantic, syntactic, and even phonetic knowledge into the models. Within Semitic language modeling, the focus has been on how best to acquire and make use of morphological information to provide context and knowledge for the language models. Modeling techniques that make this incorporation simpler, in particular factored language modeling and neural network modeling, seem most promising for Semitic applications.

The use of one model over the other will often depend not only on the task, but also on the resources available. When working with languages for which resources are scarce, simpler linguistic extraction methods, such as statistical segmenting, often do a sufficiently accurate job to provide informative features to the language models. Exploration of the simpler expanded language modeling techniques, such as skip n-grams and flex-grams, may be of greatest use in languages like Amharic, Syriac, and Maltese, where corpus data and automatic tools for parsing, etc. are difficult to acquire. These techniques have been shown to be as effective in improving language models as adding more data – a crucial factor when very little data is available.

When corpus data as well as automatic means for extracting part-of-speech labels, roots, or other syntactic or morphological information are available, then the more complex modeling methods that can incorporate this data will have an advantage. Further study of more complex methods such as Bayesian topic-based language modeling may be appropriate for languages such as Modern Standard Arabic and Modern Hebrew, for which data and tools are available.

## References

1. Abate ST (2006) Automatic speech recognition for Amharic. PhD thesis, Universität Hamburg
2. Afify M, Sarikaya R, Kuo HKJ, Besacier L, Gao Y (2006) On the use of morphological analysis for dialectal Arabic speech recognition. In: Proceedings of ICSLP, Pittsburgh
3. Al-Haj H, Lavie A (2012) The impact of Arabic morphological segmentation on broad-coverage English-to-Arabic statistical machine translation. Mach Trans 26:3–24
4. Arisoy E, Saraçlar M, Roark B, Shafran I (2012) Discriminative language modeling with linguistic and statistically derived features. IEEE Trans Audio Speech Lang Process 20(2):540–550

5. Bahl LR, Brown PF, Souza PVD, Mercer RL (1989) A tree-based statistical language model for natural language speech recognition. *IEEE Trans Acoust Speech Signal Process* 37:1001–1008
6. Bell T, Cleary J, Witten I (1990) Text compression. Prentice Hall, Englewood Cliffs
7. Bellegarda J (2000) Exploiting latent semantic information in statistical language modeling. *Proc IEEE* 88(8):1279–1296
8. Berger AL, Pietra SAD, Pietra VJD (1996) A maximum entropy approach to natural language processing. *Comput Linguist* 22(1):39–71
9. Blei DM, Ng AY, Jordan MI (2003) Latent Dirichlet allocation. *J Mach Learn Res* 3:993–1022
10. Brown P, Pietra VD, deSouza P, Lai J, Mercer R (1992) Class-based  $n$ -gram models of natural language. *Comput Linguist* 18:367–379
11. Buckwalter T (2004) Buckwalter Arabic morphological analyzer version 2.0. <http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2004L02>
12. Chelba C, Jelinek F (2000) Structured language modeling. *Comput Speech Lang* 14:283–332
13. Chelba C, Brants T, Neveitt W, Xu P (2010) Study on interaction between entropy pruning and Kneser-Ney smoothing. In: INTERSPEECH 2010, Makuhari, Chiba, pp 2422–2425
14. Chen S, Goodman J (1999) An empirical study of smoothing techniques for language modeling. *Comput Speech Lang* 13:359–394
15. Chen S, Mangu L, Ramabhadran B, Sarikaya R, Sethy A (2009) Scaling shrinkage-based language models. In: Automatic speech recognition & understanding, Merano/Meran, pp 299–304
16. Choueiter G, Povey D, Chen SF, Zweig G (2006) Morpheme-based language modeling for Arabic LVCSR. In: Proceedings of ICASSP, Toulouse, pp 1053–1056
17. Creutz M, Lagus K (2007) Unsupervised models for morpheme segmentation and morphology learning. *ACM Trans Speech Lang Process* 4:1–34
18. Deligne S, Bimbot F (1995) Language modeling by variable length sequences: theoretical formulation and evaluation of multigrams. In: Proceedings of ICASSP, Detroit, vol 1, pp 169–172
19. Duh K, Kirchhoff K (2004) Automatic learning of language model structure. In: Proceedings of COLING 2004, Geneva
20. El Kholy A, Habash N (2012) Orthographic and morphological processing for English-Arabic statistical machine translation. *Mach Trans* 26:25–45
21. Emami A, Zitouni I, Mangu L (2008) Rich morphology based n-gram language models for Arabic. In: Interspeech 2008, Brisbane, pp 829–832
22. Gale WA, Church KW (1994) What's wrong with adding one? In: Oostdijk N, De Haan P (eds) Corpus-based research into language. Rodolpi, Amsterdam
23. Gale WA, Sampson G (1995) Good-turing frequency estimation without tears. *J Quant Linguist* 22:217–37
24. Guthrie D, Allison B, Liu W, Guthrie L, Wilks Y (2006) A closer look at skip-gram modelling. In: Proceedings of LREC, Genoa
25. Guz U, Favre B, Hakkani-Tür D, Tur G (2009) Generative and discriminative methods using morphological information for sentence segmentation of Turkish. *IEEE Trans Audio Speech Lang Process* 17(5):895–903
26. Hacioglu K, Pellom B, Ciloglu T, Ozturk O, Kurimo M, Creutz M (2003) On lexicon creation for Turkish LVCSR. In: Proceedings of Eurospeech '03, Geneva, pp 1165–1168
27. Heintz I (2010) Arabic language modeling with stem-derived morphemes for automatic speech recognition. PhD thesis, The Ohio State University
28. Hirsimäki T, Creutz M, Siivola V, Kurimo M, Virpioja S, Pylkkönen J (2006) Unlimited vocabulary speech recognition with morph language models applied to Finnish. *Comput Speech Lang* 20:515–541
29. Jelinek F, Mercer R (1980) Interpolated estimation of Markov source parameters from sparse data. In: Proceedings of the workshop on pattern recognition in practice, Amsterdam, pp 381–397
30. Katz SM (1987) Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Trans Acoust Speech Signal Process ASSP-35(3)*:400–401

31. Khudanpur S, Wu J (2000) Maximum entropy techniques for exploiting syntactic, semantic, and collocational dependencies in language modeling. *Comput Speech Lang* 14:355–372
32. Kirchhoff K, Bilmes J, Henderson J, Schwartz R, Noamany M, Schone P, Ji G, Das S, Egan M, He F, Vergyri D, Liu D, Duta N (2002) Novel approaches to Arabic speech recognition: report from the 2002 Johns-Hopkins summer workshop. Technical report, Johns Hopkins University
33. Kirchhoff K, Vergyri D, Bilmes J, Duh K, Stolcke A (2006) Morphology-based language modeling for conversational Arabic speech recognition. *Comput Speech Lang* 20:589–608
34. Kneser R (1996) Statistical language modeling using a variable context length. In: Proceedings of the fourth international conference on speech and language processing, Philadelphia, pp 494–497
35. Kneser R, Ney H (1995) Improved back-off for m-gram language modeling. In: Proceedings of the IEEE international conference on acoustics, speech and signal processing, Detroit, vol 1, pp 181–184
36. Koehn P (2010) Statistical machine translation. Cambridge University Press, Cambridge
37. Kuo HK, Arisoy E, Mangu L, Saon G (2011) Minimum Bayes risk discriminative language models for Arabic speech recognition. In: IEEE workshop on ASRU, Waikoloa, pp 208–213
38. Kuo HKJ, Mangu L, Emami A, Zitouni I (2010) Morphological and syntactic features for Arabic speech recognition. In: Proceedings of ICASSP, Dallas, pp 5190–5193
39. Lee YS, Papineni K, Roukos S, Emam O, Hassan H (2003) Language model based Arabic word segmentation. In: Proceedings of the 41st annual meeting for ACL, Sapporo, vol 1, pp 399–406
40. Mackay DJC, Bauman Peto LC (1995) A hierarchical Dirichlet language model. *Nat Lang Eng* 1(3):289–307
41. Malouf R (2002) A comparison of algorithms for maximum entropy parameter estimation. In: Proceedings of Co-NLL, Taipei
42. Marton Y, el Kholy A, Habash N (2011) Filtering antonymous, trend-contrasting, and polarity-dissimilar distributional paraphrases for improving statistical machine translation. Proceedings of 6th workshop on statistical machine translation, Edinburgh, pp 237–249
43. Netzer Y, Adler M, Elhadad M (2008) Word prediction in Hebrew – preliminary and surprising results. In: ISAAC 2008, Gold Coast. <http://www.cs.bgu.ac.il/~adlerm>
44. Ney H, Essen U (1991) On smoothing techniques for bigram-based natural language modelling. In: Proceedings of the IEEE international conference on acoustics, speech and signal processing '91, Toronto, vol 2, pp 825–829
45. Ney H, Essen U, Kneser R (1994) On structuring probabilistic dependences in stochastic language modelling. *Comput Speech Lang* 8:1–38
46. Nguyen L, Ng T, Nguyen K, Zbib R, Makhoul J (2009) Lexical and phonetic modeling for Arabic automatic speech recognition. In: INTERSPEECH 2009, Brighton, pp 712–715
47. Niesler TR, Woodland P (1996) A variable-length category-based n-gram language model. In: Proceedings of ICASSP, Atlanta
48. Pellegrini T, Lamel L (2006) Investigating automatic decomposition for ASR in less represented languages. In: INTERSPEECH-2006, Pittsburgh, pp 1776–1779
49. Pellegrini T, Lamel L (2007) Using phonetic features in unsupervised word decompounding for ASR with application to a less-represented language. In: INTERSPEECH-2007, Antwerp, pp 1797–1800
50. Roark B, Saraclar M, Collins M (2007) Discriminative *n*-gram language modeling. *Comput Speech Lang* 21:373–392
51. Ron D, Singer Y, Tishby N (1996) The power of amnesia: learning probabilistic automata with variable memory length. *Mach Learn* 25:117–149
52. Rosenfeld R (1996) A maximum entropy approach to adaptive statistical language modelling. *Comput Speech Lang* 10:187–228
53. Sarikaya R, Deng Y (2007) Joint morphological-lexical language modeling for machine translation. In: NAACL-Short '07 human language technologies 2007, Rochester, pp 145–148
54. Sarikaya R, Afify M, Gao Y (2007) Joint morphological-lexical language modeling (JMLLM) for Arabic. In: Proceedings of ICASSP, Honolulu, pp 181–184
55. Schwenk H (2007) Continuous space language models. *Comput Speech Lang* 21(3):492–518

56. Sethy A, Chen S, Ramabhadran B (2011) Distributed training of large scale exponential language models. In: Proceedings of ICASSP, Prague, pp 5520–5523
57. Seymore K, Rosenfeld R (1996) Scalable backoff language models. In: Proceedings of ICSLP 1996, Philadelphia, pp 232–235
58. Shilon R, Habash N, Lavie A, Wintner S (2012) Machine translation between Hebrew and Arabic. *Mach Trans* 26:177–195
59. Siivila V, Kurimo M, Lagus K (2001) Large vocabulary statistical language modeling for continuous speech recognition in Finnish. In: Proceedings of the 7th European conference on speech communication and technology, Copenhagen, pp 737–747
60. Siivila V, Creutz M, Kurimo M (2007) Morfessor and VariKN machine learning tools for speech and language technology. In: INTERSPEECH 2007, Antwerp, pp 1549–1552
61. Siivila V, Hirsimäki T, Virpioja S (2007) On growing and pruning Kneser-Ney smoothed N-gram models. *IEEE Trans Audio Speech Lang Process* 15(5):1617–1624
62. Stolcke A (1998) Entropy-based pruning of backoff language models. In: Proceedings of the DARPA broadcast news transcription and understanding workshop, Virginia, pp 270–274
63. Tachbelie MY, Menzel W (2007) Sub-word based language modeling for Amharic. In: Proceedings of international conference on recent advances in NLP, Borovets, pp 564–571
64. Tachbelie MY, Abate ST, Menzel W (2009) Morpheme-based language modeling for Amharic speech recognition. In: The 4th language and technology conference, Poznan
65. Vergyri D, Kirchhoff K, Duh K, Stolcke A (2004) Morphology-based language modeling for Arabic speech recognition. In: Proceedings of ICSLP, Jeju Island, pp 2245–2248
66. Wallach HM (2008) Structured topic models for language. PhD thesis, University of Cambridge
67. Witten I, Bell T (1991) The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. *IEEE Trans Inf Theory* 37:1085–1094
68. Xiang B, Nguyen K, Nguyen L, Schwartz R, Makhoul J (2006) Morphological decomposition for Arabic broadcast news transcription. In: Proceedings of ICASSP, Toulouse, pp 1089–1092
69. Yuret D, Biçici E (2009) Modeling morphologically rich languages using split words and unstructured dependencies. In: Proceedings of the ACL-IJCNLP 2009 conference short papers, Singapore, pp 345–348
70. Zitouni I, Zhou Q (2007) Linearly interpolated hierarchical n-gram language models for speech recognition engines. In: Robust speech recognition and understanding. I-Tech, Vienna, pp 301–318

**Part II**

**Natural Language Processing Applications**

# Chapter 6

## Statistical Machine Translation

Hany Hassan and Kareem Darwish

### 6.1 Introduction

Machine translation (MT) is the process of using computers to automate the translation of text from one natural language into another. MT of text has been an active area of research for the past few decades. In recent years, MT research has expanded to address the automated translation of spoken language. Funding agencies around the world and commercial entities have provided outstanding support for MT research, leading to major advances in the quality of MT and making MT a mainstream application. A clear indication of this success is the availability of a number of good MT online translation systems such as Google Translate and Bing Translator.

In general, there are many aspects that make MT a challenging problem. Some of these problems include picking appropriate translations of polysemous words; appropriately transliterating or translating named entities; constructing sentences from translated words or phrases; adapting translation systems to new domains, genres, and variations in language; etc. As a matter of fact, developing machine translation systems depends on many other natural language processing components and data such as word tokenization, morphological analysis, parsing, parallel data and monolingual data.

The family of Semitic languages includes many languages such as Arabic, Hebrew, Amaharic, Aamaic, Maltese, and Syriac. Some of these languages have different dialects that have unique linguistic features that are divergent from

---

H. Hassan (✉)

Microsoft Research, Redmond, WA, USA

e-mail: [hanyh@microsoft.com](mailto:hanyh@microsoft.com)

K. Darwish

QCRI, Doha, Qatar

e-mail: [kdarwish@qf.org.qa](mailto:kdarwish@qf.org.qa)

the original language. Semitic languages have orthographic and morphological characteristics that further complicate MT. Apart from Arabic (and to a lesser extent Hebrew), MT between Semitic languages and other languages have generally been understudied by the research community. The development of machine translation systems depends on many natural language processing components and data such as word tokenization, morphological analysis, parsing, parallel data and monolingual data. Most of the Semitic languages can be categorized among low resources languages which limits the approaches that can be applied for machine translation of Semitic languages.

Arabic has received the most attention with dedicated projects and evaluation campaigns. Although the characteristics of Arabic are similar to other Semitic language, the language resources and research done for Arabic is not matched by that for any of the other Semitic languages. Hebrew has started receiving some attention, but it continues to be quite limited compared to Arabic. Thus we focus in this chapter exclusively on MT for Arabic and Hebrew as examples of Semitic languages.

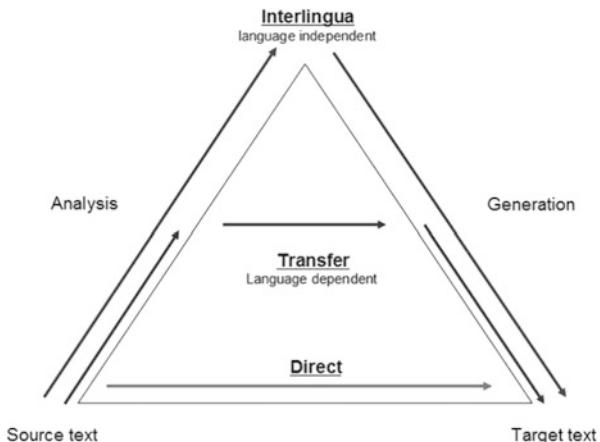
This chapter is organized as follows: Sect. 6.2 presents an overview of machine translation approaches; Sect. 6.3 introduces an overview of statistical machine translations approaches; Sect. 6.4 reviews the machine translation evaluation metrics. Section 6.5 discusses the special consideration that should be taken into account when developing SMT systems for Semitic languages. Section 6.6 discusses how to build an SMT system; then the required software resources are introduced in Sect. 6.7. A step-by-step guide for building SMT systems is presented in Sect. 6.8. Finally, Sect. 6.9 summarizes the chapter and concludes.

## 6.2 Machine Translation Approaches

### 6.2.1 *Machine Translation Paradigms*

There are many paradigms for performing MT. The classical architecture of MT systems follows the famous Vauquois triangle shown in Fig. 6.1. This representation proposes that there are three main paradigms for MT, namely the direct approach, the transfer approach, and the interlingua approach. In the direct translation approach, the translation is performed word by word where a source word is mapped directly to a target word possibly with some morphological analysis. This approach highly depends on bilingual lexicons to perform the translation. However, the direct approach is too focused on individual words and lacks knowledge of phrasal constructs.

In the transfer approach, the source language is analyzed by a parser then some transfer rules are used to transfer the analysis into the target language followed by a generation step to generate the target language. The analysis, transfer and generation components are all language dependent and have to be designed and developed

**Fig. 6.1** Vauquois triangle

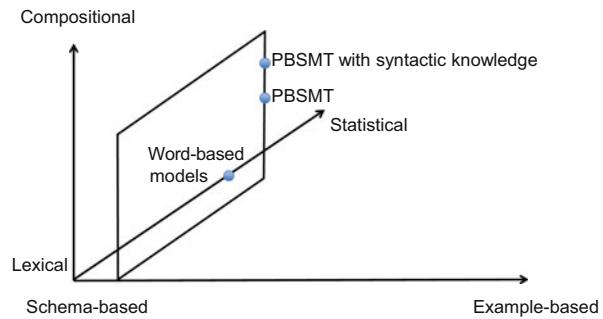
according to the corresponding language. It is worth noting that some large-scale rule-based MT systems like Systran are mostly based on the transfer approach.

In the interlingua approach, the meaning of the source language is represented through an intermediate language (interlingua) then generation rules are used on the unified representation. The interlingua approach is based on a theoretical representation and has not yet been applied on a large scale.

Though the Vauquois triangle classical architecture helps in abstracting the basic approaches and various processes that might be used in performing MT, MT systems rarely adhere to this claimed theoretical framework due to the compromised solutions assumed during systems development.

A more recent representation was proposed by D. Wu in [20], in which he presented a three-dimensional MT model space that focused on the approaches deployed to achieve the translation rather than on the process of performing the translation. Wu's 3D-representation consists of three dimensions: statistical versus logical, compositional versus lexical, and example-based versus schema-based. He defined statistical machine translation (SMT) as an MT system that makes nontrivial use of statistics and probability while the logical system makes extensive use of logical rules. Compositional MT uses compositional transfer transduction rules while lexical MT uses lexical transfer without compositional rules. Finally, example-based MT uses a large library of examples at translation runtime while schema-based MT uses abstract schemata during runtime. Figure 6.2 shows the projection of different SMT systems in this three-dimensional model. Word-based SMT models represent the statistical and lexical combination, while phrase-based SMT systems deploy more collocational information and therefore move away from the lexical towards the compositional dimension. As more syntactic knowledge is added into phrase-based SMT, the system is pushed further into the compositional dimension.

**Fig. 6.2** Various SMT models projected on the three-dimensional model space from [20]



In the following, we present the most popular MT models, namely rule-based MT, example-based MT, and phrase-based SMT models. Special attention is devoted to phrase-based SMT, because it is currently the most dominant approach to MT.

### 6.2.2 Rule-Based Machine Translation

Usually a rule-based MT system follows a transfer approach composed of three stages: analysis, transfer and generation. The analysis stage parses the source language text into an abstract language-specific syntactic structure regardless of the target language. In the transfer stage, linguistic rules specific to the language pair transform this representation into an equivalent representation specific to the target language. Differences between languages, in vocabulary and structure, are handled in the intermediary transfer program. In the third stage, the final target language text is generated. Again, this approach involves a component tailored for a specific source-target language pair and bilingual lexicons that connects the lexical units of the source and target languages.

The transfer rules and the bilingual lexicons are typically expensive to build and maintain. Another disadvantage is that, in the transfer approach, some ambiguities in source and target languages are hard to discern and hence difficult to overcome in such systems. Recently, some rule-based systems deployed a language model to disambiguate the final target translation.

### 6.2.3 Example-Based Machine Translation

The example-based machine translation (EBMT) approach is often characterized by its use of a bilingual corpus with parallel texts as its main knowledge base, at run-time. Considering the process of human translation, the idea is that translation takes place by analogy as opposed to the idea that people translate sentences by doing deep linguistic analysis. EBMT is founded on the belief that people translate

primarily by decomposing a sentence into certain phrases, then translating these phrases, and finally properly composing these fragments into one long sentence. Phrasal translations are translated by analogy to previous translations.

The principle of translation by analogy is encoded to EBMT through the example translations that are used to train such a system. The EBMT consists of: (a) a matcher that tries to find the largest phrase in the input sentence that matches a phrase in the example base; (b) an identification module that tries to find the translation of the matched phrase in the examples base; and (c) a recombination module that tries to combine fragments into one sentence similar to a decoding problem in SMT.

#### 6.2.4 Statistical Machine Translation

SMT is an MT paradigm where translations are generated using statistical models whose parameters are derived from the analysis of bilingual and monolingual text corpora. The underlying idea of SMT is motivated by information theory [19], where Weaver proposed that the statistical techniques from information theory and cryptography might make it possible to use computers to translate text from one natural language to another. Four decades later, in the late 1980s, a group of IBM researchers revisited the idea of using statistical techniques for translation. They were encouraged by the increase in computing power, the availability of large-scale parallel corpora, and the lack of progress by other methods.

Brown et al. in [3] and [4] formulated the MT problem as a noisy channel model, which has led to the rise of SMT. The noisy channel model assumes that a piece of text in some source language is transmitted over a “noisy channel” that deforms the text to produce the target language. Thus if we can learn the deformations that occur in the channel (i.e. learning  $p(\text{target}|\text{source})$ ), which can be approximated by products of probabilities of word or phrase translations, and the probability of the source text, which can be approximated using a language model, then we would be able to recover the source text from the target text.

#### 6.2.5 Machine Translation for Semitic Languages

SMT is now by far the most dominant paradigm of MT for many reasons, namely: accuracy, scalability, and rapid adaptation to new languages and domains. Moreover, it requires only parallel data and no hand-crafted rules over source and target languages as required by rule-based systems. SMT has a great advantage over example-based systems in the way it generalizes over the training data.

We think SMT is the most suitable paradigm for developing machine translation systems for Semitic languages for many reasons. First, SMT provides accurate and scalable systems, second it requires only parallel data and target monolingual

data. Third, it requires only shallow analysis and word segmentation with no need for more sophisticated syntactic analysis which is not available for most Semitic languages. Finally, Semitic languages share similarities on different levels of language elements (i.e. lexical, morphological and syntactic) which makes SMT very appealing since SMT techniques usually perform much better when translating between similar languages.

In the next section we will introduce a brief overview of the SMT system as it is by far the most dominant paradigm in machine translation nowadays and represents the most promising approach for developing a new SMT system for any languages pair.

## 6.3 Overview of Statistical Machine Translation

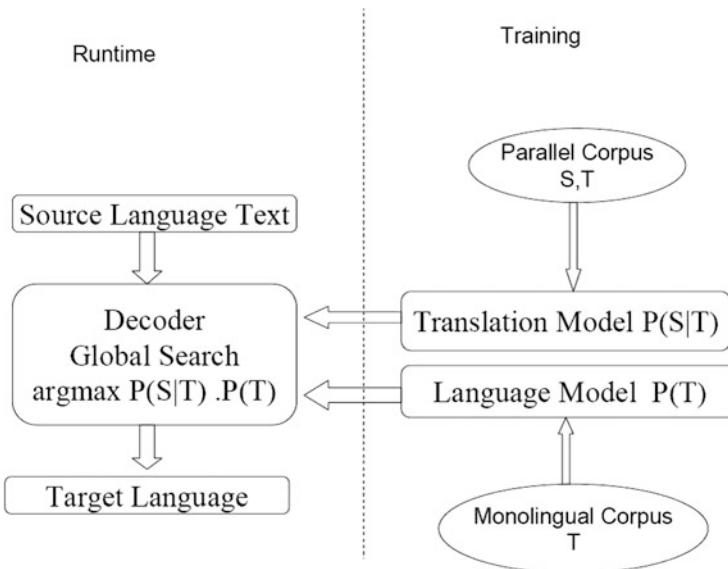
The work introduced in [3] and [4] proposed that the problem of MT can be handled as a noisy channel model. A target sentence T is transferred to a source sentence S when going through a noisy channel. If this noisy channel could be modeled, then translation from S to T could be achieved. The machine translation decoder reverses the noisy channel by reproducing the target sentence T from the source sentence S. Figure 6.3 demonstrates the SMT system in the training and decoding phases. At training time, the system uses a parallel corpus to estimate the translation model probabilities, which attempts to find the most appropriate translation, and a monolingual corpus to estimate the target language model probabilities, which attempts to find the most fluent rendering of the translation. At decoding time, the two probabilistic components are utilized within a global search technique to find the best translation for a given source sentence. The translation model can take various alignment forms such as word-based, phrase-based and syntax-based forms. The language model can be an n-gram language model, a syntax-based language model or any other model that measures how fluent the target language output is.

In the following sections we will review two translation models, namely: word-based and phrase-based models.

### 6.3.1 Word-Based Translation Models

In word-based translation models, the translation model is simply a word-to-word probabilistic translation model, which is typically estimated from word-level alignments that represent a mapping between source and target words in parallel sentence pairs. Word alignment is crucial for SMT as the accuracy of the translation component is highly dependent on it.

There are five alignment models that have been proposed in [4] with different complexities. They are known as IBM Models 1, 2, 3, 4, and 5. In all IBM alignment models, a source word can be linked to exactly one target word, thus these alignment models do not allow many-to-one, one-to-many, or many-to-many alignments.



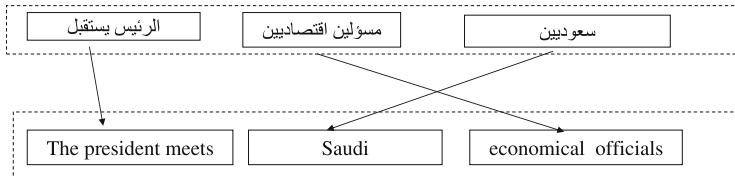
**Fig. 6.3** Phrase-based SMT system

IBM Model-1 is the simplest amongst them and aims to learn the word (lexical) translation model using the alignment links. If we already know the alignment links, we can estimate the lexical translation model by collecting counts and performing maximum likelihood estimation. On the other hand, if we have the translation model, we can assume the most likely alignment links. The problem is that we do not have either of them. This is a well-known problem with efficient solutions. The problem involves learning a model from incomplete data with hidden variables underlying the model. In the case of Model-1, we are trying to estimate the translation probabilities while the alignment links are the hidden variables in this problem. IBM Model-1 uses the Expectation Maximization (EM) algorithm to solve this problem. For detailed information about the mathematical formalization of IBM models, the reader is referred to [4]. The models are implemented in the widely used toolkit GIZA++.<sup>1</sup>

### 6.3.2 *Phrase-Based SMT*

Word-based SMT models have a major disadvantage, namely that they do not use any contextual information for estimating the translation probability. If the translation unit is larger than a single word, capturing the context would help

<sup>1</sup><http://www.fjoch.com/GIZA++.html>



**Fig. 6.4** Sentence pair with phrase boundaries and reordering

produce better translations. Capturing context would help with local reordering of words, such as noun-adjective reordering between different languages. Phrase-based SMT has been proposed to overcome these problems where the unit of translation is any sequence of adjacent words. As shown in Fig. 6.4, a phrase-based SMT system starts by segmenting the source sentence into phrases with arbitrary boundaries then translates the source phrases into target phrases and finally performs reordering if applicable. As shown in Fig. 6.4, the phrases of phrase-based SMT are not linguistically motivated and do not necessarily relate to any constituent phrase structure. In fact, these phrases are just any arbitrary sequence of words chosen according to the word alignment probabilities. The current paradigms of phrase-based SMT were proposed by different research groups (e.g. IBM, RWTH-Aachen and ISI-USC), with more similarities than differences between the various approaches. The reader is referred to [9] which provides a comprehensive review of statistical machine translation. In this section, we will focus on the commonly used techniques in the research community.

### 6.3.3 *Phrase Extraction Techniques*

IBM word alignment models provide word-to-word mapping where a source word can be aligned to exactly one target word. These alignment models do not allow for many-to-one or many-to-many alignments and so the alignments are asymmetric, i.e. the links of the alignment are not the same if the source and target languages are swapped. Some ad-hoc approaches have been proposed for extracting phrase mappings based on intersection and union of word alignments. Och and Ney [14] proposed an approach for extracting phrase mappings based on producing symmetrized alignments from word-based alignments and then using some heuristics to extract phrase pairs. First, alignments in both directions (target-source and source-target) are produced. Both alignments are intersected to produce a high-precision alignment. The union of the two alignments is used to extend the intersection with more alignment points using some heuristics such as GROW-DIAGONAL which examines all the neighboring alignment points of the intersections. If the neighboring words are not in the intersection and if both their source and target words are in the union, then the alignments are extended with the union words. Finally, phrase pairs are extracted from those extended alignments.

### 6.3.4 SMT Reordering

Reordering defines how far the target phrase should move during translation. Generally, the reordering models penalize any movement in the target translation away from the corresponding source position and depend on the language model to judge the appropriateness of a movement. As shown in Fig. 6.4, phrases can be translated monotonically, swapped or move further.

The basic reordering model is a linear reordering model that simply skips a number of source words/phrases to allow the movement of the target translation with a particular penalty. However, this simple orientation model does not depend on the actual phrase itself but on the relative position between reordered phrases. More recently, a number of sophisticated reordering approaches have been proposed such as lexicalized reordering models [18]; and hierarchical lexicalized reordering models [8].

These approaches focus on lexicalized reordering models which model the reordering based on the phrase itself not on the relative position as before. For example, the model can provide a probability for each phrase in a give context to be translated in monotone, swapped with the neighboring phrases or translated as a discontinuous phrase and moved further.

We think that the models presented above are satisfactory for modeling how to penalize the movement of the phrases; however, they depend on the language model to judge the grammaticality of the translation output with this movement. We think that the n-gram language models limit the capability of reordering models since an n-gram language model cannot judge the grammaticality of a movement beyond the n-gram scope. It is worth noting that the effect of reordering techniques is highly related to the difference in structure orders between the two languages. For example, if we are translating between Arabic and English; Arabic has SVO and VSO orders while English has SVO only. This would lead to many reordering variations and requires a sophisticated reordering technique. On the other hand, if we are translating between two languages with similar structures like French and Italian, then it will require less sophisticated reordering techniques to get good translation.

### 6.3.5 Language Modeling

The language model is a very crucial component for the machine translation performance. In MT, the main role of the language model is to judge the goodness of the candidate translation strings. The most commonly used language models in SMT are the n-gram language models, which are trained on unlabeled monolingual text. Generally, more data tends to yield better language models and better translation quality. Statistical language models have been designed to assign probabilities to strings of words where approximations are made to limit the window of the n-gram language model following the Markov assumption that only the most recent ( $n - 1$ )

Maria	no	dio una bofetada	a	la	bruje	verde
Mary	not	give a slap	to the	witch	green	
did not		a slap	by		green witch	
no		slap	to the			
did not give			to			
			the			
		slap		the witch		

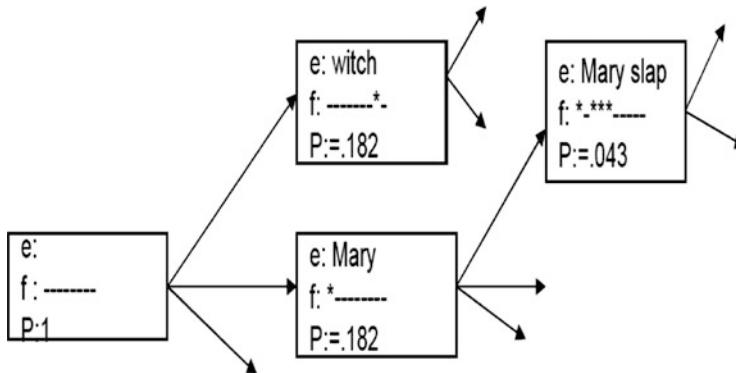
**Fig. 6.5** All possible source segmentations with all possible target translations [10]

words are relevant when predicting the next word. The reader is referred to [9] for more details on language modeling.

### 6.3.6 SMT Decoding

The task of the phrase-based SMT decoder is to search for the best translation given a source sentence, i.e. to maximize the probability of the target sentence as formalized in the SMT model. Generally, SMT decoders deploy a beam search decoder. The decoding starts by searching the phrase table for all possible translations for all possible fragments of the given source sentence. All possible source segmentations with all possible target translations are considered as shown in Fig. 6.5. The reordering is performed according to any of the approaches discussed above. Each hypothesis is expanded when considering a new translation as shown in Fig. 6.6.

The size of the search space increases exponentially due to the reordering and the large number of translation candidates; even decoding a word-based model with a bigram language model is an NP-complete problem. Some strategies have to be used to limit the exponential explosion of the search space; therefore, a beam search pruning strategy is used to prune those hypotheses having a high cost and thus reduce the search space. Moreover, similar hypotheses are combined to reduce further the search space, if they cover the same source words and share the same language model history. The decoder calculates a future cost estimation for the



**Fig. 6.6** Expanding the decoder hypothesis with possible translations [10]

uncovered parts of the source sentence and at each hypothesis the future cost is estimated based on the translation cost and the language model cost of the uncovered source words. The total cost of the hypothesis is the sum of the actual cost and the future cost. Thus, the total cost can be a good estimation of the complete hypothesis cost. The decoder keeps a number of stacks to keep all partial translations of the target sentence, and the beam search pruning is applied to all such stacks to keep the most likely hypotheses. Finally the hypothesis that covers all source words with the lowest cost is chosen as the most likely translation.

## 6.4 Machine Translation Evaluation Metrics

Evaluation of MT output is a very hard task due to the subjectivity of evaluation. Automated evaluation schemes attempt to measure the quality in a way that correlates with human evaluation. The mostly widely adopted automated evaluation measure is the so-called Bilingual Evaluation Understudy (BLEU) [15]. BLEU measures the translation quality by calculating the geometric mean of n-gram agreements between the output translation and one or more reference translations. To account for word deletion and to penalize translations with high precision but low recall, the BLEU score includes a brevity penalty factor that penalizes translations shorter than the references.

Many variations have been proposed to extend the BLEU score. For example, in METEOR the focus is on recall by incorporating the use of stemming and synonyms from Wordnet to match similar target variations. More recently, an extension of the BLEU score to measure the dependency relations between the translation and the references was proposed.

Nonetheless, automatic evaluation of MT remains a highly controversial issue due to the lack of an acceptable measure that can capture translation variations. More recently, human evaluation such Human Translation Error Rate (HTER) was

used in large-scale evaluations. HTER is a human-based version of the translation error rate metric, where a human calculates the minimum number of insertions, deletions and substitutions needed to correct the translation output according to some guidelines. While it is gaining acceptance, it is not available for everyday tasks for all researchers; BLEU is usually used to serve this purpose.

## 6.5 Machine Translation for Semitic Languages

There are several aspects that would affect the translation quality when translating to and from a Semitic language. The most notable of these aspects are word segmentation, word alignment and reordering models, and target side gender-number agreement. We will discuss each aspect in more detail here.

### 6.5.1 Word Segmentation

The main objective of word segmentation for machine translation is not to provide morphological analysis to the source or target text, but rather to make source and target sentence lengths as similar as possible (in the number of tokens). Another important goal of segmentation has to do with improving the coverage of the translation lexicon. Since languages such as Arabic and Hebrew attach clitics like prepositions, pronouns, determiners, and coordinating conjunctions to words, the number of unique surface forms is very large. Thus segmentation is critical for reducing the number of translatable units.

For example the Arabic word “wsynqlhm” can be translated to the English phrase “and he will transfer them”. In this case it may be better to segment the Arabic word to get better alignment and better coverage. We have many choices for segmenting the Arabic word such as “w s ynql hm”, “w s ynqlhm” and “w synglhm”.

The segmentation criterion is highly dependent on source and target languages. A number of papers discussed the effect of segmentation schemas for Arabic-English SMT. For example A. El-Kholi and N. Habash in [6] discussed the effect of Arabic word segmentation on translating from a Semitic language (Arabic) to a Latin language (English). The study highlighted the important effect of word segmentation on translation quality. H. Al-Haj and A. Lavie in [2] discussed the effect of Arabic word segmentation when translating from a Latin language (English) to a Semitic language (Arabic). Both studies highlight different aspects related to the direction, to/from a Semitic language, and the best segmentation is dependent on the direction of the translation.

In general, when translating from a Semitic language to a non-Semitic language, segmentation should produce segments that match the words in the target language. From the example above, segmenting “wsynqlhm” into “w + s + ynql + hm” would match the corresponding English words as (w → and; s → will; ynql → he transfer; hm → them). Over-segmenting or under-segmenting can adversely affect phrasal translation and reordering.

Meaning of source	Arabic	Segmented Arabic	Segmented Hebrew	Hebrew	Matches
and the family	والعائلَة	و+ال+عائِلَة	ו+ה+מִשְׁפָחָה	משפחָה	ו → ו (and) ال → ה (the) عائِلَة → משפחָה (family)
your brother	أخِيك	أخِي+ك	אֶחָד+ך	אֶחָד	أخِي → אֶחָד (brother) ك → ד (your)

Fig. 6.7 Arabic-Hebrew segmentation

On the other hand, translating from a non-Semitic language, particularly English, to a Semitic language introduces a new challenge, namely the produced segments need to be combined in the proper order. In the example above, translating “and he will move them” would generate “w + hw + s + ynql + hm” out of which “hw”, which corresponds to “he” would have to be dropped. Thus, over-segmentation can greatly impact the ability to reconstruct words.

Translating between two Semitic languages would alleviate some of these challenges as many Semitic languages share similar morphological structures. Figure 6.7 illustrates the segmentation similarities between Arabic and Hebrew. The examples show that Arabic and Hebrew shares a lot of segmentation similarities which would ease the design of word segmentation for translating between them.

It is worth noting that the word segmentation is a very crucial aspect for getting a good translation performance; this aspect is more important with Semitic languages than other languages because of its rich morphological structures.

### 6.5.2 Word Alignment and Reordering

Different syntactic structures between various languages introduce a challenge for word alignment techniques; the more monotonic the nature of the two languages the easier the alignment will be. For example, word alignment between English and French is much easier than that of English and Arabic since Arabic has SVO and VSO orders while English has the former only.

Semitic languages such as Arabic and Hebrew may allow similar syntactic structures [16]. Though Hebrew is mostly SVO, it may allow VSO order. Conversely, Arabic is mostly VSO, but it freely allows the SVO order as well. As for noun-phrases, the word order is the same for both Arabic and Hebrew where the adjective trails the noun. For example, consider the noun phrase that is equivalent to “the old city” in Hebrew and in Arabic; the mapping between Hebrew and Arabic is monotonic in this case as shown in Fig. 6.8.

Notice that the adjective and the noun are both preceded by determiners, which is unlike English where the noun phrase has only one determiner. Also both Arabic and Hebrew allow for other syntactic constructs such as:

**Fig. 6.8** Arabic-Hebrew reordering

העיר העתיקה = <المدينة القديمة> ה - = <ה -> עיר = <עיר> ה - = <ה -> עתיקה = <עתיקה>	אל = <المدينة> - the مدينة = <עיר> אל = <ה -> קדימה = <עתיקה>
---	---

- Genitive constructs called idafa and smikhut in Arabic and Hebrew respectively.
- Nominal sentences that don't have verbs.
- In some cases, if the subject is a pronoun, it may be dropped.

Thus local alignment between Arabic and Hebrew can be monotonic, but sentence level alignment may not be due to the difference in order. A fuller treatment of syntactic differences between Arabic and Hebrew and how they reflect on MT can be found in [16].

### 6.5.3 Gender-Number Agreement

Translating into Semitic languages would face another challenge, which is the agreement between gender and number in the output of the machine translation system. This problem is very clear when translating into a Semitic language.

Arabic and Hebrew nouns have an associated gender. For example, a “book” in Arabic (ktAb) and Hebrew (sfr) is masculine, while a “pigeon” is feminine in Arabic (ymAmp) and Hebrew (ywnh). English on the other hand does not assign gender to words. Other Latin languages, such as French, assign gender to nouns, but they may not match the assignments of Arabic or Hebrew. The gender of nouns reflects on:

- Adjectives as they must agree in gender with the noun they modify. For example, “large” modifies a chair in Arabic as (kbyr), while “large” modifies a refrigerator as (kbyrp) because they are masculine and feminine respectively.
- Verbs need to agree with the object (in Arabic) when passive tense is used. For example, the equivalent Arabic verb in “the ball was played” and “the match was played” is “lEb” and “lEb” respectively.

In Arabic and Hebrew, verbs need to agree in number with the subject and adjectives need to agree in number with the noun they modify. For example, the verb in “the boy reads” and “the boys read” in Arabic is “yktb” and “yktbwn” respectively (only in SVO order – in VSO order, the verb is “yktb” for both), and in Hebrew is “kwtb” and “kwtbym” respectively. For an example of adjective noun number agreement, the adjective in “the small book” and “the small books” in Arabic is “sgyr” and “sgyrp” respectively, and in Hebrew is “qtn” and “qtnym” respectively.

## 6.6 Building Phrase-Based SMT Systems

### 6.6.1 Data

The success of any SMT system highly depends on the availability of a reasonable amount of high-quality parallel data as well as high-quality monolingual data. For some languages such as Arabic, English, French and Chinese, there are large sets of parallel training data such as the data available from the United Nations. For many other languages, there are limited amounts of parallel data or nearly no parallel data.

For Arabic-English MT there is a fair amount of training data. For Hebrew-English MT, there is limited training data. Unfortunately, there is very limited training data for Arabic-Hebrew MT, mostly from the Quran and the Bible. For other Semitic languages, MT parallel data between them and other languages is scant at best. As for monolingual data, it is critical for building good language models and usually easier to get for many languages. In the remainder of this section we will briefly discuss the specifications and the amount of parallel and monolingual data required to train an SMT system.

### 6.6.2 Parallel Data

The SMT technology is mature and well documented in a large number of papers and tutorials. Moreover, the open source systems are very competitive and provide good capability for starting a machine translation project for any language pair. The quality and richness of parallel data is one of the strongest differentiating factors between a good and a bad SMT system. However, preparing high-quality parallel data is not a trivial task. The translation should be fluent while preserving all of the meaning present in the original text. The translation must be faithful to the original text in terms of both meaning and style. The translation should contain the exact meaning conveyed in the source text, and should neither add nor delete information. No words or phrases should be added to the translation as an explanation or aid to understanding. Special attention should be paid to name translation, which should be translated or transliterated depending on the common convention in the target language. The Linguistic Data Consortium (LDC) has provided some guidelines for producing parallel data with high quality. These guidelines can be obtained from LDC website.<sup>2</sup>

A very important question for SMT practitioners is: how much training data is required to get a good translation quality. The simple answer is that more in domain data would enhance the quality of an SMT system. Specifying the exact amount of data is a very challenging and is highly dependent on the languages pair and the domain of the translation.

---

<sup>2</sup><http://www.ldc.upenn.edu/Catalog/docs/LDC2012T06>

### ***6.6.3 Monolingual Data***

Very large amounts of monolingual data are required for training good language models. Fortunately, huge amounts of Arabic data are already available in the news domain via the LDC Arabic Giga Corpus.<sup>3</sup> More data can be easily obtained by mining the web. Similar to the parallel data, domain-specific monolingual data will have a more positive effect on the translation accuracy than general domain data.

## **6.7 SMT Software Resources**

### ***6.7.1 SMT Moses Framework***

Moses is the most popular open source implementation of the state-of-the-art phrase-based SMT engine. Moses provides a framework for training, tuning, and decoding translation models for any language pair. Moses provides fast and efficient search techniques that quickly find the most probable translation among a large number of possible translations. Moses provides a factored representation that might be helpful in representing highly inflected morphological languages like Arabic and Hebrew. For example, factored representation allows for separate translation tables for stem-to-stem translation followed by a generation step that composes the final word from stems and associated prefix-suffix combinations.

From an engineering point of view, Moses is very well designed and easy to extend to add more features to the system. Moses has a very active development community that keep adding more features. The reader is referred to the user guide for a comprehensive guide to the Moses Framework [13].

### ***6.7.2 Language Modeling Toolkits***

SRILM, which was developed by SRI International, is one of the most popular toolkits designed to allow both production and experimentation with statistical language models for various applications. SRILM is written in C++. It is widely used in the NLP community and it is integrated with Moses. It produces n-gram word counts in the so-called ARPA format, which is a de-facto standard in the speech and NLP communities, and in binary indexed format, which provides a compact representation that enables efficient utilization of the language model. The toolkit can be found at: <http://www.speech.sri.com/projects/srilm/>.

IRSTLM is another popular language modeling toolkit, which is written in C++ and was developed by researchers at ITC-irst [7]. This toolkit also integrates with

---

<sup>3</sup><http://www.ldc.upenn.edu>

Moses. Like SRILM, it produces counts in ARPA format and in a binary indexed format. IRSTLM can be obtained from: <http://sourceforge.net/projects/irstlm/>.

RANDLM is yet another C++ toolkit that uses so-called randomized language models that have much lower memory requirements at the expense of being exact [17]. Other advantages to this toolkit are: it is multi-threaded; and it can be parallelized on a Hadoop cluster, which allows for the creation of very large language models. Again, it supports the popular ARPA format and produces a binary format. It can be obtained from: <http://sourceforge.net/projects/randlm/>.

Other notable language modeling toolkits are BerkleyLM (<http://code.google.com/p/berkeleymlm/>), KenLM (<http://kheafield.com/code/kenlm/>), HLMTools (<http://htk.eng.cam.ac.uk>), and CMU SLM toolkit (<http://www.speech.cs.cmu.edu/SLM/toolkit.html>).

### 6.7.3 *Morphological Analysis*

There are a number of freely available morphological analyzers that can be used for performing the aforementioned segmentation of words to improve MT. Some of the available tools for Arabic are QADEM (<http://www1.ccls.columbia.edu/cadim/>), MADA (<http://www1.ccls.columbia.edu/MADA/>), MorphTagger [11], and AMIRA [5] which provide useful tools for analyzing Arabic. MADA the most commonly used analyzer for the purposes of Arabic-English MT, but there are some indications that MorphTagger [12] may be faster and slightly better for MT.

For Hebrew, there are a few available morphological analyzers such as [1] and MorphTagger [11].

## 6.8 Building a Phrase-Based SMT System: Step-by-Step Guide

In this section, we will describe a step-by-step scenario for building a Hebrew to Arabic SMT system.

### 6.8.1 *Machine Preparation*

We recommend to have a machine with Ubuntu distribution and then install the Ubuntu NLP package which includes Moses and all its components and makes it easier to install Moses. It can be downloaded from the Ubuntu NLP Repository.<sup>4</sup> Alternatively, for more advanced experiments where the user is expected to modify

---

<sup>4</sup><http://cl.naist.jp/eric-n/ubuntu-nlp/>

the source code of the various components, we would recommend following the Moses User Manual and Code Guide [13] for full details on how to build the framework. After installing the Ubuntu NLP repository or downloading and building the Moses framework yourself, we would recommend following instructions in the Moses User Guide to install and test a sample model.

### **6.8.2 *Data***

As we described before, each machine translation system requires parallel data and monolingual data. Usually parallel data is harder to get while monolingual data is easier to get and hence larger. It is good practice to include the target side parallel data in the monolingual data as well. As a rule of thumb, all the processing done on target data should be exactly the same on the monolingual data. At decoding time, all preprocessing done on the training data should be done the same way on the source side of the evaluation data. The Linguistics Data Consortium (LDC) provides multiple volumes containing parallel training data for Arabic and English as well as large monolingual data for a variety of languages including Arabic and English.

### **6.8.3 *Data Preprocessing***

- Simple tokenization: This simply performs white space tokenization to separate words from punctuation.
- Numerical classes: Numbers represent a challenge to SMT systems since on the one hand they are usually easy to translate though on the other hand they are sparse and can represent a large portion of the translation and language model vocabularies. As a general practice, SMT researchers used to classify numbers written in numerical form (i.e. Arabic numerals) into a pre-specified class used throughout the translation process and only replace it with its equivalent translation at the end of the translation process.
- Characters normalization: This step should normalize different forms of the same characters that might be written in different ways, like ‘hamza’ for Arabic. Similar to this is lower casing characters for cased languages either as source, as target or both.

### **6.8.4 *Words Segmentation***

Word segmentation is generally a challenging task for machine translation. Simply using a morphological analyzer would lead to suboptimal translation results even with a very good system. The words segmentation effect is crucial for the quality of machine translation as discussed in Sect. 6.5.1.

### 6.8.5 Language Model

Given parallel and monolingual data that are pre-processed as discussed before, we can build our language model on the monolingual data and target side of the parallel data. It is worth noting that we can have multiple language models. For example, we can have a general domain language model and a domain-specific language model. This would allow the parameter tuning to pick the most adequate parameters depending on our development test data. We recommend building the language model using SRILM, which is already included in the Ubuntu NLP Repository. Building a simple language model using SRILM can be done by issuing this command:

```
ngram-count -text CORPUS-FILE -lm SRILM-FILE
```

There are many advanced language modeling techniques; some are supported by SRILM and others are supported by IRSTLM.<sup>5</sup> We highly recommend reviewing the Moses Manual for other alternatives such as Class based LM, Chunk based LM and building huge Language Models as discussed above.

### 6.8.6 Translation Model

The Moses Framework provides simple tools to build translation models with a single script which can perform the training. This script goes through the following steps: prepares data, performs word alignment, calculates lexical translation table, extracts phrases, scores phrases, builds reordering models, and finally creates configuration files used for decoding. Invoking this training script on parallel Arabic-Hebrew data is done using the following command:

```
train-model.perl -root-dir . -f hr -e ar -corpus corpus/hr-ar
```

Moreover, Moses supports factored translation models which support the translation of different factors that can be combined at the end of translation to generate the target surface form. This feature is particularly important when translating to highly morphological languages like Semitic languages. Using factored translation models, we can design a translation system that translates different morphemes and then combine them at a generation step to produce the final translation form.

### 6.8.7 Parameter Tuning

The key to good translation performance is having a good phrasal translation table and a good language model. However, the performance can be improved by tuning

---

<sup>5</sup><http://sourceforge.net/projects/irstlm/>

the decoder parameters. The probability cost that is assigned to a translation is a weighted sum of probability costs of many models such as phrase translation, language model, reordering, and word penalty models. Those weights are identified by maximizing the translation performance on a given validation set. The most common methodology is Minimum Error Rate Training (MERT). This can be performed within the Moses Framework using:

```
mert-moses.pl -mertdir=PATH-TO-MERT-DIR input-text references decoder-executable decoder.
```

### **6.8.8 System Decoding**

Having built all the necessary models, we are now ready to decode and evaluate the trained system. It is very straightforward to run the decoder using the configuration file that has been produced during the training process using the command:

```
moses -f moses.ini
```

It is worth mentioning that developing an SMT system is a continuous process that keeps improving over time by iterating various experiments and ideas.

## **6.9 Summary**

In this chapter, we briefly introduced statistical machine translation for semitic language. We introduced an overview of machine translation approaches and discussed statistical machine translation in more detail. We discussed the special consideration that should be taken into account when developing SMT systems for Semitic languages. We discussed segmentation techniques for Semitic SMT; and finally we introduced a guide on how to build an SMT system using freely available resources. This chapter is not a complete guide by any means, and the reader is highly encouraged to read the Moses user guide carefully to get the required knowledge for developing sophisticated SMT systems.

## **References**

1. Adler M, Elhadad M (2006) An unsupervised morpheme-based HMM for Hebrew morphological disambiguation. In: Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the ACL, Sydney, pp 665–672
2. Al-Haj H, Lavie A (2012) The impact of Arabic morphological segmentation on broad-coverage English-to-Arabic statistical machine translation. Mach Transl 26:3–24
3. Brown P, Cocke J, Della-Pietra S, Jelinek F, Della-Pietra V, Lafferty J, Mercer R, Roossin P (1990) A statistical approach to machine translation. Comput Linguist 16(2):79–85

4. Brown P, Della-Pietra S, Della-Pietra V, Mercer R (1993) The mathematics of statistical machine translation: parameter estimation. *Comput Linguist* 19(2):263–311
5. Diab M (2009) Second generation tools (AMIRA 2.0) fast and robust tokenization, POS tagging, and base phrase chunking. In: MEDAR 2nd international conference on Arabic language resources and tools, Cairo
6. El-Kholi A, Habash N (2012) Orthographic and morphological processing for English-Arabic statistical machine translation. *Mach Transl* 26:25–45
7. Federico M, Bertoldi N, Cettolo M (2008) IRSTLM: an open source toolkit for handling large scale language models. In: Proceedings of interspeech, Brisbane
8. Galley M, Manning C (2008) A simple and effective hierarchical phrase reordering mode. In: Proceedings of EMNLP, Honolulu
9. Jurafsky D, Martin JH (2009) Speech and language processing: an introduction to natural language processing, speech recognition, and computational linguistics, 2nd edn. Prentice-Hall, Upper Saddle River
10. Koehn P (2004) Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In: Proceedings of 6th conference of the Association for Machine Translation in the Americas, AMTA, Washington, DC, pp 115–124
11. Mansour S (2010) MorphTagger: HMM-based Arabic segmentation for statistical machine translation. In: Proceedings of the 7th international workshop on spoken language translation, Paris
12. Mansour S, Sima'an K, Winter Y (2007) Smoothing a lexicon-based POS tagger for Arabic and Hebrew. In: Proceedings of the workshop on computational approaches to Semitic languages: common issues and resources, Semitic, Prague, pp 97–103
13. Moses Framework user guide. <http://www.statmt.org/moses/manual/manual.pdf>
14. Och F, Ney H (2003) A systematic comparison of various statistical alignment models. *Comput Linguist* 29:19–51
15. Papineni K, Roukos S, Ward T, Zhu W (2002) BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting of the Association for Computational Linguistics (ACL'02), Philadelphia, pp 311–318
16. Shilon R, Habash N, Lavie A, Wintner S (2010) Machine translation between Hebrew and Arabic: needs, challenges and preliminary solutions. In: Proceedings of AMTA 2010: the ninth conference of the Association for Machine Translation in the Americas, Denver
17. Talbot D, Osborne M (2007) Randomised language modelling for statistical machine translation. In: Proceedings of the 45th annual meeting of the Association of Computational Linguistics, Prague, pp 512–519
18. Tillmann C (2004) A unigram orientation model for statistical machine translation. In: Proceedings of human-language technology and North American Association of Computational Linguistics (HLT-NAACL), Boston, pp 101–104
19. Weaver W (1949) Translation. In: Locke W, Booth A (eds) *Machine translation of languages: fourteen essays*, vol 42. MIT, Cambridge, pp 15–23
20. Wu D (2005) MT model space: statistical vs. compositional vs. example-based machine translation. *Mach Transl J* 19:213–227

# Chapter 7

## Named Entity Recognition

Behrang Mohit

### 7.1 Introduction

Named entity recognition (NER) is the problem of locating and categorizing important nouns and proper nouns in a text. For example, in news stories names of persons, organizations and locations are typically important. In the following example, the highlighted named entities hold key information and are useful for language processing applications.

*Before joining UCB, Lisa North worked for Pegasus Books in North Berkeley.*

Named entity recognition plays an important role in applications such as Information Extraction, Question Answering and Machine Translation. For example, information about named entities such as *Lisa North* helps a machine translation system to avoid translating them erroneously word by word.

The NER task has been studied extensively for many languages [54] including Arabic and Hebrew. Throughout the past two decades, numerous systems and data resources have been developed for NER. Moreover, there has been several forums and evaluation programs focused on named entity recognition and other related tasks.

In this chapter, we review the general state of NER research, relevant challenges and the current state of the art works on Semitic NER. Specifically, we look into two case studies for Arabic and Hebrew named entity recognition. We also review Semitic NLP tasks which overlap with the named entity recognition. We close with an overview of the available resources for Semitic NER and some the open research questions.

---

B. Mohit (✉)  
Carnegie Mellon University in Qatar, Doha, Qatar  
e-mail: [behrang@cmu.edu](mailto:behrang@cmu.edu)

**Table 7.1** Sample NER output with the mention-level (SGML) and BIO and BIOLU representations

Representation	Example		
SGML	<PER>Dr. Doull</PER> from the <ORG>Royal College of Paediatrics</ORG> in <LOC>Wales</LOC> backed the <MIS>Fresh Start</MIS>.		
	Token	BIO	BIOLU
BIO & BIOLU	Dr.	B-PER	B-PER
	Doull	I-PER	L-PER
	from	O	O
	the	O	O
	Royal	B-ORG	B-ORG
	College	I-ORG	I-ORG
	of	I-ORG	I-ORG
	Paediatrics	I-ORG	L-ORG
	in	O	O
	Wales	B-LOC	U-LOC
	backed	O	O
	the	O	O
	Fresh	B-MIS	B-MIS
	Start	I-MIS	L-MIS
	.	O	O

## 7.2 The Named Entity Recognition Task

### 7.2.1 Definition

Named entities (NEs) are words or phrases which are named or categorized in a certain topic. They usually carry key information in a sentence which serve as important targets for most language processing systems. Accurate named entity recognition can be used as a useful source of information for different NLP applications. For example the performance of applications like Question Answering [69], Machine Translation [7] or Information Retrieval [39] has been improved by named entity information. Table 7.1 shows an example sentence annotated with the named entity information, using different representation schemes. The three intuitive classes of person (PER), location (LOC), organization (ORG) along with the loosely defined miscellaneous(MIS) class are used in most NER systems. These classes are mostly relevant to the news related corpora. For other domains, NER systems are expected to be trained and tested with other relevant class labels.

Table 7.1 also presents different representations of named entity annotation. Early NER approaches used the mention (chunk) level representation which annotated a named entity as a whole chunk [66]. As the task evolved into a statistical learning problem, the sequence labeling framework became the standard

approach [16, 49]. In sequence labeling, the entire sequence of tokens (usually the sentence) is labeled concurrently. The BIO labeling is a representation that is generally used for sequence labeling. In this representation, a token is seen to be at the **B**eginning or **I**nside or **O**utside a named entity. In the alternative BILOU representation, the **L** and **U** labels are used respectively for the **L**ast token of a multi-token entity and the **U**nit-length named entities.<sup>1</sup>

The scope of named entity recognition has evolved over the past couple of decades. Originally NER was limited to the extraction of news related proper nouns such as names of persons, organizations and locations. With the expansion of NLP in other domains, those few traditional named entity classes were not sufficient. For example, for an article about science or technology, the three traditional classes are not enough and other named entity classes need to be considered. Moreover, named entities should not be limited to proper nouns. In certain areas of studies such as nuclear physics, one might highlight terms such as *proton* or *uranium* as named entities.<sup>2</sup> Thus, despite the common focus on the person, location and organization classes one can say that *NER encompasses the extraction of all important entities in a given context.*

### 7.2.2 Challenges in Named Entity Recognition

Named entity recognition consists of the following two sub-problems: (1) recognition of named entity boundaries; (2) recognition of named entity categories (classes). These problems are usually (but not necessarily) addressed concurrently. Similar to most problems in language processing, there are ambiguities in the language which add to the challenge of the task. In the following, we present examples of ambiguities in both recognition and categorization of named entities. In the first sentence, there is an ambiguity in the recognition of the named entity *Reading* that can be confused as a gerund form of a verb or a proper noun (city name). In the second example, the ambiguity is in the named entity type; *Fox* can be interpreted either as a person, an organization or a non-named entity. Furthermore, *Washington* might refer to a person, location or organization (US. government).

- **Reading** is located between two major highways.  
 <LOC> **Reading** </LOC> is located between two major highways.
- **Fox** criticized **Washington**.  
 <ORG> **FOX** </ORG> criticized <ORG> **Washington** </ORG>.

---

<sup>1</sup>Ratinov and Roth [59] have shown that with a small linear expansion of the parameters, the BILOU representation results in a better NER performance.

<sup>2</sup>Temporal and numerical expressions are other examples named entities which are not proper nouns.

**Table 7.2** Examples of rules used to extract named entities

Pattern	“headquartered in <x>”
Known locations	<i>Nicaragua</i>
New locations	<i>San Miguel, Chapare region, San Miguel City</i>
Pattern	“to occupy <x>”
Known locations	<i>Nicaragua</i>
New locations	<i>San Sebastian neighborhood</i>

Most NER challenges lie in its heavily lexicalized and domain-dependent nature. Names take a large part of a language and are constantly evolving in different domains. In order to have a robust NER system for any given domain (e.g. tourism), we need labeled corpora and lexicons (e.g. names of monuments). Creating and updating such resources for various topics is an expensive task and requires linguistics and domain expertise. In the following we will review two frameworks of *rule-based* and *statistical* NER and will discuss their data requirements and robustness.

### 7.2.3 Rule-Based Named Entity Recognition

Early approaches to named entity recognition were primarily rule-based. Most rule-based systems used three major components: (1) a set of named entity extraction rules, (2) gazeteers<sup>3</sup> for different types of named entity classes, and (3) the extraction engine which applies the rules and the lexicons to the text. The rule set and the lexicons were either completely handcrafted by humans or were bootstrapped from a few hand-crafted examples. A successful example of the rule-based framework was the AutoSlog Information Extraction system [61]. Table 7.2 presents samples of Auto-Slog’s rules and the extracted named entities.<sup>4</sup> The system starts with a set of simple seed rules for some known entities like *Nicaragua*. In an iterative bootstrapping framework the rules were applied and got extended to extract new entities like *San Sebastian*.

Rule-based systems are relatively precise but usually have low coverage and work well on narrow domains. Their performance usually depends on how comprehensive the rules and lexicons are. Bootstrapping frameworks like [61] are still limited to the domain of the seed rules and lexicon. Furthermore, incorporation of deeper knowledge beyond the surface words and lexicons in to a rule-based system requires expensive manual effort. In contrast, statistical frameworks are more flexible in incorporating richer linguistic knowledge (e.g. syntax) which results in more robust systems.

---

<sup>3</sup>*Gazeteer* is a term that is commonly used to refer to a domain specific lexicon. For example, there are gazeteers for country and city names.

<sup>4</sup>Example is borrowed from [60].

### 7.2.4 Statistical Named Entity Recognition

The rising popularity of the statistical NLP methods along with the expansion of available data resources has directed NER research to data-driven and statistical methods. The use of statistical methods reduced the human effort needed for the tedious construction of rule sets and gazeteers. Soon after their development, statistical and hybrid systems like [51, 52] outperformed the state of the art rule-based systems.

Statistical named entity recognition usually uses the following two main components:

1. Labeled training data: text corpora where named entities are annotated (similar to examples in Table 7.1).
2. A statistical model: a probabilistic representation of the training data.

A statistical model is made of parameters which map a language *event* to a probability. For example a statistical model that is trained on our earlier example (*Fox criticized Washington*), might have parameters such as the probability of the first word in a sentence being a named entity or the probability of certain word (e.g. *Fox*) being labeled as *organization*.

As a supervised learning problem, named entity recognition can be modeled as a classification task for each individual token. However, such approach fails to consider the interdependency between different tokens. In contrast, NER is usually seen as a *structured learning* problem for a sequence of variables. That is the *sequence labeling* view where the learner predicts the labels for the entire sequence of tokens (usually a sentence). This approach allows the modeling of the dependency that exists between different tokens. For example in the earlier example, the class disambiguation for the word *Fox* is easier if the entire sequence (specially the word *Washington*) are included in the prediction.

In a sequence labeling framework a sentence is represented by a set of token variables  $t_1, t_2, \dots, t_N$ . The labeler is expected to find the most likely sequence of named entity labels,  $y_1, y_2, \dots, y_N$ . The set of labels consists of the BIO boundaries along with the named entity types. Thus, the class possibilities for a model which labels person, location, organization are: B-PER, I-PER, B-LOC, I-LOC, B-ORG, I-ORG and O.

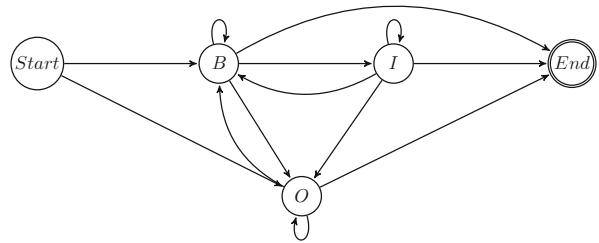
Formulating the problem probabilistically, we would like to find the label sequence which satisfies:

$$S = \underset{y_1 \dots y_N}{\operatorname{argmax}} P(y_1 \dots y_N | t_1 \dots t_N) \quad (7.1)$$

Using the Bayes' theorem of probabilities, we can rewrite and simplify the above formula as:

$$S = \underset{y_1 \dots y_N}{\operatorname{argmax}} P(t_1 \dots t_N | y_1 \dots y_N) P(y_1 \dots y_N) \quad (7.2)$$

**Fig. 7.1** An simplified HMM for detect NE boundaries



There are different ways of modeling the sequence labeling problem. One well-known approach is the hidden Markov model (HMM) [58]. HMM is based on two concepts:

1. A probabilistic graphical model in which class variables are represented by states which are able to *generate* tokens.
2. An assumption that there is a Markov process in the generation of the tokens. The assumption is that the probability of assigning a class to a token depends only on a few earlier tokens (and their class labels).

HMM formulates the labeling problem as:

$$S = \underset{y_1 \dots y_N}{\operatorname{argmax}} P(t_1 \dots t_n | y_1 \dots y_n) P(y_1 \dots y_n) \quad (7.3)$$

$$= \prod_{i=1, \dots, N} P(t_i | y_i) P(y_i | y_{i-1}) \quad (7.4)$$

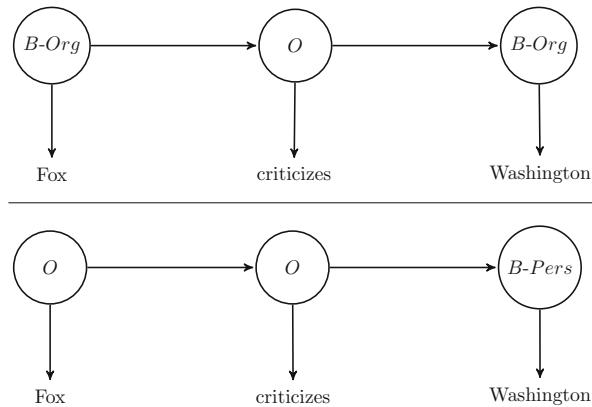
In the formulation shown above, the Markov assumption allows us to shorten the context for computing  $P(y_1 \dots y_n)$  and simply use  $P(y_i | y_{i-1})$ . This is the first order HMM in which the model includes the contextual information for one previous word. Richer models with higher order use longer context with much larger parameter space.

Figure 7.1 presents an HMM for a simplified task of finding named entity boundaries. In this model, the class labels are limited to only three boundary labels (B, I and, O). The start and the end states are used to enforce boundaries for the sequence labeling task. Here, the sequence labeling of named entity boundaries follows a *generative* story:

1. The sequence begins at the *Start* state.
2. For each token position in the sequence, there is a probabilistic state transition where the class label gets decided.
3. After each transition, the destination state *generates* a word.
4. The sequence finishes at the *End* state.

In order to follow the above HMM framework, two sets of parameters are needed to train the HMM:

**Fig. 7.2** An ambiguous example with the correct and an incorrect labeling by HMM



1.  $P(y_i | y_{i-1})$ : state transition probability which is the conditional probability of the current token's label given the previous token's label.
2.  $P(t_i | y_i)$ : the probability of generating a token, given its label.

During the training, the model learns these two sets of parameters by counting and calculating the probability of different state transitions and word generations in the training data.

Having a trained HMM, we can choose the most likely tag sequence that maximizes the product of the two parameters. Since the labeling takes place globally for the entire sequence, the model can deal with some of the class ambiguities. Figure 7.2 presents the correct and an incorrect sequence of HMM states (labels) for an ambiguous sequence. Here, the tagging of *Fox* as (news) organization influences the following state sequence and results in the tagging of *Washington* as a (government) organization. In the second labeling, the model collectively labels *Fox* as non-NE and *Washington* as person.

In general, the procedure to find the most likely label (state) sequence is named *decoding*. Methods such as the Viterbi algorithm which use dynamic programming, are commonly used for the HMM decoding.<sup>5</sup>

In order to train richer NER models, one would like to incorporate deeper linguistic information like long distance dependencies, morphological agreements, etc. HMM assumes that tokens are independent of each other. This assumption limits the scope of the contextual information that the NER model can use. Thus, learning features are limited to the current token [16].

In richer discriminative models such as the Maximum Entropy [15], the Perceptron [20] and the Conditional Random Fields (CRF) [41], there is no assumption made about the independence of the words and their class labels. This relaxed framework allows the model to benefit from diverse overlapping (non-independent) features [13, 49]. For example, the model can use different lexicons of foreign names

<sup>5</sup>Two well-explained usage of the above HMM framework can be found in [37, 48].

or cultural genres [59]. Moreover, global features which are collected in context beyond the current sentence have also been incorporated into discriminative models [19, 59].

### 7.2.5 Hybrid Systems

Hybrid named entity recognition systems combine two or more systems to reach a collective decision. These systems have shown improvement over their baseline counterparts. The work of [17] in combining statistical and rule-based systems in the MUC competitions as well as the work of [26] in combining different statistical learning algorithms are two successful examples of hybrid NER. In Sect. 7.4 we will discuss two Semitic NER systems that use hybrid frameworks, with different learning algorithms.

### 7.2.6 Evaluation and Shared Tasks

Named entity recognition systems are evaluated by running them on human-labeled data and comparing their results against this gold-standard. The comparison is usually at the phrase level, giving full credit for complete boundary and category matches and no credit for partial matches. The commonly used evaluation metrics are the *precision* and *recall* which have been borrowed from Information Retrieval evaluation. *Recall* measures the coverage of the system i.e. the percentage of gold-standard named entities that the system is able to recognize. *Precision* measures the accuracy, i.e. the percentage of the labeled named entities that agree with the gold standard.

A third measure ( $F_1$ ) is used to combine these two metrics as shown in the following:

$$\text{Precision} = \frac{C}{L} \quad \text{Recall} = \frac{C}{G} \quad F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Where:

- $L$ : Number of labeled named entities
- $G$ : Number of gold-standard named entities
- $C$ : Number of correctly labeled named entities

The  $F_1$  measure has been the de facto evaluation and optimization metric for named entity recognition, because of its simplicity and generality. However, there have been debates about how informative this metric really is. In a NLP blog

note,<sup>6</sup> Chris Manning compares various types of errors in NER and argues that  $F_1$  penalizes some types of errors too much. For example, a perfect boundary recognition with incorrect categorization receives the same penalty as a total miss of a named entity. Furthermore, Manning shows that optimization for such an evaluation metric biases the system towards labeling fewer named entities.

### 7.2.7 Evaluation Campaigns

Since its introduction, named entity recognition has been a popular subject for group evaluation. There have been three major NER evaluation campaigns as part of NLP conferences. The shared task at the 6th and the 7th Message Understanding Conference (MUC) were the first NER system competitions<sup>7</sup> which consisted of extracting entities like person, location, organization, temporal and number expressions [66]. The evaluation followed the template-filling framework of Information Extraction (IE) with the standard precision, recall metrics. MUC's evaluation counts partial credits for cases in which the boundary of the entity or its class are incorrect.

In 2002 and 2003, the Conference of Natural Language Learning (CoNLL) included a language-independent shared task on named entity recognition. These were important forums for language-independent NER<sup>8</sup> where a diverse set of learning techniques and features were explored. The BIO encoding of the NER problem, the addition of the miscellaneous (MISC) class of named entities<sup>9</sup> and also the exact matching criteria in the evaluations were protocols which were introduced in the CoNLL shared tasks and since then have been followed by many researchers.

The Automatic Content Extraction (ACE) program was a multilingual (Arabic, Chinese and English) program that was focused on tasks such as named entity recognition and mention detection [23]. The program has created substantial amount of gold-standard data for the three languages. The Arabic corpus is probably one the most important dataset for Semitic NER. ACE introduced a few new conventions for named entity recognition; in addition to the standard person, location and organization classes, ACE added additional entity types such as *facility*, *vehicle*, *weapon* and *geographic point entity (GPE)*. Furthermore, ACE used a more comprehensive evaluation framework. The evaluation incorporated several kinds of errors into an integrated scoring mechanism. This was aimed to address some of the concerns regarding the complete matching criteria of CoNLL.

---

<sup>6</sup><http://nlpers.blogspot.com/2006/08/doing-named-entity-recognition-dont.html>

<sup>7</sup>The term *named entity* was first introduced at the MUC-6 [54].

<sup>8</sup>The 2002 shared task was conducted on Dutch and Spanish [67]. The 2003 shared task was conducted on English and German [68].

<sup>9</sup>Per CoNLL definition, any named entities that does not belong to the person, location and organization classes is considered to be MIS.

### 7.2.8 Beyond Traditional Named Entity Recognition

In the past decade, the scope of named entity recognition has been extended to new categories and topics. Depending on the topic, there can be various categories of named entities. Works such as [63] constructed extended ontology of named entity categories. These ontologies are useful for NER in multi-topic texts like Wikipedia or weblogs. Balasuriya et al. [8] highlight the substantial difference between entities appearing in English Wikipedia versus traditional corpora, and the effects of this difference on NER performance. There is evidence that models trained on Wikipedia data generally perform well on corpora with narrower domains. Nothman et al. [56] and Balasuriya et al. [8] show that NER models trained on both automatically and manually annotated Wikipedia corpora perform reasonably well on news corpora. The reverse scenario does not hold true for models trained on news text and there is a major performance drop.

It is no surprise that the state-of-the-art news-based NER systems perform less impressively when subjected to new topics and domains. Domain and topic diversity of named entities has been studied within the framework of domain adaptation research. In domain adaptation studies, the traditional domain which usually matches the labeled training data in most part is the *source* domain and the novel domain which usually lacks large amount of labeled data is the *target* domain. A group of these methods use semi-supervised learning frameworks such as self-training and select the most informative features and training instances to adapt a source domain learner to a new target domain. Wu et al. [71] bootstrap the NER learner with a subset of unlabeled instances that bridge the source and target domains. Jiang and Zhai [36] as well as [21] make use of some labeled target-domain data, augmenting the feature space of the source model with features specific to the target domain.

There is also a body of work on extraction of named entities from biological and medical text.<sup>10</sup> In these works, target named entities range from the names of enzymes and proteins in biology texts to symptoms, medicines and diseases in medical records.

## 7.3 Named Entity Recognition for Semitic Languages

Named entity recognition inherits many of the general problems of Semitic NLP; complex morphology, the optional nature of short vowels (diacritics) and generally the non-standard orthography are well known problems involved in the processing of Semitic languages which also affect NER.

Except Arabic, NER is an under-studied problem for other Semitic languages. There is small to medium amount of labeled data for Arabic and Hebrew NER

---

<sup>10</sup>See [43] and [45] for an overview Biomedical NER.

**Table 7.3** Examples of morphological and orthographic challenges in Semitic NER

	Morphology	Orthography
Arabic	لأمريكيين (ل + ال + أمريكي + ين) llAmrykyyn (l + Al + Amryky + yn) <i>to the Americans</i>	براد (براد / براد) brAd (brrAd / brAd) <i>refrigerator / Brad</i>
Hebrew	באמריקה (ב + אמריקה) bamrika (b + amrika) <i>in America</i>	אלון (אלון / אלון) alwn (alun / alon) <i>to lodge / Alon</i>

and for the rest of Semitic languages there is almost no resource. In the following sections we review the common challenges and some solutions for Semitic NER with a special focus on Arabic and Hebrew.

### 7.3.1 Challenges in Semitic Named Entity Recognition

There are four main problems involved with Semitic languages which make Semitic NER a challenging task. Table 7.3 illustrates samples for some of these problems in Arabic and Hebrew.<sup>11</sup>

*Absence of capitalization:* For English and other Latin-scripted languages, the use of capitalization is a helpful indicator for named entities.<sup>12</sup> Maltese is the only Semitic language that uses capitalization in this similar fashion. The lack of capitalization in other Semitic languages like Arabic and Hebrew increases the ambiguity both in recognition and categorization of the named entities.

*Optional vowels:* Vowels are present in different levels in Semitic languages. Short vowels (diacritics) are optional in Arabic and Hebrew. In Amharic writing, vowels are mostly present (except in the case of gemination) and Maltese's Latin scripting explicitly incorporates vowels. Whenever vowels become optional (as they are in Hebrew and Arabic), ambiguity increases. For example in Table 7.3, the non-vocalized surface form of the Hebrew word *alwn* in can be interpreted as the verb *alun* or the person name *Alon*. Similarly, the Arabic token *BraD* might refer to the Arabic noun *brrAd* (with an optional gemination) or the Western name *Brad*.

*Complex morphology:* The concatenative morphology in Semitic languages makes it possible for a named entity to get attached to different clitics and form a longer phrase. For example in Table 7.3, the Arabic entity (Amryky: American) is agglutinated to a the *Al* (definite) proclitic and the *yn* (plural) suffix and forms a noun phrase (the Americans). In order to recognize and categorize such entities,

<sup>11</sup>Samples for the Arabic are shown using the Buckwalter romanization [18] and samples for the Hebrews are shown using the romanization scheme in [40].

<sup>12</sup>Capitalization is not used consistently among Latin-scripted languages. Capitalization typically applies to proper nouns in English, to all nouns in German, and to any important noun in Italian.

morphological analysis needs to be performed. Thus, morphological analysis and disambiguation is expected to play an important role in Semitic NER.

*Transliteration and diversity of spelling:* Multiple transliteration of named entities is a common problem in most languages including the Semitic family. The non-standard mapping of cross-lingual consonants results in various spellings of phonologically complex names such as *Schwarzenegger* in Arabic or Hebrew. Moreover, in most Semitic languages we observe some diversity of spelling both for local and foreign names. For example, the first letter of person name Haylü in Amharic can take multiple forms which results in six different spellings of the name [65]. Another example is the multiple mapping between the “h” or “t” consonants in the Roman languages to Arabic.<sup>13</sup>

### 7.3.2 Approaches to Semitic Named Entity Recognition

There is an extensive body of works on Arabic named entity recognition. That includes the creation of gazetteers, labeled datasets, statistical and also rule-based systems. The system in [64] is an example of a rule-based approach. The approach includes creation of name lists for the named entities and non-entities (white and black lists) along with the extraction rules (in form of regular expressions). The RENAR system [73] is a more recent rule-based approach. It is based on searching gazetteers followed by a set of hand-crafted grammar recognition rules for extracting out of lexicon entities. Finally, the system of [57] is a more recent hybrid approach in combining a rule-based system with various statistical classifiers in extracting a large set of named entity classes.

A range of statistical learning algorithms have been applied to Arabic NER: Nezda et al. [55] and Benajiba et al. [11] use Maximum Entropy, Benajiba et al. [12], Abdul-Hamid and Darwish [1] use Support Vector Machines and Farber et al. [24] as well as [53] use Perceptron. A range of lexical, morphological and syntactic features have been used in these statistical systems. The development and the distribution of tools such as MADA [30] and AMIRA [22] and SAMA [46] led to studies on the role and effects of morphological features in Arabic named entity recognition. Moreover, the English translation information provided by MADA has provided useful bilingual features. For example, Farber et al. [24] use the gloss translations to estimate a capitalization feature for Arabic words. In other studies such as [12], the MADA package has been extensively used to explore different morphological features with different learning frameworks. In the next section we will review the work in [12] as a case study for Semitic NER.<sup>14</sup>

---

<sup>13</sup>For example, foreign person names such as Hayato (Japanese) or Tahvo (Finish) can be mapped to different Arabic spellings.

<sup>14</sup>Other relevant works on Arabic NER: [25, 47, 50, 62].

There are two major published works on Hebrew NER. Lemmerski [44]<sup>15</sup> uses a Maximum Entropy sequence classifier and a set of lexical and morphological features. Features include lexeme, POS tag, several named entity lexicon and information extracted from hand-crafted regular expression patterns. In order to train the system with labeled data, a morphologically tagged corpus was manually annotated with the named entity information. The annotation was in the framework of MUC-7 on a set of 50 Hebrew news articles. In an extended work, Ben Mordecai and Elhadad [14] use three systems separately and jointly for Hebrew named entity recognition. In the following section we will review this work as a case study for Semitic NER.

Similar to English, the majority of the systems for Arabic and Hebrew NER are trained and evaluated on the news corpora. The named entity categories usually include the traditional person, organization, location classes. Some of the Arabic NER works go beyond the traditional classes and introduce additional classes relevant to the domain. Shaalan and Raza [64] extract ten named entity classes related to the business news domain. Some of the numeric classes are non-conventional (e.g. phone number) and contributed to the development of new labeled dataset for evaluation. The system in [55] uses an extensive annotation of text from the Arabic Tree Bank with 18 classes of named entities. The categories include several quantitative and temporal classes such as money and time.

Arabic Wikipedia has been the test-bed for a few recent studies on named entity recognition. Mohit et al. [53] demonstrate that traditional named entity classes are insufficient for a multi-topic corpus like Wikipedia. They use a relaxed annotation framework in which article-specific classes are considered and labeled. For example, for an article about *Atom*, annotators introduced and labeled particle names (e.g. electron, proton). Furthermore, Mohit et al. [53] develop an NER system which recognize (but does not categorize) their extended set of named entity classes for Arabic Wikipedia. Extended classes of named entities have also been used as a taxonomy for Arabic Wikipedia. Alotaibi and Lee [4] use a supervised classification framework to assign Wikipedia articles to one of their eight coarse-grained named entity classes.

Semitic NER has been studied as part of other relevant tasks. For example, Kirschenbaum and Wintner [40] locate named entities for the purpose of translating them from Hebrew to English. We will review these works in Sect. 7.5 along with other works relevant to Semitic NER.

## 7.4 Case Studies

In this section we review the work of Benajiba et al. [12] and also Ben Mordecai and Elhadad [14] as case studies in (respectively) Arabic and Hebrew named entity recognition. The two works share a common approach to Semitic NER: Exploring

---

<sup>15</sup>Published in Hebrew.

different learning algorithms and features sets and also lexicon construction to achieve an optimal performance. Benajiba et al. [12] aim at finding the optimal feature set for different classes of Arabic named entities. Ben Mordecai and Elhadad [14] include a brief analysis of effective features, but mainly focus on combining different learning methods for optimizing Hebrew NER. In the following we review different aspects of these two works:

### **7.4.1 Learning Algorithms**

The system in [12] is an empirical framework to study the effects of different features on Arabic NER. It uses two discriminative learners (support vector machines and conditional random fields) to construct classifiers for each named entity class. Thus, there are classifiers for the person class, location class, etc. that label the named entity boundaries. After the initial per-class labeling, a collective NER classification takes place with a voting mechanism.

Ben Mordecai and Elhadad [14] explore a baseline rule-based system made of regular expressions and two statistical classifiers (Hidden Markov Model and Maximum Entropy). After trying different HMM schemes, they chose a structure where each state is made of a named entity class joined with the POS tag. Moreover, the HMM states omit a feature representation of the words. By such joint inclusion of the class label and the POS tag, they incorporate some structural knowledge in to their model. In contrast, their standard maximum entropy model of NER is not constrained and freely uses features independent of each other.

### **7.4.2 Features**

Feature selection is an important component of these two case studies and also most other Arabic and Hebrew NER studies. As discussed earlier, NER is a heavily lexicalized task and models rely strongly on lexical and contextual features. A standard set of contextual features such as the preceding and following tokens and morphemes are inherited from the English systems. Furthermore, morphological complexities of Semitic languages requires explicit inclusion of morphological features into the models. In Arabic, for example the gender or number agreements between adjacent proper nouns are important hints to find the spans of the named entity. In the absence of robust morphological and syntactic analyzers (e.g. in Hebrew systems), models benefit from shallow structural and morphological features such as affixes or the token's position in the sentence.

Table 7.4 compares features used in our two cases studies [12, 14]. The feature set used in the Arabic system includes lexical, contextual features and morphological features as well as features from named entity lexicons built from resources like Wikipedia. Most of the morphological features are extracted by using the Arabic MADA toolkit. The effectiveness of features has been estimated for each of the

**Table 7.4** Features in the Arabic [12] and the Hebrew [14] systems

Feature	Arabic	Hebrew
Context (prev. and follow. $n$ tokens)	×	×
Affixes (shallow morphology) tag	×	×
POS tag	×	×
Gazeteers	×	×
Base phrase chunk	×	
Corresponding English capitalization	×	
Morphological analysis (person, gender, number, etc.)	×	
Frequency features (being a frequent nouns, phrase, token)		×
Structural features (token's position in the sentence)		×
Regular expressions		×
Lemma		×

named entity classes. Some of these features tend to be contributing for most named entity classes (e.g. the morphological aspect or English capitalization). However, because each class holds its own classifier and feature analysis, there is not always a strong consensus about the general effectiveness of a certain feature.

The feature set in [14] comprises of morphological, structural lexical and contextual features. For morphological features there is not much Hebrew-specific analysis and they are limited to POS tags, affixes and the lemma. However, there is a set of regular expressions and structural features which provide some language specific flavor to the model. Furthermore, gazetteer features use a few lexicons that hold a comprehensive list of frequent nouns and expressions and also use geographical and organizational lists.

### 7.4.3 Experiments

Both studies use system combination algorithms. However, the combination is aimed toward different goals. For Benajiba et al. [12], each entity class has a separate classifier and feature set. The feature-based ranking framework (Fuzzy Borda Voting Scheme) is a mechanism to combine these different classifiers into one final classifier. There is an average of 2 % improvement in the  $F_1$  score after reaching the optimum feature set of classifier voting. The support vector machines classifier outperforms others for the majority of classes and datasets while lexical features are the most contributing ones in most experiments.

System combination in [14] is based on a simple recall-oriented heuristic: Take the output of the best individual system (maximum entropy) and use the other two taggers as the back-off. Finally, the empirical experiments show that dictionary features along with the POS tag tend to be the most contributing features.

To summarize, the Arabic system in [12] and the Hebrew system in [14] are successful examples of Semitic NER using a hybrid mixture of supervised learners.

Both systems explore language-specific aspects of the problem, but in different ways; Ben Mordecai and Elhadad [14] use language-specific regular expressions to locate potential entities. Benajiba et al. [12] explicitly incorporate linguistic knowledge (e.g. Arabic morphology) as features in to its hybrid learning framework.

## 7.5 Relevant Problems

The importance of named entities for multilingual applications such as machine translation and cross language information retrieval has led researchers to focus on a few other problems which overlap with NER. Here we have a brief overview on three of such problems where Semitic languages (Arabic and Hebrew) have been studied.

### 7.5.1 *Named Entity Translation and Transliteration*

The multilingual named entity information is useful for applications such as cross language information retrieval or machine translation. For example, Hermjakob et al. [32] have shown that inclusion of transliteration information improves machine translation quality. Also, Babych and Hartley [7] showed that incorporation of bilingual named entity information in general improves machine translation quality.

Named entities usually are either translated or transliterated across languages. Compound named entities which are composed of simple nominals (as opposed to proper nouns) might be translated across languages. For example an organizational entity like *The State Department* usually gets translated. In contrast, named entities composed of proper nouns such as *IBM* or *Adidas* usually get transliterated across languages. Table 7.5 presents examples of translation and transliterations for Arabic and Hebrew named entities.

There is a body of work on translation and transliteration of named entities for Arabic and Hebrew. Al-Onaizan and Knight[3] address the named entity translation problem. Their approach has two folds: baseline translation and transliteration of the named entities and later, a filtering based on the target language corpus. The underlying assumption is based on the occurrences of the named entities in the international news: names which are important and frequent in the source language (Arabic), are also frequent in the target language (English).

An important decision for a multilingual system (e.g. machine translation) is whether to translate or transliterate a given source language named entity. Hermjakob et al. [32] address this problem using a supervised classification approach. They use a parallel corpus of phrases which include bilingual transliterated name pairs. The Arabic side of the transliterated bitext is used to train a classifier which highlights words of a monolingual (Arabic) text that can be

**Table 7.5** Translation vs. transliteration of named entities in Arabic and Hebrew

		Translation	Transliteration
Arabic	<b>Source:</b>	البحر المتوسط / the-sea the-middle	rwmAnsyp / رومنسية
	<b>Gloss:</b>		Romantism
	<b>Translation:</b>	Mediterranean Sea	Romantism
Hebrew	<b>Source:</b>	הים התיכון / the-sea the-central	eqzistentzializm / אקסיסטנציאליזם
	<b>Gloss:</b>		Existentialism
	<b>Translation:</b>	Mediterranean Sea	Existentialism

transliterated. Similar classification frameworks have also been examined for the decision making of translation vs. transliteration for Hebrew[28,40].

Machine transliteration deals with named entities that are translated with *preserved pronunciation* [38]. There are specific challenges in Arabic and Hebrew orthography and phonetics which add to the transliteration challenge. These include the optional nature of vowels, the absence of certain sounds (e.g. *p* in Arabic), zero or many mapping of certain sounds to Latin-based letters (e.g. multiple *h* in Arabic or *khaf* in Hebrew). An earlier approach to the problem is described in [2] which is a hybrid combination of phonetic-based and spelling-based models. The extracted transliterations are post-processed by a target language (English) spell checker. There are also transliteration studies which do not involve transliterating the term from scratch. In [32], the transliterated candidates are extracted from a bilingual phrase corpus and the transliteration problem is practically converted to a search problem. There, the system uses a scoring function to filter out the noisy transliterations using a large English corpus.<sup>16</sup> In a relevant framework, the work of Azab et al. [6] aims at automating the English to Arabic translation vs. transliteration decision and reducing the out of vocabulary terms of the MT system. They model the decision as a binary classification problem and later use their classifier within a SMT pipeline to direct a subset of source language named entities to a transliteration module.

For Hebrew, Goldberg and Elhadad [28] identify the borrowed and transliterated words. Their decision is binary: A word is either generated by a Hebrew language model, or by a foreign language model. They train a generative classifier using a noisy list of borrowed words along with regular Hebrew text. The work of Kirschenbaum and Wintner in [40] is also an effort to locate and transliterate the appropriate Hebrew terms. The framework is a single-class classifier which locates entities that are supposed to be transliterated.

<sup>16</sup>For more information about Arabic transliteration, see: [29].

**Table 7.6** An Arabic example of entity detection and tracking with gloss and literal translations

Source	أَوْبَاماً قَامَ بِحُلفِ اليمينِ بِوَصْفِهِ عَضُوِّ مَجْلِسِ الشُّورَى وَهُوَ خَامِسٌ عَضُوٌّ مِنْ أَعْصُولِ إِفْرِيقِيَّةِ.
Gloss	<b>AwbAmA</b> qAm bHlf Alymyn bwSfh EDw mjls Al\$ywx <b>whw</b> xAms EDw bmjls Al\$ywx mn >Swl >frqyp
Translation	<i>Obama took oath as a senate member which is the fifth African-American senator</i>

### 7.5.2 Entity Detection and Tracking

Mention detection is a subtask of information extraction which is focused on the identification of entities and the tracking of their associations to each other. Mentions can be named entities, nominals, or pro-nominals. Table 7.6 presents an Arabic example of entity detection, along with gloss and literal translations. A detection system is expected to highlight and link the two bold segments of the Arabic example. Entity detection is usually modeled as a sequence classification task where each token in a sentence gets assigned to an entity within the sentence. Similar to NER, there are tokens which are independent of entities and get an *O* label. The detection part of the task is similar to the NER. The tracking part might involve a separate linking model and coreference decoding.

Arabic mention detection was one of the tasks introduced in the ACE program. Florian et al. [27] presented a multi-lingual system which included an Arabic mention detection component. Their system uses two Maximum Entropy models, one for the detection and the other one for tracking. The tracking component is a binary linking model where each token gets either linked to another entity or starts a new entity. Also, there have been two recent studies on the effects of morphology and syntactic analysis on Arabic mention detection[9, 10] in which, richer Arabic linguistic knowledge boosted the performance.

### 7.5.3 Projection

Availability of parallel corpora, automatic word alignment and translation systems resulted in a body of work on resource projection [72]. In a projection framework we use a word-aligned corpus to *project* some linguistic information (e.g. named entity boundaries) from a language (e.g. English) to another language (e.g. Hebrew). This has been a useful framework for equipping resource-poor languages with some labeled data. Projection is not always a deterministic operation and cross lingual

**Fig. 7.3** An NER projection example from English to Hebrew

differences can make it a challenging task. Figure 7.3 demonstrate an example of named entity projection from English to Hebrew. It can be seen that morphological richness of the Hebrew does not allow a 1-1 entity mapping across two languages. Thus morphological analysis and segmentation should be considered as part of the a projection pipeline.

There have been some successful attempts on the projection of entity information for Arabic. Hassan et al. [31] extract bilingual named entity pairs from parallel and comparable corpora using similarity metrics that use phonetic and translation model information. Zitouni and Florian [74] study the use of projection (through English to Arabic machine translation) to improve Arabic mention detection. Benajiba and Zitouni [10] directly project the mention detection information using automatic word alignments. The projected Arabic corpus provides new features which augments and improves the baseline Arabic mention detection system. Huang et al. [34] study the problem of finding various English spelling of Arabic names which affects machine translation and information extraction systems. They use a projection framework to locate various spelling of a given Arabic name.

## 7.6 Labeled Named Entity Recognition Corpora

Similar to the research, data resources for the Semitic NER have been limited to Arabic and Hebrew. The Automatic Content Extraction (ACE) program is a multilingual information extraction effort focused on Arabic, Chinese and English. Over the past decade, Arabic has been one of the focus languages of the Entity Detection and Tracking (EDT) task of the ACE. As a result, ACE has prepared a few standard Arabic corpora with named entity information [70]. These corpora are primarily in the newswire domain with recent additions of weblogs and broadcast news text. The named entity categories are targeted towards the political news. They include Person, Location, Organization, Facility, Weapon, Vehicle and Geo-Political Entity (GPE). The Arabic named entity annotations are performed with character-level information which boosts the accuracy of the data for morphologically compound tokens.<sup>17</sup> ACE has been releasing most of its dataset through the Linguistic Data Consortium (LDC).

In addition to the standard ACE datasets, a few projects have resulted in annotation of new NER datasets. The Ontonotes project [33] is an ongoing large scale multilingual annotation effort with several layers of linguistic information on texts collected from a diverse set of domains. Arabic Ontonotes includes annotation

---

<sup>17</sup>See [42] for more information about the Arabic ACE dataset.

of parsing, word senses, coreferences and named entities.<sup>18</sup> The publicly released<sup>19</sup> Arabic ANER corpus [11] is a token-level annotated newswire corpora with four named entity classes: person, location, organization and miscellaneous. Mohit et al. [53] also have released a corpus of Arabic Wikipedia articles with an extended set of named entity categories. Finally, Attia et al. [5] created a large scale lexicon of Arabic named entities from resources such as Wikipedia.<sup>20</sup>

Named entity annotation for Hebrew has been limited to a few projects that we discussed earlier. Hebrew corpus annotation of named entities are reported in [14, 44]. Furthermore, the annotated corpora in [35] includes a layer of named entity information.

## 7.7 Future Challenges and Opportunities

Named entity recognition is still far from a solved problem for Semitic languages. Amharic, Syriac and Maltese lack the basic data resources for building a system. The  $F_1$  performance of the best Arabic and Hebrew systems varies between 60 and 80 % depending on the text genres. Most of the available labeled datasets are mainly news wire corpora which might degrade the NER performance in other topics and domains.

There are many interesting open questions to be explored. For the low resource languages like Amharic or Syriac, well established frameworks such as active learning or projection can be explored to create the basic data requirements and estimating basic models. Online resources such as Wikipedia can also provide the basic named entity corpora and lexicons.<sup>21</sup>

For medium-resource languages like Arabic and Hebrew, NER needs to be tested in new topics and genres with extended named entity classes. To do so, semi-supervised learning frameworks along with domain adaptation methods are the natural starting solutions. Morphological information plays an important role in Semitic NER. Thus, richer incorporation of morphology in NER models in form of joint modeling is an interesting avenue to explore. Moreover richer linguistic information such as constituency and dependency parsing, semantic resources such as the Wordnet and Ontonotes are expected to enrich NER models.

---

<sup>18</sup>The fourth release of Ontonotes includes named entity annotation for a corpus of 300,000 words.

<sup>19</sup>currently at <http://www1.ccsl.columbia.edu/~ybenajiba/downloads.html>

<sup>20</sup>Work presented in [55, 64] also report a large scale annotation of named entity information. However the datasets were not released publicly.

<sup>21</sup>According to Wikipedia statistics, Amharic Wikipedia has more than 10,000 articles which is a promising resource for gazetteer construction.

## 7.8 Summary

We reviewed named entity recognition (NER) as an important task for processing Semitic languages. We first sketched an overview of NER research, its history and the current state of the art. We followed with problems specific to Semitic NER and reviewed a wide range of approaches for Arabic and Hebrew NER. We observed that complex morphology and the lack of capitalization create additional challenges for Semitic NER. We focused on two case studies for Arabic and Hebrew and reviewed their learning frameworks and features. Moreover, we explored the state of data resources and research on relevant tasks such as named entity translation, transliteration and projection for Hebrew and Arabic. We concluded that Semitic NER is still an open problem. For low resource languages such as Amharic and Syriac basic data resources are still needed for constructing baseline systems. For Arabic and Hebrew, inclusion of richer linguistic information (e.g. dependency parsing) and adaptation of the current systems to new text domains are interesting avenues to explore.

**Acknowledgements** I am grateful to Kemal Oflazer, Houda Bouamor, Emily Alp and two anonymous reviewers for their comments and feedback. This publication was made possible by grant YSREP-1-018-1-004 from the Qatar National Research Fund (a member of the Qatar Foundation). The statements made herein are solely the responsibility of the authors.

## References

1. Abdul-Hamid A, Darwish K (2010) Simplified feature set for Arabic named entity recognition. In: Proceedings of the 2010 named entities workshop. Association for Computational Linguistics, Uppsala, pp 110–115
2. Al-Onaizan Y, Knight K (2002a) Machine transliteration of names in Arabic texts. In: Proceedings of the ACL-02 workshop on computational approaches to Semitic languages, Philadelphia. Association for Computational Linguistics
3. Al-Onaizan Y, Knight K (2002b) Translating named entities using monolingual and bilingual resources. In: Proceedings of 40th annual meeting of the Association for Computational Linguistics, Philadelphia. Association for Computational Linguistics, pp 400–408
4. Alotaibi F, Lee M (2012) Mapping Arabic Wikipedia into the named entities taxonomy. In: Proceedings of COLING 2012: posters, Mumbai. The COLING 2012 Organizing Committee, pp 43–52
5. Attia M, Toral A, Tounsi L, Monachini M, van Genabith J (2010) An automatically built named entity lexicon for Arabic. In: Calzolari N, Choukri K, Maegaard B, Mariani J, Odijk J, Piperidis S, Rosner M, Tapia D (eds) Proceedings of the seventh conference on international language resources and evaluation (LREC'10), Valletta. European Language Resources Association (ELRA)
6. Azab M, Bouamor H, Mohit B, Oflazer K (2013) Dudley North visits North London: learning when to transliterate to arabic. In: Proceedings of the 2013 conference of the North American chapter of the Association for Computational Linguistics: human language technologies (NAACL-HLT 2013), Atlanta. Association for Computational Linguistics
7. Babych B, Hartley A (2003) Improving machine translation quality with automatic named entity recognition. In: Proceedings of the 7th international EAMT workshop on MT and other language technology tools, EAMT '03, Dublin

8. Balasuriya D, Ringland N, Nothman J, Murphy T, Curran JR (2009) Named entity recognition in Wikipedia. In: Proceedings of the 2009 workshop on the people's web meets NLP: collaboratively constructed Semantic resources. Association for Computational Linguistics, Suntec, pp 10–18
9. Benajiba Y, Zitouni I (2009) Morphology-based segmentation combination for Arabic mention detection. ACM Trans Asian Lang Inf Process (TALIP) 8:16:1–16:18
10. Benajiba Y, Zitouni I (2010) Enhancing mention detection using projection via aligned corpora. In: Proceedings of the 2010 conference on empirical methods in natural language processing, Cambridge. Association for Computational Linguistics, pp 993–1001
11. Benajiba Y, Rosso P, BenediRuiz JM (2007) ANERsys: an Arabic named entity recognition system based on maximum entropy. In: Gelbukh A (ed) Proceedings of CICLing, Mexico City. Springer, pp 143–153
12. Benajiba Y, Diab M, Rosso P (2008) Arabic named entity recognition using optimized feature sets. In: Proceedings of the 2008 conference on empirical methods in natural language processing, Honolulu. Association for Computational Linguistics, pp 284–293
13. Bender O, Och FJ, Ney H (2003) Maximum entropy models for named entity recognition. In: Proceedings of the seventh conference on natural language learning at HLT-NAACL 2003, Edmonton, pp 148–151
14. Ben Mordecai N, Elhadad M (2005) Hebrew named entity recognition. Master's thesis, Department of Computer Science, Ben Gurion University of the Negev
15. Berger AL, Pietra VJD, Pietra SAD (1996) A maximum entropy approach to natural language processing. Comput Linguist 22:39–71
16. Bikel D, Miller S, Schwarz R, Weischedel R (1997) Nymble: a high-performance learning name-finder. In: Proceedings of the applied natural language processing, Tzgov Chark
17. Borthwick A (1999) A maximum entropy approach to named entity recognition. Phd thesis, Computer Science Department, New York University
18. Buckwalter T (2002) Buckwalter Arabic morphological analyzer version 1.0
19. Chieu HL, Ng HT (2002) Named entity recognition: a maximum entropy approach using global information. In: Proceedings of the 19th international conference on computational linguistics – vol 1, COLING '02. Taipei
20. Collins M (2002) Discriminative training methods for hidden Markov models: theory and experiments with Perceptron algorithms. In: Proceedings of the ACL-02 conference on empirical methods in natural language processing (EMNLP), Philadelphia, pp 1–8
21. Daumé III H (2007) Frustratingly easy domain adaptation. In: Proceedings of the 45th annual meeting of the Association of Computational Linguistics, Prague. Association for Computational Linguistics, pp 256–263
22. Diab M (2009) Second generation tools (AMIRA 2.0): fast and robust tokenization, pos tagging, and base phrase chunking. In: Proceedings of the 2nd international conference on Arabic language resources and tools, Cairo
23. Doddington G, Mitchell A, Przybocki M, Rambow L, Strassel S, Weischedel R (2004) The automatic content extraction (ACE) program-tasks, data and evaluation. In: Proceedings of LREC 2004, Lisbon, pp 837–840
24. Farber B, Freitag D, Habash N, Rambow O (2008) Improving NER in Arabic using a morphological tagger. In: Calzolari N, Choukri K, Maegaard B, Mariani J, Odijk J, Piperidis S, Tapia D (eds) Proceedings of the sixth international language resources and evaluation (LREC'08), Marrakech. European Language Resources Association (ELRA), Marrakesh, pp 2509–2514
25. Fehri H, Haddar K, Ben Hamadou A (2011) Recognition and translation of Arabic named entities with NooJ using a new representation model. In: Proceedings of the 9th international workshop on finite state methods and natural language processing, Blois. Association for Computational Linguistics, pp 134–142
26. Florian R, Ittycheriah A, Jing H, Zhang T (2003) Named entity recognition through classifier combination. In: Proceedings of the seventh conference on natural language learning at HLT-NAACL 2003, Edmonton, pp 168–171

27. Florian R, Hassan H, Ittycheriah A, Jing H, Kambhatla N, Luo X, Nicolov N, Roukos S (2004) A statistical model for multilingual entity detection and tracking. In: Dumais S, Marcu D, Roukos S (eds) Proceedings of the human language technology conference of the North American chapter of the Association for Computational Linguistics: HLT-NAACL 2004, Boston. Association for Computational Linguistics
28. Goldberg Y, Elhadad M (2008) Identification of transliterated foreign words in Hebrew script. In: Proceedings of the 9th international conference on Computational linguistics and intelligent text processing, CICLing'08, Haifa, pp 466–477
29. Habash N, Soudi A, Buckwalter T (2007) On arabic transliteration. *Text Speech Lang Technology* 38:15–22
30. Habash N, Rambow O, Roth R (2009) MADA+TOKAN: a toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In: Choukri K, Maegaard B (eds) Proceedings of the second international conference on Arabic language resources and tools, the MEDAR consortium, Cairo
31. Hassan A, Fahmy H, Hassan H (2007) Improving named entity translation by exploiting comparable and parallel corpora. In: Proceedings of the conference on recent advances in natural language processing (RANLP '07), Borovets
32. Hermjakob U, Knight K, Daumé III H (2008) Name translation in statistical machine translation – learning when to transliterate. In: Proceedings of ACL-08: HLT, Columbus. Association for Computational Linguistics, pp 389–397
33. Hovy E, Marcus M, Palmer M, Ramshaw L, Weischedel R (2006) OntoNotes: the 90% solution. In: Proceedings of the human language technology conference of the NAACL (HLT-NAACL), New York City. Association for Computational Linguistics, pp 57–60
34. Huang F, Emami A, Zitouni I (2008) When Harry met Harri: cross-lingual name spelling normalization. In: Proceedings of the 2008 conference on empirical methods in natural language processing, Honolulu. Association for Computational Linguistics, pp 391–399
35. Itai A, Wintner S (2008) Language resources for Hebrew. *Lang Resour Eval* 42:75–98
36. Jiang J, Zhai C (2006) Exploiting domain structure for named entity recognition. In: Proceedings of the human language technology conference of the NAACL (HLT-NAACL), New York City. Association for Computational Linguistics, pp 74–81
37. Jurafsky D, Martin JH (2008) Speech and language processing. Pearson Prentice Hall, Upper Saddle River
38. Karimi S, Scholer F, Turpin A (2011) Machine transliteration survey. *ACM Comput Surv* 43:17:1–17:46
39. Khalid MA, Jijkoun V, De Rijke M (2008) The impact of named entity normalization on information retrieval for question answering. In: Proceedings of the IR research, 30th European conference on advances in information retrieval, Glasgow, Springer, pp 705–710
40. Kirschenbaum A, Wintner S (2009) Lightly supervised transliteration for machine translation. In: Proceedings of the 12th conference of the European chapter of the ACL (EACL 2009), Athens. Association for Computational Linguistics, pp 433–441
41. Lafferty JD, McCallum A, Pereira FCN (2001) Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of the eighteenth international conference on machine learning, ICML '01, Williamstown. Morgan Kaufmann, pp 282–289
42. LDC (2005) ACE (automatic content extraction) Arabic annotation guidelines for entities, version 5.3.3. Linguistic Data Consortium, Philadelphia
43. Leaman R, Gonzalez G (2008) Banner: an executable survey of advances in biomedical named entity recognition. In: Proceedings of pacific symposium on biocomputing, Kohala Coast, pp 652–663
44. Lemmerski G (2003) Named entity recognition in Hebrew. Master's thesis, Department of Computer Science, Ben Gurion University
45. Leser U, Hakenberg J (2005) What makes a gene name? Named entity recognition in the biomedical literature. *Brief. Bioinform* 6:357–369
46. Maamouri M, Graff D, Bouziri B, Krouna S, Bies A, Kulick S (2010) LDC standard Arabic morphological analyzer (SAMA) version 3.1, LDC2004L02. Linguistic Data Consortium, Philadelphia

47. Maloney J, Niv M (1998) TAGARAB: a fast, accurate arabic name recognizer using high precision morphological analysis. In: Proceedings of the workshop on computational approaches to Semitic languages, Montreal
48. Malouf R (2002) Markov models for language-independent named entity recognition. In: Proceedings of the 6th conference on natural language learning – vol 20, COLING-02, Stroudsburg. Association for Computational Linguistics, pp 1–4
49. McCallum A, Li W (2003) Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In: Daelemans W, Osborne M (eds) Proceedings of the seventh conference on natural language learning at HLT-NAACL 2003, Edmonton, pp 188–191
50. Mesfar S (2007) Named entity recognition for Arabic using syntactic grammars. In: Kedad Z, Lammari N, Métais E, Meziane F, Rezgui Y (eds) Natural language processing and information systems. Lecture notes in computer science, vol 4592. Springer, Berlin, pp 305–316
51. Mikheev A, Moens M, Grover C (1999) Named entity recognition without gazetteers. In: Proceedings of the ninth conference of the European chapter of the Association for Computational Linguistics (EACL-99), Bergen. Association for Computational Linguistics
52. Miller S, Crystal M, Fox H, Ramshaw L, Schwartz R, Stone R, Weischedel R, The Annotation Group (1998) Algorithms that learn to extract information BBN: description of the sift system as used for MUC-7. In: Proceedings of the seventh message understanding conference (MUC-7), Fairfax
53. Mohit B, Schneider N, Bhowmick R, Oflazer K, Smith NA (2012) Recall-oriented learning of named entities in arabic wikipedia. In: Proceedings of the 13th Conference of the European Chapter of the ACL (EACL 2012), Avignon. Association for Computational Linguistics
54. Nadeau D, Sekine S (2007) A survey of named entity recognition and classification. Lingvisticae Investigationes 30:3–26
55. Nezda L, Hickl A, Lehmann J, Fayyaz S (2006) What in the world is a *Shahab*? Wide coverage named entity recognition for Arabic. In: Procedding of LREC, Genoa, pp 41–46
56. Nothman J, Murphy T, Curran JR (2009) Analysing Wikipedia and gold-standard corpora for NER training. In: Proceedings of the 12th conference of the European chapter of the ACL (EACL 2009), Athens. Association for Computational Linguistics, pp 612–620
57. Oudah M, Shaalan K (2012) A pipeline Arabic named entity recognition using a hybrid approach. In: Proceedings of COLING 2012, Mumbai. The COLING 2012 Organizing Committee, pp 2159–2176
58. Rabiner LR (1989) A tutorial on hidden Markov models and selected applications in speech recognition. Proc IEEE 77:257–286
59. Ratnoff L, Roth D (2009) Design challenges and misconceptions in named entity recognition. In: Proceedings of the thirteenth conference on computational natural language learning (CoNLL-2009), Colorado. Association for Computational Linguistics, Boulder, pp 147–155
60. Riloff E, Jones R (1999) Learning dictionaries for information extraction by multi-level bootstrapping. In: Proceedings of the sixteenth national conference on artificial intelligence and the eleventh innovative applications of artificial intelligence conference innovative applications of artificial intelligence, Orlando. American Association for Artificial Intelligence, pp 474–479
61. Riloff EM, Phillips W (2004) Introduction to the sundance and autoslog systems. Technical report, University of Utah
62. Samy D, Moreno A, Guirao JM (2005) A proposal for an Arabic named entity tagger leveraging a parallel corpus. In: Proceedings of the conference of the recent advances in natural language processing (RANLP-05), Borovets
63. Sekine S, Sudo K, Nobata C (2002) Extended named entity hierarchy. In: Proceedings of LREC, Las Palmas
64. Shaalan K, Raza H (2009) NERA: named entity recognition for Arabic. J Am Soc Inf Sci Technol 60(8):1652–1663
65. Sintayehu Z (2001) Automatic classification of Amharic news items: the case of Ethiopian news agency. Master's thesis, School of Information Studies for Africa, Addis Ababa University

66. Sundheim BM (1995) Named entity task definition, version 2.1. In: Proceedings of the Sixth Message Understanding Conference (MUC-6), Columbia
67. Tjong Kim Sang EF (2002) Introduction to the CoNLL-2002 shared task: language-independent named entity recognition. In: Proceedings of the sixth conference on natural language learning (CoNLL-2002), Taipei
68. Tjong Kim Sang EF, De Meulder F (2003) Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In: Daelemans W, Osborne M (eds) Proceedings of the seventh conference on natural language learning at HLT-NAACL 2003, Edmonton, pp 142–147
69. Toral A, Noguera E, Llopis F, Muñoz R (2005) Improving question answering using named entity recognition. In: Natural language processing and information systems, vol 3513/2005. Springer, Berlin/New York, pp 181–191
70. Walker C, Strassel S, Medero J, Maeda K (2006) ACE 2005 multilingual training corpus. LDC2006T06, Linguistic Data Consortium, Philadelphia
71. Wu D, Lee WS, Ye N, Chieu HL (2009) Domain adaptive bootstrapping for named entity recognition. In: Proceedings of the 2009 conference on empirical methods in natural language processing, Singapore. Association for Computational Linguistics, pp 1523–1532
72. Yarowsky D, Ngai G, Wicentowski R (2001) Inducing multilingual text analysis tools via robust projection across aligned corpora. In: Proceedings of the first international conference on human language technology research, HLT '01, Stroudsburg. Association for Computational Linguistics, pp 1–8
73. Zaghouani W (2012) RENAR: A rule-based Arabic named entity recognition system. ACM Trans Asian Lang Inf Process (TALIP) 11:1–13
74. Zitouni I, Florian R (2008) Mention detection crossing the language barrier. In: Proceedings of the 2008 conference on empirical methods in natural language processing, Honolulu. Association for Computational Linguistics, pp 600–609

# Chapter 8

## Anaphora Resolution

Khadiga Mahmoud Seddik and Ali Farghaly

### 8.1 Introduction: Anaphora and Anaphora Resolution

Anaphora is a Greek expression that means “carrying back”. For computational linguistics, anaphora resolution (AR) is concerned with determining the relation between two entities in the text, i.e. defining the conditions under which an entity refers to another one which usually occurs before it. An entity may be a pronoun, a verb, definite descriptions, a lexical modifier, a noun phrase, or a proper noun. The “pointing back” (reference) is called an anaphor and the entity to which it refers is its antecedent. The following sentence has four antecedents and four anaphors. The coreference relationship is indicated by giving both the antecedent and its referent the same index and underlining both.

(1) صرَحَ وزَيْرُ الصَّفَهِ وَالسَّكَانِ<sup>1</sup> بِأَنَّ تَوْفِيرَ الْأَمْنِ<sup>2</sup> فِي الْمُسْتَشْفَيَاتِ سَيَكُونُ لَهُ<sup>3</sup> الْأَوْلَى يَوْمَهُ خَلَالِ الْمَرْحَلَةِ الْمُقْبَلَةِ لِكَيْ يَسْتَطِعَ الْفَرِيقُ الطَّبِيُّ<sup>3</sup> الْقِيَامُ بِعَمَلِهِ<sup>3</sup> عَلَى أَكْمَلِ وَجْهٍ وَأَكْدِ أَنَّهُ<sup>1</sup> سَوْفَ يَلْتَزِمُ بِجَمِيعِ الْقَرَارَاتِ<sup>4</sup> الَّتِي إِتَّخَذَهَا<sup>4</sup>

(1) The minister for Public Health and Population<sub>1</sub> has stated that ensuring security<sub>2</sub> at hospitals will have priority in the next phase so that the medical teams<sub>3</sub> can best perform their<sub>3</sub> jobs. He<sub>1</sub> also confirmed that he is committed to all decrees<sub>4</sub> that he<sub>4</sub> has issued.

The scope of coreference resolution is more general than anaphora resolution since coreference is a set of coreferent referring expressions, and coreference

---

KH.M. Seddik (✉)

Faculty of Computers and Information, Cairo University, Ahmed Zewel St., Giza, Egypt  
e-mail: kh\_seddik88@yahoo.com

A. Farghaly

Computational Linguistics Software Researcher, Networked Insights, 200 W. Adams Street,  
Chicago, IL 60606, USA  
e-mail: alifarghaly@yahoo.com

resolution is the task of identifying all the noun phrases that refer to the same entity in the text, sometimes referred to as mention detection and chaining [50]. On the other hand, anaphora is about coreference of one referring expression with its antecedent, and the process of anaphora resolution is finding the antecedent of an anaphor.

Anaphora resolution is one of the challenging tasks of natural language processing. It is very important since without it a text would not be fully and correctly understood, and without finding the proper antecedent, the meaning and the role of the anaphor cannot be realized.

In this chapter, we present an account of the anaphora resolution task. The chapter consists of ten sections. The first section is an introduction to the problem. In the second section, we present different types of anaphora. Section 8.3 discusses the determinants and factors to anaphora resolution and its effect on increasing system performance. In Sect. 8.4, we discuss the process of anaphora resolution. In Sect. 8.5 we present different approaches to resolving anaphora and we discuss previous work in the field. Section 8.6 discusses the recent work in anaphora resolution, and Sect. 8.7 discusses an important aspect in the anaphora resolution process which is the evaluation of AR systems. In Sects. 8.8 and 8.9, we focus on the anaphora resolution in Semitic languages in particular and the difficulties and challenges facing researchers. Finally, Sect. 8.10 presents a summary of the chapter.

## 8.2 Types of Anaphora

There are various types of anaphora, but we will introduce the most widespread types in the computational linguistics literature.

### 8.2.1 Pronominal Anaphora

The most frequent type of anaphora is the pronominal anaphora, which is characterized by anaphoric pronouns. Pronouns form a special class of anaphors because of their empty semantic structure, i.e. they do not have an independent meaning from their antecedent. Moreover, not all pronouns are anaphoric, such as the *non-anaphoric* “it”.

An example of pronominal anaphora is given in (2):

(2) جدي صفيه<sub>1</sub> ليس لها<sub>1</sub> من الأهل سوانا و سوى بنت<sub>2</sub> واحدة تعيش في ديار الغرب مع زوجها<sub>2</sub> و أبنائهما<sub>2</sub> لم تر هما<sub>1</sub> منذ سنوات و هي<sub>1</sub> تفتقدهما<sub>2</sub> كثيرا

(2) My grandmother “Safeyyah”<sub>1</sub> has only one girl<sub>2</sub> living in a foreign land with her<sub>2</sub> husband and her<sub>2</sub> children who she<sub>1</sub> hasn’t seen for years. She<sub>1</sub> misses her<sub>2</sub> very much.

This is an example of pronominal anaphora, which also has several ambiguities. For example “ها/her” in “زوجها/her husband” and “أبنائهما/her children”

could refer either to the grandmother or to the granddaughter. Also “هي/she” and “تفقدها/misses her” are ambiguous. Who is missing who?

### 8.2.2 Lexical Anaphora

Lexical anaphora is sometimes called “definite noun phrase anaphora” [31]. It is identified when the referring expressions are definite descriptions or proper nouns [17]. The antecedent is referred to by a definite noun phrase representing either the same concept or semantically close concepts (e.g. synonyms, superordinates), as shown in (3).

(3) حضر علماء الكمبيوتر من جميع دول العالم الاجتماع. وقد وجد الحاضرون صعوبة في مجاراة سرعه العرض

(3) Computer scientists<sub>1</sub> from many different countries attended the meeting. The participants<sub>1</sub> found it hard to cope with the speed of the presentation.

In this example, “participants” refers back to “Computer scientists/”علماء الكمبيوتر which is a definite noun phrase that represents a semantically close concept.

### 8.2.3 Comparative Anaphora

The comparative anaphora is when the anaphoric expressions are introduced by lexical modifiers (e.g. other, another) or comparative adjectives (e.g. better, greater) [17].

The “one anaphora” is a type of comparative anaphora and there are many researchers who focus on the “one anaphora” [31]. An example of the “one anaphora” is shown in (4) where the word “one” refers back to the word “rose”:

(4) زرعت الفتاه وردة حمراء بجوار واحدة صفراء

(4) The girl planted a red rose<sub>1</sub> next to a yellow one<sub>1</sub>.

## 8.3 Determinants in Anaphora Resolution

There are several factors that facilitate identifying the antecedent of an anaphoric expression. These are: the distance between an anaphoric expression and its antecedent; lexical constraints such as gender and number agreement that are used to eliminate some antecedent candidates; syntactic roles which can indicate preference for particular antecedent candidates; etc.

These factors are divided into two categories: “eliminating” and “preferential” factors [31]. **Eliminating factors** are those which exclude certain NPs from the

set of possible candidates and **preferential factors** are those which give more preference to certain candidates over others.

There are other terminologies used in the field for the eliminating and preferential factors. Carbonell and Brown [5] use terms such as “constraints” and “preferences”, whereas E. Rich and S. LuperFoy [41] use terms like “constraints” and “proposers”. Other authors call them simply “factors”, “attributes”, or “symptoms” [26].

### 8.3.1 Eliminating Factors

The following are some of the eliminating factors used in anaphora resolution.

#### Gender Agreement

The sentence in (5) illustrates the requirement that the anaphor and its antecedent must agree in gender. The closest antecedent to the pronoun in “أُنْهِ” is the proper noun “سَمِيرَة/Samira” but the mismatch in gender between the pronoun and the closest antecedent should eliminate “سَمِيرَة/Samira” from being the correct antecedent and as shown, coreference of this pronoun goes to the proper noun “أَحْمَد/Ahmed” although it is not the closest antecedent. This is an interesting case showing how the agreement factor has priority over the closest antecedent factor.

(5) قال أَحْمَد١ لـسَمِيرَة٢ أَنَّهُ أَحْضَرَ لـهَا هديَّة

(5) Ahmed<sub>1</sub> told Samira<sub>2</sub> that he<sub>1</sub> had brought her<sub>2</sub> a gift

#### Number Agreement

Number agreement requires that the anaphor and its antecedent must agree in number.

(6.1) كل السيارات<sub>1</sub> في المعرض حمراء ماعدا سيارة<sub>2</sub> لونها<sub>2</sub> اسود

(6.a) All cars<sub>1</sub> in the exhibition are red except one car<sub>2</sub>, its<sub>2</sub> color is black.

(6.2) كل السيارات<sub>1</sub> في المعرض ماعدا سيارة<sub>2</sub> واحدة لونهم<sub>1</sub> ااحمر

(6.b) All cars<sub>1</sub> in the exhibition except one car<sub>2</sub>, their<sub>1</sub> color is black.

In (6.a), “its/ها” refers to “car/سيارة” not to “cars/السيارات” because they are both inanimate objects. But in (6.b) we can notice that “their/هم” is plural pronoun so that it refers to “cars/السيارات” which agrees with it in number.

#### Preconditions/Postconditions

Preconditions and postconditions use real-world knowledge to determine the plausible antecedents to a certain anaphor by checking for some actions that must

have occurred between the anaphor and its antecedent which prove that they denote one and the same object or event as shown in (7).

(7) أحمد١ أعطى تفاحه لـمحمود٢ . هو٢ أكلها عندما شعر بالجوع

(7) Ahmed<sub>1</sub> gave an apple to Mahmoud<sub>2</sub>. He<sub>2</sub> ate it when he felt hungry.

In the above example, “هو/he” refers to “محمود/Mahmoud” because of the postcondition on the verb “يعطي/give” that the actor no longer has the object being given, and the precondition on the verb “يأكل/eat” that the actor must have the item being eaten.

### 8.3.2 Preferential Factors

We now turn to illustrate some of the preferential factors used in anaphora resolution:

#### Syntactic Parallelism

Antecedents must have the same syntactic function as that of the anaphor. This constraint is particularly useful when other constraints or preferences do not point to an unambiguous antecedent.

(8. a) نجح المبرمج في دمج البرولوج<sub>1</sub> مع السي<sub>2</sub> . لكنه دمجها<sub>1</sub> مع البسكال  
من قبل .

(8.a) The programmer successfully combined Prolog<sub>1</sub> with C<sub>2</sub>, but he had combined it<sub>1</sub> with Pascal before.

(8. b) نجح المبرمج في دمج البرولوج<sub>1</sub> مع السي<sub>2</sub> . لكنه دمج البسكال معها<sub>2</sub>  
من قبل .

(8.b) The programmer successfully combined Prolog<sub>1</sub> with C<sub>2</sub>, but he had combined Pascal with it<sub>2</sub> before.

In (8.a) the word “Prolog/برولوج” has the same syntactic function (object) as the pronoun “it/ها”. However in (8b), the word “Prolog/برولوج” does not have the same syntactic function as the pronoun “it/ها”, while the word “C/سي” has the same syntactic function as that of the pronoun “it/ها” (postpositive). This is called syntactic parallelism.

#### Semantic Parallelism

The antecedent must have the same semantic role as the anaphor. This is useful for systems which can automatically identify semantic roles.

(٩.١) أَحْمَد أَرْسَلَ خطاباً إِلَى مُحَمَّدٍ ١ . عَمَر٢ أَيْضًا أَرْسَلَ إِلَيْهِ خطاباً.

(9.a) Ahmed sent a letter to Mohamed1. Omar<sub>2</sub> also sent a letter to him1.

(٩.٢) أَحْمَد١ أَرْسَلَ خطاباً إِلَى مُحَمَّد٢ . هُوَ١ أَيْضًا أَرْسَلَ خطاباً إِلَى عَمَرٍ.

(9.b) Ahmed<sub>1</sub> sent a letter to Mohamed2. He<sub>1</sub> also sent a letter to Omar.

In (9.a), “Ahmed/أَحْمَد” has the semantic role of sender, while “Mohamed/مُحَمَّد” has the semantic role of receiver. The system uses information about the sentence as well as real-world semantic knowledge to obtain the antecedent of the anaphor. So, according to semantic parallelism, the NP which has the same semantic role as the anaphor, is favored. It means that “Mohamed/مُحَمَّد” should be the antecedent of the pronoun “him/هُوَ”.

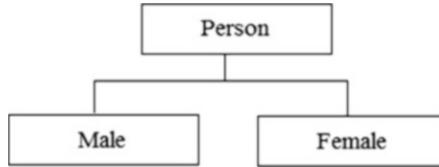
### 8.3.3 *Implementing Features in AR (Anaphora Resolution) Systems*

In a system which uses vector based machine learning techniques, we need to define a set of features to fill the feature vectors. Each AR system uses its own set of features which must be compatible with the technique used, the language of implementation, and the purpose of the system. Choosing the right set of features is necessary for improving the performance of the system. We introduce two examples of systems and the features used in each one to illustrate how the set of features affect the performance of the system. The first system is Soon et al. [44], while the second is Mitkov [28].

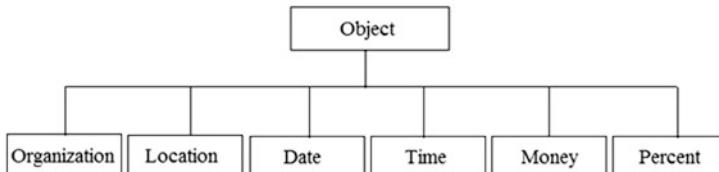
#### Feature-Based Anaphora Resolution System

In the [44] AR System, 12 feature types are used. These features are described as follows:

1. **Distance** feature is used as the number of sentence boundaries between an anaphor and its antecedent. If the anaphor and its antecedent are in the same sentence, the value is 0. If they are one sentence apart, the value is 1; if they are two sentences apart, the value is 2, and so on.
2. **i-Pronoun** feature returns true if the antecedent is a pronoun, else returns false.
3. **j-Pronoun** feature returns true if the anaphor is a pronoun, else returns false.
4. **String Match** feature returns true if the string of the antecedent matches the string of the anaphor, after removing the articles and demonstrative pronouns (a, an, the, this, these, that, those). So that *computer* matches *this computer*, and *the car* matches *this car*.



**Fig. 8.1** Semantic classes used by Soon et al. (example. 1)



**Fig. 8.2** Semantic classes used by Soon et al. (example. 2)

5. **Definite noun phrase** feature returns true if the antecedent is a noun phrase starting with word “the”, else returns false.
6. **Demonstrative Noun Phrase** feature returns true if the antecedent is a noun phrase starting with (that, this, these, those), else returns false.
7. **Number Agreement** feature returns true if the antecedent and anaphor agree in number (both are single or both are plural), else returns false.
8. **Semantic Class Agreement.** Each system can define its own semantic classes; [44] defines semantic classes like “Male”, “Female”, “Person”, “Organization”, “Time”, etc. The semantic classes are arranged in *ISA hierarchy*.<sup>1</sup> As the author mentioned in his work, the hierarchy consists of some semantic classes which are subclasses of a parent semantic class (e.g. Figs. 8.1 and 8.2). Each semantic class is mapped to a WordNet Synset. The anaphor and its candidate antecedent are in agreement if one of them is the parent of the other in the hierarchy, or on the same level. The values returned are true, false, or unknown.
9. **Gender agreement** feature returns true if the anaphor and its antecedent agree in gender, false if they do not agree in gender and unknown if either the anaphor or antecedent’s value is unknown.
10. **Proper-Names** feature returns true if both the anaphor and its antecedent are proper nouns, else returns false.

---

<sup>1</sup>ISA hierarchy, also called “is a” relationship, is an arrangement of items or objects in which the above item is represented as being the parent item for its derived items, and the derived items are represented as children for the above item. In Object Oriented, it means attributes inherited; i.e., if we declare A ISA B, every A entity is also considered to be a B entity. For example: if we have class A = {person} and class B = {male, female}. if B isa A, every entity in B is A, which means every male and female is a person.

**Table 8.1** MUC-6 result to study the contribution of the features

System ID	Recall	Precision	F-measure	Remarks
DSO	58.6	67.3	62.6	Our system
DSO_TRG	52.6	67.6	59.2	Our system using RESOLVE's method of generating positive and negative examples
RESOLVE	44.2	50.7	47.2	The RESOLVE coreference system at the University of Massachusetts

11. **Alias** feature returns true if the anaphor is an alias of the antecedent or vice versa, which occurs when both anaphor and antecedent are named entities that refer to the same entity; else returns false.
12. **Appositive** feature. If the anaphor is in apposition to the antecedent, returns true; else returns false. For example “The president of the central bank” is in apposition to “Mr. John” in the sentence “The president of the central bank, Mr. John said...”

### Determination of the Features

Let us now discuss how the researcher chooses the features used in his system. As stated previously, to build a machine learning based AR System, we have to include a useful and convenient set of features [44]. Researchers take into account that the selected features must be generic enough to be used in a different domain. Also they take into consideration that the features must be able to handle all types of noun phrases and give different coreference decisions based on different types of noun phrases. The researchers use five features to determine the type of the noun phrase: definite noun phrases, demonstrative noun phrases, pronouns, or proper names, and they also use the distance feature to enable the learning algorithm to determine the distribution for the different classes of noun phrases. Also they include knowledge sources which are not too difficult to compute.

### Evaluation

Tables 8.1 and 8.2 show the MUC-6 and MUC-7 results of the system as found in [44]. The tables show the result of this system under system ID “DSO” compared to the coreference system at the University of Massachusetts (RESOLVE). The author mentioned that RESOLVE is the only machine learning-based system among MUC-6 systems which he can directly compare to.

**Table 8.2** MUC-7 result to study the contribution of the features

System ID	Recall	Precision	F-measure	Remarks
DSO	56.1	65.5	60.4	Our system
DSO_TRG	53.3	69.7	60.4	Our system using RESOLVE's method of generating positive and negative examples

### Mitkov's Work on Anaphora Resolution

Ruslan Mitkov [28] discussed the significance of a good set of factors in combination with the strategy used for the system. Mitkov made a case study based on two different approaches using the same set of factors. He reported that while there are a number of approaches that use a similar set of factors, the computational strategies may differ and so the results will also differ. He differentiated the computational strategies by the way the antecedents are computed and tracked down.

Mitkov [28] compared two different approaches using the same factors. The first approach is the Integrated Anaphora Resolution approach (IA) [25], which uses constraints to discount implausible candidates and then uses preferences to rank order the most likely candidates. The second approach is the Uncertainty Reasoning Approach (URA) [26], which employs only preferences on the basis of uncertainty reasoning which estimates the antecedent on the basis of incomplete information, and the assumption that the Natural Language Understanding systems are not able to understand the input completely.

#### Determination of the Features

The factors used in both approaches are: **gender agreement**, **number agreement**, **syntactic parallelism**, **semantic consistency**, **semantic parallelism**, **domain concepts**, **subjects** which prefer the subject of the previous sentence to be the candidate antecedent, **object** preference indicated by verbs, **object** preference indicated by nouns, **repetition** as the repeated NPs are preferred to be the candidate for antecedent, **heading** which means if an NP occurs in the head of the section that contains the current sentence, then consider it as the candidate, **topicalization** which means topicalized structures are given preferential treatment as possible antecedents. For example: in the two sentences: *I won't eat that pizza*, and, *that pizza, I won't eat*; topicalization of the object argument that pizza. And finally, **distance** which prefers the candidates from the previous clause or sentence to be the antecedent.

#### Evaluation

Mitkov's system used a manually annotated corpus contains 113 paragraphs. The result showed a success rate of 83 % for IA over 82 % for URA. Out of the 17 %

**Table 8.3** Evaluation result of Mitkov’s system

System ID	Success rate (%)	Remarks
IA	83	Out of 17 % incorrectly solved anaphors by the IA, 5 % were solved correctly by the URA
URA	82	Out of the 18 % incorrectly solved anaphors by the URA, 4 % were solved correctly by the IA

incorrectly solved anaphors by the IA, 5 % were solved correctly by the URA. Out of the 18 % incorrectly solved anaphors by the URA, 4 % were solved correctly by the IA. The URA went down to level of accuracy of 71 % with a higher threshold of 0.8. Table 8.3 shows the result of Mitkov’s system (cf. Table 8.3).

However although it seems that the IA has a slightly better performance over URA, URA was in general “safer”. Mitkov’s conclusions are:

- In most cases both approaches were correct.
- When information is insufficient, the URA is less “decisive”.
- The IA is more decisive but could be “iffy”.
- When information is ample, the URA is more “confident”.
- The URA is better in cases of gender and number discrepancy between anaphor and antecedent.
- The IA is better in cases where “it” occurs frequently and refers to different antecedents.

After showing the importance of selecting a good set of factors in the AR systems, and how the same set of factors can obtain different results based on the different approaches, there are still some ambiguous questions about factors and their effect on the systems.

Mitkov summarizes these questions in four points at the end of his work [28]: (i) How dependent are factors? (ii) Are preferences better than constraints? (iii) Do factors hold for all genres? and (iv) Which is the best order in which to apply the factors?

## 8.4 The Process of Anaphora Resolution

Most anaphora resolution systems [30, 31, 35] adhere to the following steps:

1. The first step must be the identification of the search scope for the antecedent. All noun phrases (NPs) preceding an anaphor are initially regarded as potential candidates for antecedents; so that the search scope has to be identified. The search scope is the number of sentences or clauses in which we search for the antecedents. Some systems consider all NPs preceding an anaphor, while others impose a maximum number of previous sentences to be considered. Most approaches set their search scope to the current and preceding sentence.

2. After identifying the search scope, we must identify all NPs that precede the anaphor within the search area as candidates for being the antecedents for the specific anaphor.
3. The next step is the extraction of features that describe the selected noun phrases by applying anaphora resolution on those candidates to select the correct antecedent. These features may be lexical, syntactic, semantic, and others. An optional step mentioned in [35] is to determine if the NP is new in the discourse and determine whether it is anaphoric before trying to find the antecedent for it.
4. The final step is scoring and searching candidates. This is the core part of an anaphora resolution system in which the main module processes the features extracted and determines the antecedent of the proposed anaphor. In this step the machine-learning or knowledge-based algorithm is applied.

In the next section we will discuss the approaches to anaphora resolution which may be used in these steps.

## 8.5 Different Approaches to Anaphora Resolution

There are two main approaches to anaphora resolution in terms of the way antecedents are computed and tracked down.

**Traditional approach:** Discount unlikely candidates until a minimal set of plausible candidates is obtained (and then make use of center or focus).

**New statistical approach:** Compute the most likely candidate on the basis of statistical or AI techniques/models.

Much research has been performed in the field of anaphora and coreference resolution especially in the field of pronominal resolution. Research with significant importance includes [3, 20–22, 32, 33]. The cited approaches differ in the set of anaphora that are processed, in the use of syntactic information in the analysis and in the employment of focusing and centering theory techniques.

### 8.5.1 Knowledge-Intensive Versus Knowledge-Poor Approaches

Much of the earlier work in anaphora resolution heavily exploited domain and linguistic knowledge [5, 7, 41, 43], which was difficult both to represent and to process, and which required considerable human input. It encouraged researchers to move away from extensive domain and linguistic knowledge to knowledge-poor anaphora resolution strategies [3, 10, 11, 21, 22, 27, 30, 34, 46].

Knowledge-rich systems require:

1. Domain-specific knowledge.
2. Semantic and discourse knowledge.
3. Sophisticated inference mechanisms.

Knowledge-lean systems, however use:

1. Morphological and shallow syntactic information.
2. They avoid the use of sophisticated knowledge and complex analysis during resolution.

### **Hobbs' Naive Approach**

One of the first (and best) methods relying mostly on syntactical knowledge is Hobbs' algorithm [18,19]. The appeal of this algorithm is that its simplicity yields a respectable performance. Behind the apparent simplicity, however, are some non-trivial assumptions concerning the semantic knowledge provided by the system within which the algorithm runs.

### **Carbonell and Brown Approach (1988)**

Carbonell's approach solves the AR problem with a combination of a set of strategies rather than by a single method. The authors propose a general framework based on multiple knowledge sources, most of which require fairly sophisticated linguistic or world knowledge: sentential syntax, case-frame semantics, dialogue structure, and general world knowledge. The author uses constraints and preferences as follow: local anaphor constraints (agreement constraints) and precondition/postcondition constraints.

Example (10): “John gives Mark an apple. He eats it.” In this example, “he” may refer to “John” or “Mark” but when it is done with real-world knowledge, the postcondition of “give” is that the person no longer has the object being given to him, so “he” must refer to Mark not John. The preferences used are case-role persistence preference, semantic alignment preference, syntactic parallelism preference and syntactic topicalization preference. Shown by the example:

“Mary drove from the park to the club. Peter went there too” (there = club) and  
 “Mary drove from the park to the club. Peter left there too” (in this example, because of the semantics of the verb “to leave”, “there” aligns semantically with “park” and not with “club”).

### **Rich and LuperFoy [38]**

Semantic knowledge is combined with various kinds of information that is typically used in knowledge-poor approaches, such as gender agreement, number agreement,

and recency, as well as information about which entities are globally salient throughout a discourse. Another example of knowledge-intensive anaphora resolution is given by Asher and Lascarides [1], who present a theory called Segmented Discourse Representation Theory.

In the next section we will summarize the most well-known works by grouping them into “traditional” and “alternative” approaches.

### 8.5.2 *Traditional Approach*

#### D. Carter’s Shallow Processing Approach [6]

Carter’s shallow approach mainly depends on the heavy use of syntax, semantic, and local focusing as much as possible regardless of the amount of domain and world knowledge presented. He said in his PhD thesis [6], that reliance on a large amount of world knowledge is hard to process accurately.

Carter’s algorithm resolves anaphora resolution and other linguistics problems using English stories as corpus. The algorithm output paraphrases for each sentence in the story. Carter’s program combines other existing theories such as Sidner’s theory and Wilks’ theory.

#### Rich and LuperFoy’s Distributed Architecture [41]

The distributed structure of Rich and LuperFoy [41] represents a pronominal anaphor resolution system consisting of a loosely coupled set of modules. The authors call these modules a “constraints source” which handles a large number of factors such as recency, number agreement, gender agreement, animacy, disjoint reference, semantic consistency, local focusing, global focus, cataphora and logical accessibility. Each module implements one of the theories of anaphora resolution which contains some constraints that restrict the choice of antecedents.

The “constraints source” consists of four functions; each function returns a score for each candidate. The selection of the correct antecedent depends on the score attached to each candidate given by the constraint source.

### 8.5.3 *Statistical Approach*

#### Nasukawa’s Knowledge-Independent Approach [34]

Nasukawa’s approach to pronoun resolution is relatively independent of external knowledge. His approach is based on the “preference according to existence of identical collocation patterns in the text”, “preference according to the frequency

of repetition in preceding sentences” as well as “preference according to syntactic position”.

Unlike some approaches which use world knowledge in finding the antecedent of an anaphor, Nasukawa’s approach uses collection patterns to determine that the frequency in preceding sentences of a noun phrase with the same lemma as the candidate, is an indication for preference when selecting an antecedent.

For example, if the statement “He moved his residence” is found in the discourse, the information works as a selectional constraint by indicating that the word “residence” can be the object of the verb “move”.

### **Statistical/Corpus Processing Approach [10,11]**

Dagan and Itai [10, 11] follow a corpus-based approach for disambiguating pronouns. A table lists the patterns produced by substituting each candidate with the anaphor and the number of times each of these patterns occurs in the corpus. Based on the frequency of these patterns, the most likely candidate is selected as the antecedent of the anaphor. According to this statistic, the most likely candidate is preferred as the antecedent of the anaphor.

The authors illustrate their approach on a sentence taken from the Hansard corpus:

They knew full well that the companies held tax money<sub>1</sub> aside for collection later on the basis that the government<sub>2</sub> said it<sub>2</sub> was going to collect it<sub>1</sub>.

In this example, there are two occurrences of the anaphor “it”. The frequency of the three candidates: “money”, “collection”, and “government” are determined. Then a table would list the patterns produced by substituting each candidate with the anaphor, and the number of times each of these patterns occurs in the corpus. The definite semantic constraints eliminate all the candidates but the correct one.

#### **8.5.4 Linguistic Approach to Anaphora Resolution**

Linguists have made a principled distinction between two types of noun phrases: lexical NPs and anaphors. The analogy in computer science, to a certain extent, is like the distinction between constants and variables where a constant usually has a fixed value while the value of a variable can change in the program. Similarly, a lexical NP like “Barak Obama” is a noun phrase that refers only to itself and its value is not likely to change. In contrast, anaphors such as pronouns and reflexives do not have intrinsic specific reference but they acquire their reference from context. For example, the referent of the anaphor “I” depends on who says it. It will have a different referent every time a different person utters the pronoun “I”. The widely held view is that anaphors have antecedents that they refer to and that it is usually the most recent antecedent that an anaphor refers to. However, there are cases

when the referent precedes the anaphor as in the following sentence that shows the pronoun “he” precedes the referent “Shafiq”. This situation is usually referred to as “cataphora” to distinguish it from cases of “anaphora” where the referent precedes the anaphor.

(11) عندما علم بأنه 1 استبعد من قائمه المرشحين ، قدم شفيق 1 تلتمسا

(11) When he knew that he1 was eliminated from the candidates' list, Shafiq1 submitted a petition.

NLP applications are concerned with all types of coreference whether we are dealing with anaphora or cataphora.

Linguists are interested in determining the conditions under which an anaphor and a lexical NP can be coreferential or disjoint in reference. Chomsky [8] made a principled distinction between three types of noun phrases: pronominals such as I, you, he etc., reflexives like yourself, herself, themselves, etc., and R-expressions which are lexical NPs like “John”, “the man”, or “Seattle”. Chomsky [8] claims that there are universal principles that govern their behavior.

He defines these principles in his theory of government and binding [8, 9]. The principles account for the different behavior of pronominals and reflexives. It can be stated as follows:

- **Principle A:** an anaphor (a reflexive or a reciprocal such as each other) must be bound in its governing category (its clause).
- **Principle B:** a pronominal must be free (not bound) in its governing category (its clause).
- **Principle C:** an R-expression must be free (not bound).

A governing category is the clause containing the governor and its governee. The following sentences show how the principles of the government and binding theory work.

(12) الرئيس مرسى قبله أمس.

(12) President Morsy met him yesterday.

(13) الرئيس مرسى قبل الرئيس مرسى أمس.

(13) President Morsy met President Morsy yesterday.

(14) أحاط الرئيس مرسى نفسه بجموعة من المستشارين.

(14) President Morsy surrounded himself with a group of advisors.

In (12) “President Morsy” and “him” must be disjoint in reference. They cannot be referring to the same entity. This follows naturally from Chomsky's principle A which states that a pronominal such as “him” must be free in its clause. So without looking into the context and on purely structural grounds we were able to determine that in (12) the pronominal “him” does not refer to the otherwise, possible antecedent “President Morsy”. In contrast, Principle A predicts that in (14) the reflexive must be bound and therefore “himself” and “President Morsy” must

be coreferential which is the case. The government and binding theory predicts that (13) is ungrammatical because it violates Principle C which requires that an R-expression such as “President Morsy” be free in its clause. In (14) it is not free because there is another identical R-expression that it refers to. The interpretation of coreference and disjoint reference between the antecedents and the anaphors in (12), (13) and (14) does not depend on our real-world knowledge or even the linguistic contexts. It is determined solely on the basis of the type of anaphor, in this case a pronominal versus a reflexive, and the linguistic structure in which both the anaphor and the antecedent occur. Farghaly [13] tested the government and binding theory against data from Egyptian Arabic and showed that in most cases these principles hold for Egyptian Arabic as well.

The linguistic approach though interesting, is not easy to implement in NLP applications. It requires deep linguistic analysis to determine complex linguistic constructs such as c-command, governors, governee and governing categories [13] which casts serious doubts on the practicality of implementing deep theoretical constructs in current NLP applications. However, Arabic statistical models for mention detection and chaining [50] show that incorporating lexical and syntactic features improve the precision of such systems.

## 8.6 Recent Work in Anaphora and Coreference Resolution

Actually there has been a lot of work on Anaphora and coreference resolution since the 1980s, but it is not possible to cover all the work on AR in a single chapter. In this section we will discuss some of the recent work in anaphora and coreference resolution.

### 8.6.1 Mention-Synchronous Coreference Resolution Algorithm Based on the Bell Tree [24]

Luo et al. [24] propose a new approach for coreference resolution using the Bell tree. The approach uses the Bell tree algorithm to find the best path from the root of the tree to the leaf node, then the maximum entropy algorithm is applied to compute the probability of these paths. The system uses the same terminologies used in ACE [37]. The term *mention* is used to describe the instance of a reference to an object, while the term *entity* describes the collection of mentions to the same object. For example, in the sentence:

(15) صرخ الرئيس<sub>1</sub> محمد مرسى<sub>1</sub> أنه<sub>1</sub> سوف يعيد الحقوق إلى أصحابها.

(15) President<sub>1</sub> Mohammed Morse<sub>1</sub> said that he<sub>1</sub> will restore the rights to their owners.

“President/الرئيس”, “Mohammed Morse/محمد مرسى”, and “he/هـ” belong to the same entity as they refer to the same object.

**Table 8.4** Basic features used in maximum entropy model

Category	Features	Remark
Lexical	Exact_strm	1 if two mentions have the same spelling; 0 otherwise
	Left_subsm	1 if one mention is a left substring of the other; 0 otherwise
	Right_subsm	1 if one mention is a right substring of the other; 0 otherwise
	Acronym	1 if one mention is an acronym of the other; 0 otherwise
	Edit_dist	Quantized editing distance between two mention strings
	ncd	Number of different capitalized words in two mentions
	Spell	Pair of actual mention strings
	token_dist	How many tokens two mentions are apart (quantized)
Distance	Sent_dist	How many sentences two mentions are apart (quantized)
	gap_dist	How many mentions in between two mentions in question (quantized)
Syntax	Apposition	1 if two mentions are appositive; 0 otherwise
	POS_pair	POS-pair of two mention heads
Count	Count	Pair of (quantized) numbers, each counting how many times a mention string is seen
Pronoun	Gender	Pair of attributes of {female, male, neutral, unknown}
	Number	Pair of attributes of {single, plural, unknown}
	Possessive	1 if pronoun is possessive; 0 otherwise
	Reflexive	1 if pronoun is reflexive; 0 otherwise

The process of forming entities from mentions can be represented by a tree structure. The root node represents the first mention in the document, while the second mention is added to the tree either by linking to an existing node or starting a new entity. Subsequent mentions are added to the tree in the same manner. Each leaf node in the Bell tree represents a possible coreference outcome.

The process of coreference resolution can be cast as finding the “best” leaf node, or finding the best path from the root of the tree to the leaf node. For the coreference resolution process, the maximum entropy algorithm is used to compute the linking probability between a partial entity and a mention. Table 8.4 illustrates the basic features used in a maximum entropy model. For further details on Luo’s work, the reader is referred to [24].

### 8.6.2 A Twin-Candidate Model for Learning-Based Anaphora Resolution [47, 48]

The main idea behind the twin-candidate learning model is to recast anaphora resolution as a preference classification problem. The purpose of the classification is to determine the preference between two competing candidates for the antecedent of a given anaphor.

**Table 8.5** Basic features used in twin-candidate system

Features	Remark
Ana_Reflexive	Whether the anaphor is a reflexive pronoun
Ana_PronType	Type of the anaphor if it is a pronoun (he, she, it, or they)
Candi_Def	Whether the candidate is a definite description
Candi_Indef	Whether the candidate is an indefinite NP
Candi_Name	Whether the candidate is a named entity
Candi_Pron	Whether the candidate is a pronoun
Candi_FirstNP	Whether the candidate is the first mentioned NP in the sentence
Candi_Subject	Whether the candidate is the subject of a sentence, the subject of a clause, or not
Candi_Object	Whether the candidate is the object of a verb, the object of a preposition, or not
Candi_ParallelStruct	Whether the candidate has an identical collocation pattern with the anaphor
Candi_SentDist	The sentence distance between the candidate and the anaphor
Candi_NearestNP	Whether the candidate is the candidate closest to the anaphor in position

Yang et al. [47, 48] propose a learning model for anaphora resolution called the “twin-candidate model”. The baseline system is built based on the single-candidate model.

The single-candidate model assumes that if  $i$  is an anaphor, and  $C$  is a candidate antecedent for that anaphor, the probability that  $C$  is an antecedent for  $i$  is only dependent on the anaphor  $i$  and the antecedent  $C$ , and is independent of all other candidates. The assumption that the candidate antecedent is independent of all other candidates makes the process of selecting the correct antecedent unreliable, since the correct antecedent should be selected as the “best” one among the set of candidates.

To address this problem, Yang et al. present a so-called Twin-candidate model in which antecedent candidates compete against each other in pairs and the one that wins the most competitions is selected as antecedent.

The system uses positional, lexical and syntactic features described in Table 8.5. A vector of features is specified for each training instance. The features may describe the characteristics of the anaphor and the candidate, as well as their relationships.

### 8.6.3 *Improving Machine Learning Approaches to Coreference Resolution [36]*

Ng and Cardie [36] contributed two types of extensions to the approach followed by Soon et al. [44]. First, they propose three extra-linguistic modifications to the machine learning framework. Second, they improve the performance of Soon et al.’s [44] corpus-based system by expanding the feature set from 12 to 53 features

that include a variety of linguistic constraints and preferences. They use a decision tree induction system to train the classifier to decide whether two given NPs are coreferent or not. After the training step, a clustering algorithm is applied to create one cluster for each set of coreferent NPs.

The three changes to the general machine learning framework are:

1. A best first technique, which selects the candidate with the highest coreference likelihood value from among the preceding NPs instead of selecting the closest antecedent.
2. A different training model is used. Instead of creating a positive training model for each anaphor and its closest antecedent, it is created for the anaphor and its most confident antecedent.
3. The third change is the modification of the string match feature in Soon et al. The algorithm splits this feature into several primitive features which give the learning algorithm more flexibility in creating the coreference rule.

Ng and Cardie mentioned that the result reflects decreased performance when using the full feature set. They addressed the problem by manually selecting a subset of 22–26 features only, which resulted in a significant gain in performance. The modifications to the general machine learning framework boost the performance of Soon’s coreference resolution approach from an F-measure of 62.6–70.4.

## 8.7 Evaluation of Anaphora Resolution Systems

Evaluating the performance of the anaphora resolution system is an extremely important aspect of the algorithm performance. Several metrics have been proposed for the task of scoring coreference resolution, and each of them presents advantages and drawbacks.

From among the different scoring measures that have been developed, we will discuss the four most widely used measures: MUC [45], B3 [2], CEAF [23], and the ACE-value [12] and the newly introduced measure BLANC [40]. The first four measures have been widely used, while BLANC is a proposal of a new measure that is interesting to test. The researchers have employed these measures to compare their results with previous works.

### 8.7.1 MUC [45]

The Message Understanding Conference (MUC) program was the first to define a scoring metric, known as the MUC metric [45]. MUC was the first and most widely used measure.

The MUC initiatives introduced the measures for recall and precision. Also the F-measure, precision rate and success rate are used as part of the MUC-6 and MUC-7 evaluation tasks on coreference and anaphora resolution [4, 16].

MUC-1 (1987) was basically exploratory; and there was no formal evaluation. MUC-2 (1989) was the first to define the primary evaluation measures of recall and precision. It presents a template with 10 slots to be filled with information from the text. In MUC-3 (1991) the template becomes more complex (18 slots). MUC-4 (1992) used the same tasks as MUC-3 but with some increase in template complexity. The innovation of MUC-5 (1993) was the use of a nested template structure. In earlier MUCs, each event had been represented as a single template. At MUC-6 (1995), Named Entities Recognition and Coreference resolution tasks were added.

In anaphora, Precision is defined as:

$$P = \frac{\text{number of correctly predicted antecedents}}{\text{number of all predicted antecedents (correctly or false predicted)}} \quad (8.1)$$

And Recall as:

$$R = \frac{\text{number of correctly predicted antecedents}}{\text{number of all correct antecedents (predicted or missed)}} \quad (8.2)$$

F-measure is calculated as the weighted mean of precision and recall as:

$$F = \frac{2 \times P \times R}{P + R} \quad (8.3)$$

Therefore, if the approach is robust and proposes an antecedent for each pronominal anaphor, the success rate would be equal to both recall and precision.

$$\text{SuccessRate} = \frac{\text{successfully resolved anaphors}}{\text{number of all anaphors}} \quad (8.4)$$

Although recall and precision are very important in measuring the efficiency of an anaphora resolution system, they cannot alone provide a comprehensive overall assessment of the system. It is necessary to assess the system against other benchmarks.

Bagga and Baldwin [2] and Luo [23] observed two main shortcomings in MUC measure: First, it only takes into account coreference links and ignores single-mention entities (entities that occur in chains consisting only of one element, the entity itself) so no link can be found in these entities.

Second, all errors are considered to be equal. The MUC scoring algorithm penalizes the precision numbers equally for all types of errors. However some errors are more significant than others. It favors systems that produce fewer coreference chains which may result in higher F-measures for worse systems. Also there is

no specific scoring scheme proposed to evaluate anaphora resolution other than coreference resolution.

### 8.7.2 *B-Cube* [2]

Bagga and Baldwin introduce the B-Cube metric to address the shortcomings of MUC discussed earlier.

It successfully overcomes the shortcoming of the MUC-6 algorithm by calculating the precision and recall numbers for each entity in the document including singletons and does not care whether the entity is part of a coreference chain or not. It then takes the weighted sum of these individual precisions and recalls as the final metric.

Also it takes into account that the error of linking the two large chains in the second response is more damaging than the error of linking one of the large chains with the smaller chain in the first response.

For entity  $i$ , B-Cube defines the precision and recall as:

$$\text{Precision} = \frac{\text{number of correct elements in the output chain containing entity}_i}{\text{number of elements in the output chain containing entity}_i} \quad (8.5)$$

$$\text{Recall} = \frac{\text{number of correct elements in the output chain containing entity}_i}{\text{number of elements in the truth chain containing entity}_i} \quad (8.6)$$

The final precision and recall numbers are calculated by the following formulas:

$$\text{FinalPrecision} = \sum_{i=1}^N W_i \times \text{Precision}_i \quad (8.7)$$

$$\text{FinalRecall} = \sum_{i=1}^N W_i \times \text{Recall}_i \quad (8.8)$$

I refer the reader to [2] for a detailed explanation of the B-Cube algorithm.

### 8.7.3 *ACE (NIST 2003)*

The terminology used in the Automatic Content Extraction (ACE) task [37] calls each individual phrase a mention and equivalence class an entity. We should mention that the ACE entity is called the *coreference chain* or *equivalence class* in MUC, and the ACE mention is called an entity in MUC.

In the ACE task, a value-based metric called ACE-value [38] is used. The counted number of errors (a missing element, a misclassification of a coreference chain, a mention in the response not included in the key) is used to compute the ACE-Value. Each error is associated with cost, which depends on the type of entity (e.g. person, location, organization) and on the kind of mention (e.g. name, nominal, pronoun). The sum of the three costs yields the total cost. This cost is normalized against the cost of a nominal system that does not output any entity. The final ACE-Value is computed by subtracting a normalized cost from  $(1 - \text{normalized cost})$ . A robust system will get 100 % ACE-Value, while a system with no output will get 0 % ACE-Value. Although calculating ACE-Value in such way avoids the shortcoming of the MUC F-measure, it has its own shortcomings. ACE-Value is very task-specific and not really useful for a general coreference problem that is not limited to a set of specific semantic types.

Another problem mentioned in [23] is that the ACE-Value is hard to interpret. A system with 90 % does not mean that 90 % of the system entities or mentions are correct, but that the cost of the system, relative to the one producing no entity, is 10 %. For more details about the ACE program, the reader is referred to (NIST 2003).

#### 8.7.4 CEAf [23]

Luo [23] observes that coreference systems are to recognize entities and proposes a metric called Constrained Entity-Aligned F-Measure (CEAF).

The metric finds the best one-to-one entity alignment between the subsets of reference (GOLD)<sup>2</sup> and system responses (SYS) before computing precision and recall. Each SYS entity is aligned with at most one GOLD entity and vice versa. The best mapping is that which maximizes the similarity over pairs of chains. Once the total similarity is defined, it is straightforward to compute recall, precision and F-measure. Luo [23] proposes two definitions of CEAF: mention-based CEAF and entity-based CEAF.

He defined them as: mention-based CEAF which reflects the percentage of mentions that are in the correct entities; entity-based CEAF which reflects the percentage of correctly recognized entities [23].

Luo [23] overcomes the B-cube's problem where the response with all mentions in the same chain obtains 100 % recall, whereas a response with each mention in a different chain obtains 100 % precision. The constraint imposed in the CEAF entity alignment makes it impossible to get this error as a system that outputs too many entities will be penalized in precision while a system that outputs too few entities

---

<sup>2</sup>The set of mentions contained in the gold standard, produced by a human expert, are referred to as TRUE or GOLD mentions, as opposed to the set of mentions contained in the system output, which are called SYSTEM or SYS mentions. Gold standard annotation is correctly identifying all NPs that are part of coreference chains.

will be penalized in recall. Also a perfect system will get an F-measure of 1 and a system with no output will get an F-measure of 0.

Although CEAF is a good evaluation measure, it also has its own problems. It suffers from the same singleton problem as B-cube. Because CEAF and B-Cube allow the annotation of the singleton in the corpus, and consider it in the process of resolution, their scores are higher than that of the MUC simply because a great percentage of the score is due to the resolution of singletons.

Both the B-Cube and CEAF could be computed with a mention-weighted version such that it gives a unique score for each mention, allowing an entity with more mentions to have a higher score than one with fewer mentions.

### 8.7.5 **BLANC** [40]

Recasens and Hovy [40] introduce a new coreference resolution evaluation algorithm: BiLateral Assessment of Noun Phrase Coreference (BLANC). Their algorithm makes use of the existing Rand index [39] which is an evaluation metric that measures the similarity between two partitions or clusters. We will not focus on the Rand index algorithm here, as one can read more about it in [39]. BLANC is proposed to overcome the shortcomings of the above-mentioned measures (MUC, B-Cube, and CEAF).

BLANC introduces a new measure that considers two types of decisions:

1. Coreference decision which takes coreference and also non-coreference links into account. A coreference link is the link that holds between every two mentions that corefer, while a non-coreference link holds between every two mentions that do not corefer.
2. Correctness decision: when the system is correct, the Right link has the same value in Gold and SYS for coreference and non-coreference links, and if the system is wrong the Wrong link does not have the same value in Gold and SYS for coreference and non-coreference links.

We notice that BLANC balances coreference and non-coreference links equally. In this way, singletons are neither ignored (shortcoming of MUC) nor given greater importance than multi-mention entities (shortcoming of CEAF and B-cube). However, a basic assumption behind BLANC is that the sum of all coreferential and non-coreferential links is constant for a given set of mentions.

## 8.8 Anaphora in Semitic Languages

Many natural language processing applications require deep knowledge of the language, and hence transferring a technology from one language to another requires intensive effort and substantial resources.

Moreover, many problems are unique to Semitic languages, which in general are less investigated; the result is that language processing is less advanced for Semitic languages than it is for other languages.

Building anaphora resolution systems for Semitic languages is not so different from building such systems for other Latin languages. The steps to building any AR system are the same for Semitic languages but for each step we have to consider the special characteristics of the languages. First, we have to prepare corpora annotated with anaphoric links and take into account the characteristics of the language of these corpora. For example, Semitic languages have very complex morphology [15]. Both prefixes and suffixes combine with a single base stem, such as “**لَبِّيَ**/loves her”, as we have seen. The single word “**لَبِّيَ**/loves her” contains a verb and a pronoun, which is an anaphor. In this case we need an additional step in the annotation process to segment the words into tokens and check if the token is an anaphor, an antecedent or none of them. The preprocessing step (including corpora annotation, POS tagging, tokenization, text disambiguation and other preprocessing) contains most of the language-specific work. Basically the syntactic, morphological and semantic features related to the Semitic languages are considered and processed in this step.

After preprocessing the text, the steps in building any AR system are the same for Semitic languages and statistical approach techniques can be used to build the base line system for Semitic language AR systems just like any other Latin systems. In the next section, we will discuss some previous work on the Arabic language which is one of the five Semitic languages (see Chap. 1).

### **8.8.1 Anaphora Resolution in Arabic**

#### **Multilingual Robust Anaphora Resolution Approach [30]**

We present here a brief description of [30]. The approach used in this work consists of a modification of Mitkov’s approach [29], and operates on texts pre-processed by a part-of-speech tagger. The system checks input against number and gender agreement of an antecedent indicator. A score is set for each candidate according to each indicator, and the candidate with the highest score is returned as the antecedent. The indicators are used in the same way for English, Polish and Arabic. The approach was initially developed and tested for English, but it has been subjected to some modifications to work with Polish and Arabic.

The approach resolves pronoun anaphora resolution by making use of corpus-based NLP techniques such as sentence segmentation, part-of-speech tagging, noun phrase identification and the high performance of the antecedent indicators.

The detected noun phrases are passed on to a gender and number agreement test. Agreement rules in Arabic are different from those in Latin languages. For instance, a set of non-human items (animals, plants, objects) is referred to by a

**Table 8.6** Evaluation of Mitkov’s system

System ID	Success rate (%)
Arabic direct	90.5
Arabic improved	95.2

singular feminine pronoun, such as the pronoun “ها/its” in the sentence “لۇزىھا جىيل“/رأيت فراشات/I saw butterflies their color is beautiful”, which refers to “فراشات“/butterflies”. But the same pronominal suffix in Arabic can also refer to a singular feminine object. Thus, an Arabic AR system has to disambiguate such a pronoun. Moreover, the same pronoun in Arabic may appear as a suffix of a verb (e.g. /فعلها/do it), or as a possessive pronoun when it is attached to a noun (e.g. كتابهـ/his book), or as an object of a preposition (e.g. فيـ/in it). In particular, pronouns have several types in Arabic. Each type is divided into many categories.

Mitkov [30] uses a set of antecedent indicators which are used for tracking the correct antecedent from a set of candidate antecedents. Antecedent indicators have been identified on the basis of empirical studies of numerous hand-annotated technical manuals. A score is set for each candidate according to each antecedent indicator (preferences), with a value (-1, 0, 1, or 2). The candidate with the highest score is proposed as the antecedent.

This robust approach for Arabic is evaluated in two modes. The first mode consists of using the approach without any specific modification or enhancement (referred to as “Arabic direct” in Table 8.6), whereas the second mode uses an enhanced version which includes rules to capture some specific characteristics of Arabic plus additional indicators for Arabic features.

The evaluation for Arabic showed a very high “critical success rate” as well. The robust approach used without any modification scored a “critical success rate” of 78.6 %, whereas the improved Arabic version scored 89.3 %.

For a detailed description of the system, refer to [30].

### Arabic Mention Detection and Coreference Resolution [49, 50]

Zitouni et al. [49, 50] presented a statistical-based mention detection system. Mention detection is a very important task in many NLP systems like data mining, question answering, summarization, etc. It is also very central to anaphora and coreference resolution since mention detection is about finding the different ways one conceptual entity is referred to.

Zitouni et al. [49, 50] discussed their Arabic entity detection and recognition system (EDR) for the Arabic language. The system is statistical and built around maximum entropy, and works under the ACE 2004 framework. The EDR system proceeds in two main phases: the mention detection phase, and the coreference resolution phase to corefer the detected mentions into entities. The approach is modified to accommodate the special characteristics of the Arabic language.

**Table 8.7** The result of Zitouni et al.'s system

	Base		Base + Stem	
	ECM-F	ACE-val	ECM-F	ACE-val
Truth	77.7	86.9	80.0	88.2
System	62.3	61.9	64.2	63.1

The authors explain the additional complexity involved in building the system due to the challenges in Arabic such as the different forms of the Arabic words that result from the derivational and inflectional processes, as most prepositions, conjunctions, pronouns, and possessive forms are attached to the Arabic surface word.

The coreference resolution system is similar to the one described in Luo et al. [24]. The first step in building the system is the segmentation process. The text is segmented into words using white space delimiters, and then every single word is separated into prefix-stem-suffix. The phase of coreference resolution groups the mentions referring to the same entity. The system uses the Bell-tree algorithm described in Luo et al. [24]. The root of the tree contains the first mention in the document. The second mention is added to the tree either by linking to the existing entity, or by creating a new entity. The other mentions in the documents are added to the tree in the same way.

The next step in the coreference resolution phase is the maximum entropy algorithm. The maximum entropy algorithm is used to compute the probability of an active mention  $m$  linking with an in-focus partial entity  $e$ .

The authors introduced a set of features to be used by the maximum entropy algorithm, such as the relationship between an entity and a mention (or a pair of mentions) which could be characterized by certain features. There are two types of features used, entity-level and mention-pair. The entity-level features capture some characteristics such as gender and number agreement, while the mention-pair features encode the lexical features. The results are reported with ACE-value (NIST 2003) and ECM-F [24], and are shown in Table 8.7.

For details on the EDR system, the reader is referred to “A Statistical Model for Arabic Mention Detection and Chaining” in [14].

## 8.9 Difficulties with AR in Semitic Languages

### 8.9.1 *The Morphology of the Language*

Semitic languages have both prefixes and suffixes and sometimes both kinds of affixes combine with a single base form (as in Hebrew: the verb inflection *guard*, second person, plural, future tense; furthermore, the base form itself can be modified in the different paradigms).

Zitouni et al. [49, 50] explain how Arabic is a highly inflected language. Arabic nouns encode information about gender, number, and grammatical cases. Seddik et al. [42] state that pronouns have many types in Arabic, and each type is divided into many categories. For example a pronoun in Arabic can be personal, possessive, relative, or demonstrative, etc. In addition Arabic nouns must have gender: masculine or feminine. Number in Arabic has three classifications: singular, dual, and plural, and finally there are three grammatical cases: nominative, genitive, and accusative.

### 8.9.2 Complex Sentence Structure

Unlike many other languages, Arabic may combine the verb, subject, and object in one word such as (أَنْجَيْنَاهُمْ) which means “we saved them”. We have to break up the phrase, and identify the subject before starting the process of anaphora resolution.

### 8.9.3 Hidden Antecedents

In some cases, especially in Quranic texts, the pronoun may refer to something which is not presented in the text, such as in Example 16. The pronoun (هو) refers to “Allah”, which is not presented in the text. The human mind can determine the hidden antecedent by using the knowledge that Allah is the only one who knows the unseen. But for the anaphora resolution algorithm, it could be difficult to recognize a hidden antecedent.

(١٦) وَعِنْهُ مَفَاتِحُ الْغَيْبِ لَا يَعْلَمُهَا إِلَّا هُوَ.

(16) With Him<sub>1</sub> are the keys of the unseen, the treasures that none know it but He<sub>1</sub>.

In this example, the pronouns “Him/ه” and “He/ه” are ambiguous. The pronouns refer to “Allah/الله”, which is not presented in the text.

### 8.9.4 The Lack of Corpora Annotated with Anaphoric Links

The major problem in anaphora resolution is the lack of corpora annotated with anaphoric relations for Semitic languages although it is very much needed in most NLP systems. The annotation task of anaphoric relations is very time consuming and requires a significant effort from the human annotator. To the best of our knowledge there is no available resource for Arabic except Hammami et al. [17].

## 8.10 Summary

In this chapter we have discussed anaphora and anaphora resolution, and the different types of anaphora. We have described the two types of factors used for anaphora resolution: “eliminating” and “preferential”, also we have shown how an optimal set of factors can increase the performance of a system. Two examples of systems that use different set of features are introduced to show how these features affect the performance of the systems. Then we presented the general steps of an anaphora resolution system. The techniques for anaphora resolution constitute an important topic introduced in the chapter. We have classified the systems in terms of the way the antecedents are computed and tracked down, and therefore divided the systems into traditional and statistical types. Previous works have been introduced to show the differences among the approaches and how it is very important to select the appropriate approach when designing an AR system.

We also discussed the evaluation of anaphora resolution systems. We started by giving a brief history of several coreference tasks and described evaluation metrics such as MUC, ACE-Value, CEAF, and others. Then we focused on anaphora resolution in Semitic languages, and showed how Semitic languages differ from other European languages because of their complex structure and their unique features. Then we introduced the difficulties and challenges in AR in those languages. One problem that AR suffers from in all Semitic languages is the scarcity of annotated corpora in anaphora resolution.

## References

1. Asher N, Lascarides A (2003) Logics of conversation. Cambridge University Press, Cambridge/New York
2. Bagga A, Baldwin B (1998) Algorithms for scoring coreference chains. In: Proceedings of the linguistic coreference workshop at the first international conference on language resources and evaluation (LREC'98), Granada, pp 563–566
3. Baldwin B (1997) CogNIAC: high precision coreference with limited knowledge and linguistic resources. Proceedings of the ACL'97/EACL'97 workshop on operational factors in practical, robust anaphora resolution, Madrid, pp 38–45
4. Baldwin B, Morton T, Bagga A, Baldrige J, Chandraseker R, Dimitriadis A, Snyder K, Wolska M (1998) Description of the UPENN CAMP system as used for coreference. In: Proceedings of message understanding conference (MUC-7), Fairfax
5. Carbonell JG, Brown RD (1998) Anaphora resolution: a multi-strategy approach. In: COLING'88 proceedings of the 12th conference on computational linguistics, Budapest, vol 1. Association for Computational Linguistics, pp 96–101
6. Carter DM (1986) A shallow processing approach to anaphor resolution. PhD thesis, University of Cambridge
7. Carter DM (1987) A shallow processing approach to anaphor resolution. Computer Laboratory, University of Cambridge
8. Chomsky N (1981) Lectures on government and binding. Foris Publications, Dordrecht/Cinnaminson

9. Chomsky N (1986) Knowledge of language: its nature, origin and use. Greenwood Publishing Group, USA
10. Dagan I, Itai A (1990) Automatic processing of large corpora for the resolution of anaphora references. In: Proceedings of the 13th international conference on computational linguistics (COLING'90), Helsinki, vol 3, pp 1–3
11. Dagan I, Itai A (1991) A statistical filter for resolving pronoun preferences. In: YA Feldman, A Bruckstein (eds) Artificial intelligence and computer vision. Elsevier, Burlington, pp 125–135
12. Doddington G, Mitchell A, Przybocki M, Ramshaw L, Strassel S, Weischedel R (2004) The automatic content extraction (ace) program tasks, data, and evaluation. In: NIST, Lisbon, pp. 837–840
13. Farghaly A (1981) Topic in the syntax of Egyptian Arabic. PhD dissertation, University of Texas at Austin, Austin
14. Farghaly A (2010) Arabic computational linguistics. CSLI Publications, Center for the Study of Language and Information, Stanford
15. Farghaly A, Shaalan Kh (2009) Arabic natural language processing: challenges and solutions. ACM Trans Asian Lang Inf Process 8(4):1–22. Article 14
16. Grishman R, Sundheim B (1996) Message understanding conference – 6: a brief history. In: Proceedings of the 16th international conference on computational linguistics (COLING), Copenhagen, vol I, pp 466–471
17. Hammami S, Belguith L, Ben Hamadou A (2009) Arabic anaphora resolution: corpora annotation with coreferential links. Int Arab J Inf Technol 6:480–488
18. Hobbs JR (1976) Pronoun resolution. Research report. Department of Computer Science, University of New York, New York, pp 76–1
19. Hobbs JR (1978) Resolving pronoun references. Lingua 44:339–352
20. Kameyama M (1997) Recognizing referential links: an information extraction perspective. In: Proceedings of the ACL'97/EACL'97 workshop on operational factors in practical, robust anaphora resolution, Madrid, pp 46–53
21. Kennedy Ch, Boguraev B (1996) Anaphora for everyone: pronominal anaphora resolution without a parser. In: Proceedings of the 16th international conference on computational linguistics (COLING'96), Copenhagen, pp 113–118
22. Lappin Sh, Leass H (1994) An algorithm for pronominal anaphora resolution. Comput Linguist 20(4):535–561
23. Luo X (2005) On coreference resolution performance metrics. In: Proceedings of the conference on human language technology and empirical methods in natural language processing (HLT'05), Vancouver. Association for Computational Linguistics, Stroudsburg, pp 25–32. <http://dl.acm.org/citation.cfm?id=1220579>
24. Luo X, Ittycheriah A, Jing H, Kambhatla N, Roukos S (2004) A mention synchronous coreference resolution algorithm based on the bell tree. In: Proceedings of ACL'04, Barcelona
25. Mitkov R (1994) An integrated model for anaphora resolution. In: Proceedings of the 15th conference on computational linguistics (COLING'94), Stroudsburg, vol 2. Association for Computational Linguistics, pp 1170–1176
26. Mitkov R (1995) An uncertainty reasoning approach for anaphora resolution. In: Proceedings of the natural language processing pacific rim symposium (NLPERS'95), Seoul, pp 149–154
27. Mitkov R (1996) Anaphora resolution: a combination of linguistic and statistical approaches. In: Proceedings of the discourse anaphora and anaphor resolution (DAARC'96), Lancaster
28. Mitkov R (1997) Factors in anaphora resolution: they are not the only things that matter. A case study based on two different approaches. In: Proceedings of the ACL'97/EACL'97 workshop on operational factors in practical, robust anaphora resolution, Madrid, pp 14–21
29. Mitkov R (1998a) Evaluating anaphora resolution approaches. In: Proceedings of the discourse anaphora and anaphora resolution colloquium (DAARC'2), Lancaster
30. Mitkov R (1998b) Robust pronoun resolution with limited knowledge. In: Proceedings of the 18th international conference on computational linguistics (COLING'98)/ACL'98 conference, Montreal, pp 869–875

31. Mitkov R (1999) Anaphora resolution: the state of the art. Technical report based on COLING'98 and ACL'98 tutorial on anaphora resolution, School of Languages and European Studies, University of Wolverhampton
32. Mitkov R, Belguith L, Stys M (1998) Multilingual robust anaphora resolution. In: Proceedings of the third international conference on empirical methods in natural language processing (EMNLP-3), Granada, pp 7–16
33. Mitkov R, Lappin S, Boguraev B (2001) Introduction to the special issue on computational anaphora resolution. MIT, Cambridge, pp 473–477
34. Nasukawa T (1994) Robust method of pronoun resolution using full-text information. In: Proceedings of the 15th international conference on computational linguistics (COLING'94), Kyoto, pp 1157–1163
35. Ng V (2003) Machine learning for coreference resolution: recent successes and future challenges. Technical report cul.cis/tr2003-1918, Cornell University
36. Ng V, Cardie C (2002) Improving machine learning approaches to coreference resolution. In: Proceedings of the 40th annual meeting of the association for computational linguistics (ACL), Philadelphia, pp 104–111
37. NIST (2003a) The ACE evaluation plan. [www.nist.gov/speech/tests/ace/index.htm](http://www.nist.gov/speech/tests/ace/index.htm)
38. NIST (2003b) Proceedings of ACE'03 workshop, Adelaide. Booklet, Alexandria
39. Rand WM (1971) Objective criteria for the evaluation of clustering methods. *J Am Stat Assoc* 66(336):846–850
40. Recasens M, Hovy EH (2010) BLANC: implementing the rand index for coreference evaluation. *Nat Lang Eng* 17:485–510
41. Rich E, LuperFoy S (1988) An architecture for anaphora resolution. In: Proceedings of the second conference on applied natural language processing (ANLP-2), Austin, pp 18–24
42. Seddik MK, Farghaly A (2011) Arabic anaphora resolution in Holy Qur'an text. In: Proceedings of ALTIC 2011 conference on Arabic language technology, Alexandria, pp 21–28
43. Sidner CL (1979) Towards a computational theory of definite anaphora comprehension in English discourse. Technical report No. 537. MIT, Artificial Intelligence Laboratory
44. Soon W, Ng H, Lim D (2001) A machine learning approach to coreference resolution of noun phrases. *Comput Linguist* 27(4):521–544
45. Vilain M et al (1995) A model-theoretic coreference scoring scheme. In: Proceedings of the sixth message understanding conference (MUC-6), Columbia, pp 45–52
46. Williams S, Harvey M, Preston K (1996) Rule-based reference resolution for unrestricted text using part-of-speech tagging and noun phrase parsing. In: Proceedings of the international colloquium on discourse anaphora and anaphora resolution (DAARC), Lancaster, pp 441–456
47. Yang X, Zhou G, Su J, Tan CL (2003) Coreference resolution using competition learning approach. In: ACL'03: proceedings of the 41st annual meeting on Association for Computational Linguistics, pp 176–183
48. Yang X, Su J, Tan CL (2008) A twin-candidate model for learning-based anaphora resolution. *Comput Linguist* 34(3):327–356. Iida, R
49. Zitouni I, Sorensen J, Luo X, Florian R (2005) The impact of morphological stemming on Arabic mention detection and coreference resolution. In: Proceedings of the ACL workshop on computational approaches to Semitic languages. 43rd annual meeting of the association of computational linguistics (ACL2005), Ann Arbor, pp 63–70
50. Zitouni I, Luo X, Florian R (2010) A statistical model for Arabic mention detection and chaining. In: Farghaly A (ed) Arabic computational linguistics. CSLI Publications, Center for the Study of Language and Information, Stanford, pp 199–236

## Further Reading

1. Asher N, Lascarides A (2003) Logics of conversation. Cambridge University Press, Cambridge/New York
2. Bengtson E, Roth D (2008) Understanding the value of features for coreference resolution. In: Proceedings of the 2008 conference on empirical methods in natural language processing, Honolulu, pp 294–303
3. Boguraev B, Christopher K (1997) Salience-based content characterisation of documents. In: Proceedings of the ACL'97/EACL'97 workshop on intelligent scalable text summarisation, Madrid, pp 3–9
4. Cai J, Strube M (2010) End-to-end coreference resolution via hypergraph partitioning. In: Proceedings of the 23rd international conference on computational linguistics, Beijing, 23–27 Aug 2010, pp 143–151
5. Chen B, Su J, Tan CL (2010) A twin-candidate based approach for event pronoun resolution using composite Kernel. In: Proceedings of the COLING 2010, Beijing, pp 188–196
6. Diab M, Hacioglu K, Jurafsky D (2004) Automatic tagging of Arabic text: from raw text to base phrase chunks. In: Dumas S, Marcus D, Roukos S (eds) HLT-NAACL 2004: short papers. Association for Computational Linguistics, Boston, pp 140–152
7. Elghamry K, El-Zeiny N, Al-Sabbagh R (2007) Arabic anaphora resolution using the web as corpus. In: Proceedings of the 7th conference of language engineering, the Egyptian society of language engineering, Cairo, pp 294–318
8. Gasperin C (2009) Statistical anaphora resolution in biomedical texts. Technical report, Computer Laboratory, University of Cambridge
9. Ng V (2010) Supervised noun phrase coreference research: the first fifteen years. In: Proceedings of the 48th annual meeting of the association for computational linguistics, Uppsala, pp 1396–1411
10. Noklestad A (2009) A machine learning approach to anaphora resolution including named entity recognition, PP attachment disambiguation, and animacy detection. PhD, Faculty of Humanities, The University of Oslo, Norway, p 298
11. Recasens M, Mart T, Taul M, Marquez L, Sapena E (2010) SemEval-2010 task 1: coreference resolution in multiple languages. In: Proceedings of the NAACL HLT workshop on semantic evaluations: recent achievements and future directions, Los Angeles, pp 70–75
12. Wintner S (2004) Hebrew computational linguistics: past and future. *Artif Intell Rev* 21(2):113–138
13. Zhao Sh, Ng HT (2010) Maximum metric score training for coreference resolution. In: Proceedings of the 23rd international conference on computational linguistics (COLING'10), Stroudsburg, pp 1308–1316

# Chapter 9

## Relation Extraction

Vittorio Castelli and Imed Zitouni

### 9.1 Introduction

By *relation* we denote a connection between two entities (such as persons, organizations, countries, and locations), or between an entity and an event, explicitly supported by the text. For example, the sentence “Ginni Rometty works for IBM” explicitly establishes an employment relation between “Ginni Rometty” and “IBM”. Thus, the definition of relation addressed in this chapter is aligned with that of the NIST Automatic Content Extraction (ACE) evaluations [31]. We will be interested in *closed-set* relations, that is, we will assume that the types of the relations belong to a predefined, finite set. We will investigate statistical approaches to relation extraction, where the task is cast into a classification framework. Our discussion will be prevalently rooted in the supervised learning framework but should be easily extended to semi-supervised approaches.

The term “relation extraction” has been broadly used in the literature, and covers many NLP problems not addressed in this chapter. This chapter doesn’t consider the detection of implicit relations. For example, the sentence “Ginni Rometty works for IBM and Paul Horn belongs to the same company as Ginni Rometty” does not explicitly establish that “Paul Horn” is employed by “IBM”; rather, this conclusion is reached by following an inference chain connecting multiple explicit relations. Also, this chapter doesn’t address the automatic discovery of ontology-style relations (e.g., hyponymy, meronymy, etc.), nor the detection of temporal relations between events (A comes before B). It doesn’t cover *open-set relation*, a

---

V. Castelli (✉)

IBM T.J. Watson Research Center, Yorktown Heights, NY, USA

e-mail: [vittorio@us.ibm.com](mailto:vittorio@us.ibm.com)

I. Zitouni

Microsoft, Redmond, WA, USA

e-mail: [izitouni@microsoft.com](mailto:izitouni@microsoft.com)

data-mining problem consisting of identifying common patterns connecting pairs of entity mentions and clustering them into groups that might have semantic meaning. We only discuss statistical approaches to relation extraction, leaving out topics such as rule-based systems.

The rest of the chapter is organized as follows: Sect. 9.2 describes in more detail the concept of relation and the problem of relation extraction; Sect. 9.3 provides an overview of methods for relation extraction and discusses feature-based methods, kernel methods, and semi-supervised methods; Sect. 9.4 addresses language-specific issues that arise when extracting relations from Semitic languages; Sect. 9.5 is a brief overview of available data for training relation extraction systems; Sect. 9.6 describes results on Arabic data.

## 9.2 Relations

A sentence often contains *mentions* of physical or abstract *entities*, such as persons, groups of people, locations, countries, numerals, substances, vehicles, weapons, etc. These mentions can be proper nouns (e.g., Barak Obama), nominals (e.g., author, editor, president), or pronouns. *Mention detection* is the process of automatically identifying the spans of text corresponding to mentions within a textual document. Often, multiple mentions within the same document refer to the same entity; the process of partitioning mentions into mutually exclusive sets, each of which corresponds to a different entity, is called *within-document coreference resolution*, or, more simply, coreference resolution, and each set of mentions is sometimes called a *document-level entity*. Consider as an example the sentence “Bob bought a new car on Friday and drove it to California the following day”. The sentence contains a person mention (“Bob”), two vehicle mentions (“car” and “it”), and two time mentions (“Friday” and “day”). The two vehicle mentions refer to the same entity, that is, the car that Bob bought on Friday. The reader may refer to the previous chapter on anaphora resolution for more details about a similar research problem. Different documents could have mentions referring to the same entity. Identifying document-level entities in different documents that correspond to the same real-world entity is called *cross-document coreference resolution*. Mention detection (which can be addressed as a named-entity recognition problem), coreference resolution and cross-document coreference resolution are complex problems, some of which are addressed in the previous two chapters of this book. In this chapter, we assume that they are solved, and that their products are available when relation extraction is performed.

Consider again the sentence “Bob bought a new car on Friday and drove it to California the following day”. This sentence establishes that Bob became the owner of a new car, because he purchased it. The sentence describes an *ownership relation* between Bob and the car. The textual description of this relation is a *relation mention*. The ownership relation is stated as factual; a similar relation is described in hypothetical terms by the following sentence “Bob would have bought a new car on Friday but the dealer did not have it ready.” The relation is established in the past,

since the sentence does not provide information as to whether Bob is still the owner of the car – he might have sold it in California. This example suggests that relation extraction is a multi-faceted problem: it consists of determining whether there is a relation between a pair of mentions, of inferring the type of the relation, and of computing attributes of the detected relations, such as the *roles* of the mentions in the relation, whether the relation is *factual* or *hypothetical*, the *time* aspect of the relation, etc. Thus, a relation mention can be represented as a tuple  $(m_1, m_2, a_1, \dots, a_m)$  where  $m_1$  and  $m_2$  are entity mentions, and  $a_1, \dots, a_m$  are attributes. A relation can then be identified with the collection of its relation mentions.

Binary relations can be extended to  $n$ -ary relations that connect  $n$  entities: for example the sentence “Bob, Mary, and their daughter Jane went to the movies” establishes a family relation between Bob, Mary, and Jane. The problem of relation extraction can be tied to that of *event detection*. A simple representation of an event consists of determining a keyword in the sentence, which we call the event *anchor*, and extracting relations between the anchor and other mentions. This simple representation captures who did what to whom, with what, where, and when. An event anchor in the first example is the verb “bought”, which takes part in a set of relations: Bob is the agent of bought, the new car is the patient of bought, and Friday is the time of the action. A second event can be extracted, centered on the anchor “drove”: Bob is the agent of drove, “it” (the car) is the patient, California is the explicitly mentioned location of the action, and the “following day” is the time of the action. In this chapter we will restrict the attention to binary relations, in the interest of simplicity.

The vast majority of the literature on relation extraction addresses the case of relations between entity mentions that occur in the same sentence. While there are a few investigations of cross-sentential relations [18], we will assume throughout the chapter that the entity mentions occur within the same sentence.

## 9.3 Approaches to Relation Extraction

Relation extraction approaches can be broadly categorized into three main classes: feature-based classifiers, which approach the problem from a statistical classification angle, kernel-based methods, which efficiently compute the similarity between appropriate representations of the training data and the data being analyzed and make a decision based on the result, and semi-supervised methods that augment a training set consisting of labeled samples with unlabeled data.

### 9.3.1 Feature-Based Classifiers

#### Methods

For each pair of mentions to be analyzed, a feature-based classifier [23] extracts features that are fed as input to one or more classifiers devoted to the computation

of the relation attributes. For many relation sets, such as those used in the ACE evaluation [31], detecting the existence of a relation is substantially more difficult than computing the attributes of detected relations: for example, misses and false alarms dominate over relation type errors. The cause is the absence of relations between most pairs of mentions even within the same sentence. The training set is heavily biased by examples that describe the absence of relations, and a classifier trained with this kind of data tends to produce highly precise results but at the same time would miss a large number of relations. Selecting a different point on the ROC curve improves recall at the expense of precision. Due to the difficulty of relation detection, some authors have concentrated their investigation on this problem [16] without addressing the computation of relation attributes.

There are two possible approaches to classifying relations. The first is to build a single classifier that computes all the relation attributes at the same time; in other words, the classifier prediction lies in the Cartesian product of the attribute spaces augmented by a “No Relation” label. The second approach consists of having a classifier devoted to detecting the existence of relations and one or more classifiers that produce the attributes of the detected relations.

As an example of using a single classifier, Kambhatla [24] addressed the probem of detecting relations in the ACE relation taxonomy. The ACE relation taxonomy consists of a handful of different relation types and subtypes. By combining type, subtype, and role of the two entity mentions in the relation, one produces 46 different classes, to which a “NONE” class label is added for a total of 47 class labels. With this approach, the training set is highly skewed towards the “NONE” class. Kambhatla proposed a solution consisting of creating numerous training sets by sampling-with-replacement from the original training set, training a separate classifier with each generated training set, and combining the classifiers using an “At-Least-N” strategy: if at least N classifiers agree on the same label  $\ell$  other than “NONE” and no other label receive more votes, the ensemble outputs the label  $\ell$ , otherwise the label “NONE” is produced.

The alternative common approach consists of using a binary classifier to detect whether a relation exists between a pair of mentions, followed by other classifiers that compute different attributes for the relations detected by the detector. The attribute classifiers can be used independently, or they can be cascaded, that is, the prediction of each classifier is used to compute features for the following classifiers. There are several advantages to this approach. First, detection is a binary problem and all the training data is used to decide between two labels, rather than between many labels. Second the designer has the flexibility to select appropriate features for each classifier: features useful for detection need not be advantageous for attribute computation and vice versa. The architecture of the system can be flexibly designed: for example, classifiers can be cascaded so that each classifier can rely on features computed from the predictions of previous stages. Finally, rule-based systems can be selectively applied, especially in a cascaded architecture. A case in point is the role of the entity mentions in the relation: in certain relation types, they play specific roles (e.g., the sentence “Bob is the manager of John” expresses the relation (Bob, John, ManagerOf), which is different from (John, Bob, ManagerOf)), while in others

they are exchangeable (e.g., the sentence “Bob and Jane got married” expresses the relation (Bob, Jane, spouseOf), which is identical to the relation (Jane, Bob, spouseOf)). A simple rule can be constructed that invokes the mention-role classifier only when the relation-type classifier produces a label for which the roles of the entity mentions are not symmetric. Similarly, the types of the entities involved in a relation constrain the possible set of relations. For example, a person can be employed by a company, can own a company, but cannot be married to a company (although the authors concede that on occasion it feels that way). Such constraints can, and should, be incorporated in the relation detection system.

## Features

Different authors have proposed and used a wide array of features for classification-based relation extractors. A study that analyzes several features classes in the context of relation extraction can be found in the paper by Jiang and Zhai [21].

**Structural features.** Consider the sentence: “While Julius Caesar was in Britain, his daughter Julia, who was married to Pompey, died in childbirth in Rome”. This sentence contains numerous relations, for example: (1) Julius Caesar is located in Britain, (2) he is the father of Julia, (3) Julia is the spouse of Pompey, and (4) Julia is located at Rome. Note, further, that the second relation mention is explicitly supported by the words *his daughter*, where the pronoun *his* is coreferenced to *Julius Caesar* and the nominal *daughter* is coreferenced to *Julia*. Similary, the third relation mention is between *who*, coreferenced to *Julia*, and *Pompey*. We note immediately that entity mentions that are close to each other in the text are substantially more likely to be part of an explicit relation than mentions that are far from each other – we remark, incidentally, that the fact that Julius Caesar is the father-in-law of Pompey is an implicit relation that follows from inference. Also note that the mentions *Julia* and *Rome* are separated by several words, but the shortest path between them in the parse tree is actually rather short. These considerations lead us to consider a class of features that capture the distance between entity mentions, such as the number of intervening words, the length of the shortest path between the mentions in the parse tree, or the length of the shortest path between the mentions in the dependency tree. Another class of structural features captures whether the entity mentions being analyzed already play a role in a relation. These features must be *causal*, in the sense that they must only fire for relations that could be detected by the extraction algorithm before the candidate relation between the current entity mentions; in other terms, if the relation detection algorithm analyzes all entity mention pairs in a given order, these features should fire only if an entity mention  $m_1$  in the current pair  $(m_1, m_2)$  appears in a relation with another entity mention  $m_p$  and the pair  $(m_1, m_p)$  occurs before  $(m_1, m_2)$  in the order induced by the relation detection algorithm.

**Lexical features.** Different relations can be expressed in identical structural, grammatical, and syntactic fashion. “His mother” and “his sister” express two very

different relations (*parent-child* and *sibling of*), which can only be distinguished by looking at the actual words. Lexical features are therefore often used in relation extraction systems. Examples of lexical features are: (1) the spelling of the mentions, possibly stemmed; (2) the verb of the verb phrase containing the two mentions, possibly stemmed; (3) verbs to the left and to the right of the mention pair; (4) *bag-of-words* features that collect all the words between the two mentions or in the same syntactic constituent that contain the two words; etc. Unlike structural features, lexical features will increase dramatically the dimensionality of the feature space. As a consequence, for languages with rich morphology it is advisable to use a morphological analyzer or a stemmer to remove affixes.

**Entity features.** Consider the two sentences: “I visited France” and “France holds a seat in the U.N. Security Council”. In the first sentence, the geo-political entity *France* plays the role of a geographic area; in the second, it plays the role of a sovereign state. A person cannot be *located at* a sovereign state, which is a political organization with a centralized government that has independent and supreme authority over a geographic area. Similarly, a geographic area cannot be a *member of* an organization. The entity type and its role in a sentence dictate the type of relations in which it can take part. Additionally, knowing whether the entity mentions are represented by proper names, nominals, or pronouns appears to be useful in relation detection.

**Syntactic features.** A first class of syntactic features are derived from parse trees. These can be further divided into two categories: label-based and path-based. *Label-based features* capture the non-terminal labels associated with the words of the entity mentions, including part-of-speech tags. *Path-based features* represent various encoding of the labels of non-terminal nodes encountered in paths along subtrees that dominate the entity mentions. For example, authors have considered the smallest subtree that cover the entity mentions and extracted features from the shortest path between the head words of the mentions. Examples of extracted features include: (1) the constituent label of the root of the smallest subtree covering the pair; (2) the list of the labels of the children of the root; (3) the list of all or of selected constituent labels along the shortest path between the two mentions; etc. Useful syntactic features include those that fire when the mentions are in the same phrase, noun phrase, sentence, etc. Binary features can be constructed from syntactic patterns, for example *mention<sub>1</sub>-PP containing mention<sub>2</sub>*, which fires if the leftmost mention is in apposition to a propositional phrase that contains the rightmost mention.

A second class of syntactic features are derived from dependency trees. These include features that concatenate the constituent labels encountered along the full or reduced path along the dependency tree connecting the mentions, as well as features that fire when specific dependency patterns are encountered.

**Semantic features.** If the parse-tree constituents are decorated with semantic roles [14], it is possible to construct features that capture semantic connections between the mentions. Features of this type fire when the two mentions being analyzed belong to constituents that have semantic roles associated with the same

verb. Examples include: (1) the semantic role labels (SRLs) of the constituents that contain the two mentions; (2) the (possibly stemmed) verb or verbs; and (3) indicator functions that fire if the entity mentions span the head words of the constituents with semantic role labels.

## Classifiers

As in many other NLP tasks, there are two basic approaches to relation extraction as classification; the first consists of considering each candidate relation mention independently of all the others (i.i.d. classification), and the second is to perform joint extraction, for example, using a sequential classification approach. While conceptually any classifier can be used to detect relations, the dimensionality and sparsity of the feature vectors requires methods that are somewhat impervious to the curse-of-dimensionality. Thus, authors typically select log-linear models, such as Maximum-Entropy classifiers [2] (or Maximum Entropy Markov Models [28] for sequential classification) or Conditional Random Fields [26]. These models estimate probabilities using the generic formula

$$P(Y_i = y_i \mid \mathbf{X} = \mathbf{x}) = K \exp \left( \sum_j \lambda_j^{(y_i)} f_j^{(y_i)}(y_0^{i-1}, y_i, \mathbf{x}, i) \right)$$

Other classifiers, such as large-margin methods, have been used both for feature-based methods and for kernel-based methods, and are briefly discussed in the next section.

### 9.3.2 Kernel-Based Methods

Linguistic patterns are central to natural languages. Even in grammatically permissive languages such as English, people express themselves using patterns of speech, and deviations from established patterns sound odd even when they are perfectly comprehensible and correct according to the rules of grammar.

As an illustrative example consider the following two passages: “Dr. John E. Kelly III, Senior Vice President of IBM Research, spoke . . .” and “Gary Locke, the U.S. Ambassador to China, met with . . .”. Both exemplify common patterns that concisely describe how a person is related to an organization or country by specifying the role or the title of the person. These patterns are very well established and have limited variability: for example, the passage “Dr. John E. Kelly III, who is the Senior Vice President of IBM Research, spoke . . .” is grammatically correct, expresses the same relations as the previous version, but is substantially less likely to occur in written text or in a spoken conversation. It is therefore natural to address the problem of relation extraction by learning structural patterns and capturing their

limited variability. Luckily, this goal can be accomplished efficiently by means of a class of “similarity” functions called *kernels*, applied to shallow parse trees, regular parse trees, and dependency trees.

Kernels are strictly linked to a historical classifier, the perceptron, and its derivatives. The perceptron is a classical learning algorithm invented by Rosenblatt [33] that has inspired a large number of classifiers, including Support Vector Machines [10] and other large margin classifiers, such as the Voted Perceptron [13]. In its simplest form, the perceptron is a linear binary classification algorithm: a sample to be classified is represented as a vector of features  $\mathbf{x}$ , and the perceptron uses a weight vector  $\mathbf{w}$  to predict the label of  $\mathbf{x}$  as

$$\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x}),$$

where  $\cdot$  denotes the inner product and *sign* is the signum function. Thus, the weight vector defines a separating hyperplane in the feature space, and samples are classified according to half of the feature space they belong to. Incidentally, note that an inner product defines a natural distance function:

$$d(\mathbf{x}, \mathbf{w}) = \sqrt{\mathbf{x} \cdot \mathbf{x} + \mathbf{w} \cdot \mathbf{w} - 2\mathbf{x} \cdot \mathbf{w}},$$

and therefore it can be thought of as a similarity function.

To account for the interaction between features, it is customary to project the feature vector in a higher-dimensional space: for example, if  $\mathbf{x} = [x_1, \dots, x_n]$ , using the vector  $\mathbf{x}^{(2)} = [x_1, \dots, x_n, x_1^2, x_1 x_2, \dots, x_{n-1} x_n, x_n^2]$  in the perceptron algorithm accounts for all the two-way interactions between features. It is evident that accounting for multi-way interactions substantially increases the length of the feature vector, and consequently the computational cost of training the model and classifying the samples.

A solution consists of computing the inner product by means of a Mercer Kernel  $K(\mathbf{w}, \mathbf{x})$ , which is a real-valued, symmetric, continuous, positive-semidefinite function. Mercer’s theorem shows that, under a few mild conditions, there is an orthonormal basis of the space of square-integrable functions defined over the range of the vectors  $\mathbf{w}$  and  $\mathbf{x}$ , such that the eigenvalues  $\{\lambda_i\}$  corresponding to the eigenfunctions  $\{e_i\}$  are non-negative, and the kernel has the representation

$$K(\mathbf{w}, \mathbf{x}) = \sum_{i=1}^{\infty} \lambda_i e_i(\mathbf{w}) e_i(\mathbf{x}),$$

where the sum on the right-hand side converges absolutely and uniformly. Thus, computing the value of the right-hand kernel is tantamount to projecting  $\mathbf{w}$  and  $\mathbf{x}$  in a high-dimensional (possibly infinite-dimensional) space and computing a weighted inner product between the projections. It is possible to construct kernels that are substantially less computationally intensive to evaluate than the computation of the inner product in the corresponding high-dimensional space.

Even more appealing from the viewpoint of relation extraction is the existence of kernels on discrete structures such as strings, graphs, and trees: these can be used to automatically capture linguistic patterns. Haussler [17] pioneered these kernels and called them *convolutional kernels*. If  $\mathbf{x}$  is a structure that can be decomposed into parts  $\vec{x} = x_1, \dots, x_D$  and  $\mathbf{z}$  can be decomposed into parts  $\vec{z} = z_1, \dots, z_D$ , and for each  $d \in \{1, \dots, D\}$  there is a kernel  $K_d$  that can take  $x_d$  and  $z_d$  as arguments, then the  $R$ -convolution or  $K_1, \dots, K_D$  is (the zero-extension of)

$$K(\mathbf{x}, \mathbf{z}) = \sum_{\vec{x} \in R^{-1}(\mathbf{x}), \vec{z} \in R^{-1}(\mathbf{z})} \prod_{d=1}^D K_d(x_d, z_d),$$

where  $R^{-1}(\mathbf{x})$  is the set of possible decompositions of  $\mathbf{x}$ .

Collins and Duffy [9] extended the convolution kernels to parse trees. Their kernel implicitly finds all subtrees in common between two parse trees, assigns to each subtree a weight that decays exponentially with its size, and sums the weights. They propose a dual-space algorithm that efficiently computes the kernel without having to explicitly enumerate and check all the subtrees of the parse trees.

Convolutional kernels and related kernels have been applied to relation extractions by several authors. Zelenko et al. [36] propose the use of kernels closely related to convolutional kernels to extract relations from shallow parse trees [1]. Shallow parse trees identify the key elements of a sentence, rather than its entire structure. Relation extraction is cast as a node-labeling problem: the nodes in the shallow parse tree are labeled with the role they play in defining the relation, and nodes that do not participate in the relation are not labeled. The author define a matching function, which decides whether a pair of nodes from two different parses are comparable, and a similarity function, which computes the similarity between the attributes of matchable nodes. By appropriately crafting and combining these two functions, the authors construct a kernel that efficiently recursively computes the similarity between subtrees. The results reported on two relation types (person-affiliation and organization-location) are encouraging, and this approach is potentially useful for resource-poor languages, since a chunker can be trained with a smaller training set than a full-fledged parser.

Culotta and Sorensen [11] define a convolutional kernel that extends that of Zelenko et al., and that operates over dependency trees. Dependency trees are structures that capture the grammatical relations between words. To extract a relation between two entity mentions, Culotta and Sorensen find the smallest common subtree of the dependency tree that include both mentions, and represent each node of the subtree as a vector of features. A subset of these features is used to compute the compatibility between tree nodes and another subset, possibly overlapping with the first, is used to compute the similarity between compatible nodes. This approach, applied to a set of five high-level ACE relations yields a detection F-measure of 63.2 % and a classification F-measure of 45.8 %.

Bunescu and Mooney [4] further refine Culotta and Sorensen's method by restricting the attention to the shortest path between two entities in the undirected

version of the dependency graph. The authors define lexical features, namely, the words on the dependency path, and additional features constructed by substituting the words in the lexical feature with their hypernyms, their part-of-speech tags, etc. This approach yields a 2-point increase in F-measure.

Zhang et al. [37] extend the previous works by relaxing some of the technical requirements of Culotta and Sorensen, and of Bunescu and Mooney. In particular, they relax the requirement that the subtrees being compared have the same number of nodes and that they lie at the same depth from the roots. The authors explore how the selection of subtrees affects the performance of the relation extraction algorithm, by defining seven different approaches and comparing their performance. The methods described in the paper rely on syntactic features, and show noticeable gains in F-measure.

Khayyamian et al. [25] apply kernel methods to syntactic parse trees, define different weighting functions based on a decaying parameter, and compare their results on individual relation types to those obtainable with the kernel of Collins and Duffy [9].

### **9.3.3 *Semi-supervised and Adaptive Learning***

The methods described in the previous sections rely on the assumption that a labeled corpus is available for training classifiers. This corpus must contain sentences where the mentions of entities and events as well as the relations between these mentions are annotated. The annotations must contain the information needed to extract features for the learning algorithm, including all the desired relation attributes. There are very few annotated corpora available even for resource-rich languages such as English, and annotating a large corpus is a delicate and expensive task.

To address this issue, authors have proposed the use of semi-supervised and of adaptive methods that require only a limited amount of labeled data to train a classifier. Traditional semi-supervised methods rely on a small number of labeled examples for the specific problem at hand and on a large unlabeled corpus to construct a classifier. Adaptive methods use labeled data created for related domains and unlabeled domain-specific data to construct relation extraction systems. Since both classes of approaches use both labeled and unlabeled data, we group them together and refer to them as semi-supervised methods.

In this chapter we review several semi-supervised approaches to relation extraction, provide a high-level description of the methods, and describe some implementations.

**Bootstrap Learning.** Bootstrap learning consists of using a very small number of labeled samples or of seed patterns to iteratively extract new labeled samples or patterns from a corpus. This method has been used for relation extraction by Glass and Barker [15]. In this paper, the authors construct extractors for individual relation types using a set of examples of the desired relation and a corpus. For each example of the desired relation, they search the corpus for sentences containing

both entities found in the example; they add the sentence as a positive example to a new training set. Once all initial labeled examples have been exhausted, a new classifier is constructed with the new training set. This classifier is used to discover new instances of the desired relation type, which are treated as the initial set of labeled examples to construct a new training set for the next iteration of the process. The authors provide an extensive analysis of experimental results and conclude that the method can be beneficial for specific tasks under certain conditions. A similar approach is used in Hoffman et al. [19], and Wu and Weld [35], where the labeled examples are Wikipedia infobox entries and bootstrapping consists of labeling as positive examples the sentences of the Wikipedia article containing the infobox where the two entities taking part in the relation co-occur.

**Multiple Instance Learning.** Multiple instance learning (MIL) [12] addresses the case in which the training data is grouped into “bags”, and a label is assigned to each bag. In the case of binary classification, bags are positive or negative, positive bags are guaranteed to contain at least one positive instance (but can also contain negative instances) and negative bags are guaranteed to contain only negative instances. Bunescu and Mooney [6] extend a subsequence kernel approach [5] to the multiple instance learning framework, using a support-vector machine as the learning algorithm. The authors recast the problem into a classical classification setting, where training instances belonging to positive bags are treated as positive examples and training instances belonging to negative bags are treated as negative examples. Since not all examples in a positive bag are positive, the authors modify the SVM optimization problem by assigning a smaller misclassification penalty to examples from a positive bag than to those from a negative bag.

**Distant Supervision.** Distant supervision is a subclass of semi-supervised learning where the unlabeled data is “weakly labeled”, that is, it is labeled by a labeling function often derived from a knowledge base. To create the training set, a knowledge base is used both to obtain a small set of labeled examples and to construct the labeling function. The labeling function is often constructed using heuristics that guides its applicability. The labeling functions is then applied to a corpus to obtain automatically annotated training examples that are then used to learn an extraction system.

Mintz et al. [29] use Freebase [3] as a structured knowledge base. Freebase contains more than 100 million instances of relations between 9 million entities, and these relations belong to more than 7,000 different types. The gist of the approach of Mintz et al., is to project the relations between entities captured by Freebase onto a corpus and to use the projected relations as training examples. In particular, every phrase in a corpus that contains mentions of two entities linked by a specific relation in Freebase are considered as positive examples for that relation. The training corpus used in the paper is a collection of 1.2 million Wikipedia articles, and the approach yields 1.8 million weakly labeled examples covering 102 types of relations.

Nguyen and Moschitti [30] take a similar approach but use a corpus that is coupled with the knowledge base. The authors use the Wikipedia infoboxes as a relation knowledge base and project the relations from an infobox to the

corresponding Wikipedia page. The advantage of the method is that if two entities are linked by a specific relation in the knowledge base, they are not necessarily linked by the same relation in an unrelated text passage. Thus, projecting a knowledge base indiscriminately onto a large collection will produce a large number of false alarms as well as of relation type errors, because pairs of entities can be arguments of different types of relations. On the contrary, if a relation is expressed in a Wikipedia infobox, the text of the corresponding page typically contains a passage expressing that relation. Thus, Nguyen and Moschitti's method produces fewer training examples than that of Minz et al., but the resulting training set is probably of higher quality. The authors also combine distant supervision with regular (direct) supervision and test their approach on the ACE 2004 dataset. First, they map the relation types obtained with distant supervision onto the seven main classes of ACE relations; then, they train two classifiers, one with the ACE dataset and another with the combination of the ACE dataset and the remapped training set produced with distant supervision; finally, they combine the classifier posterior probabilities via a linear combination. The cross-validation results on the English ACE data yield an F-measure of 67.6 %.

**Transfer Learning and Multi-task Learning.** Transfer Learning [7, 32, 34] is a class of techniques that uses data or classifiers trained for a specific task to improve learning in a related task for which limited training data is available. Transfer learning could be used, for example, to leverage relation extraction training data and classifiers in resource-rich languages to help relation extraction in resource-poor languages; to help in learning a new relation taxonomy given classifiers and data for an existing taxonomy. Multi-task learning [7, 34] is a closely related paradigm where classifiers for multiple related problems are learned jointly.

Jiang [20] applies transfer learning to the problem of constructing a classifier for new relation types using information for other relation types. The author proposes a general multi-task learning framework where classifiers for related tasks share one or more common model components and are trained together. The problem of learning classifiers for different relation types is then cast into this framework by noting that often different types of relations share very similar syntactic structures. The author notes that this approach improves the recall of the relation extractor system and shows empirically that, on the ACE 2004 dataset, the method is substantially better than two baseline approaches. In particular, when using transfer learning to learn a new relation type using a previously learned relation, the author shows a jump in F1 measure from 0.1532 to 0.4132 even when using as few as 10 labeled examples of the new relation type.

Chieu et al. [8], apply transfer learning to cross-domain relation extraction by investigating how similarity measures from separate domains correlate with each other and use the results to regularize the inference of the model parameters in a multi-task learning environment. The authors also address multi-task learning applied to different types of relations; they also observe how different relations share similar syntactic patterns, and they propose a method in which each relation type is learned with a separate classifier and the problems of learning different relation types are considered as related tasks.

## 9.4 Language-Specific Issues

The authors of the introductory chapter of this book give a detailed characterization of basic linguistic facts about Semitic languages, covering orthography, morphology, and syntax. The chapter also presents a contrastive analysis of some of these phenomena across various Semitic languages. In this section, however, we discuss major challenges affecting detecting relations in Semitic language documents, which we can summarize as follows:

1. **Lack of Diacritics:** Semitic languages have two types of vowels. The long vowels are written as letters in the text and short vowels are presented as diacritics. Diacritics are often omitted from text, especially in news-wire texts, because it is assumed that the reader can still detect the meaning and the syntax while reading the text. NLP systems on the other hand will have to deal with this extra layer of ambiguity because when the short vowels are, completely or partially, omitted new homographs are created. As an example, we take the Arabic diacritic-less word كتب. When adding diacritics, this word represents *books* in the case of كُتُب (kutubuN) and the verb *to write* in the case of كَتَبْ (kataba).
2. **Morphological Challenges:** Semitic languages have a complex morphology and hence its handling is very important for the relation extraction task. In Semitic languages such as Amharic, Arabic, Hebrew, Maltese and Syriac, words are derived from a set of roots. Stems are constructed using template and vowel patterns that may involve the insertion of vowels, adding prefixes and suffixes, or doubling constants. Stems may accept the attachment of prefixes and suffixes that include pronouns, determiners, plural and gender markers. Enclitic pronouns are attached to the word they modify: pronouns are often realized as prefixes or suffixes attached to a base form (stem). Consequently, Semitic languages exhibit a high morpheme-per-word ratio that results in a sparseness of data. Applying templates often involves introducing infixes or deleting or replacing letters from the root. Also, prefixes include coordinating conjunctions, determiners, and prepositions, and suffixes include attached pronouns and gender and number markers. This leads to cases where the word may have multiple valid analyses, only one of which is typically correct in context. Taking the Arabic word ولید (wlyd) as an example, it could be the proper name “Waleed” or it may mean “and to hand” يد + ل + و (w+l+yd). Further challenges are related to the fact that in Arabic as an example, some letters are often used in place of others due to varying orthographic conventions, common spelling mistakes, and morphological transformations.

These linguistic phenomena make the task of building any NLP system, including relation extraction, harder than building its counterpart dealing with languages such as English. Hence, one important step when building a relation extraction model for Semitic languages is to define the unit of analysis, also denoted as a token,

when processing the input text. Because we want a system generic enough to detect enclitic pronouns as mentions, it is essential to define a unit of analysis which splits words into sub-word tokens (segments) or characters. On one side of the spectrum, if we consider a character as the subject of analysis (i.e. split all words into individual characters), the mention detection system will make a decision for every character in the document to know whether it is the beginning of a mention, inside a mention, or outside a mention. This idea was successfully applied for languages like Chinese [22], but for a language having rich morphology, this will be suboptimal, as contexts would have to be quite long to capture interesting phenomena needed for extracting relations. On the other side of the spectrum, using a white-space delimited word as the analysis unit leads to data sparseness and does not allow the detection of any sub-word units (such as inflected pronouns).

Therefore, we chose to use *segments* as the unit of analysis when processing Semitic languages. We start with the segmentation of the document into a sequence of segments, which then become the subject of analysis (tokens). The segmentation process consists of separating the normal white-space delimited words into (hypothesized) prefixes, stems, and suffixes. The mention detection system proceeds then on each token to make a decision if it is the beginning of a mention, inside a mention or outside a mention (cf. name-entity recognition chapter). The relation extraction system will then process every pair of mentions to examine whether a relation exists between them.

The resulting granularity of breaking words into segments allows a prefix or a suffix to receive a label different from that of the stem (for instance, in the case of pronominal mentions). In addition, stems carry important information that can be used to boost the performance of the system as demonstrated in [27].

The following is an example of how we process a text in Arabic: for the word مکانہم (their location), the segmentation system splits it into two tokens: مكان (location) and هم (their). The system then detects that the token مكان (location) is a mention that refers to the entity location, whereas the token هم (their) is a mention, but one that might refer to a group of persons. In addition, the prepositions – i.e., ب (by) and ل (to) in عکسہم (by their location) and لکنہم (for their location) respectively – should not to be considered as part of a mention. Then, the relation extraction model will be able to detect a relation of type *belong-to* between the two mentions مكان (location) and هم (their).

## 9.5 Data

There is limited data available for training relation extraction systems in Semitic languages. One important resource is the publicly available relation dataset for Arabic prepared by the Linguistic Data Consortium (LDC) for the NIST Automatic Content Extraction (ACE) evaluation. This dataset is divided into a training set and an evaluation set. The training set contains 511 annotated Arabic documents

**Table 9.1** ACE relation types and subtypes for the ACE 2004 evaluation

Type	Subtype	Type	Subtype
ART <i>(agent-artifact)</i>	User-or-owner	PHYSICAL	Located
	Inventor/manufacturer		Near
	Other		Part-whole
EMP-ORG	Employ-executive	PER-SOC	Business
	Employ-staff	<i>(personal/ social)</i>	Family
	Employ-undetermined		Other
	Member-of-group		
	Partner		
	Subsidiary		
	Other		
GPE-AFF <i>(GPE affiliation)</i>	Citizen-or-resident	OTHER-AFF	Ethnic
	Based-in	<i>(PER-ORG affiliation)</i>	Ideology
	Other		Other
DISCOURSE	-None-		

that contain 4,126 relation mentions. The test set is composed of 178 documents containing 1,894 relation mentions.

Relations in the ACE Arabic dataset have both a *Type* and a *Subtype* attribute. There are six relation types: *ART*, capturing the association between a person and an artifact; *ORG-AFF*, the association between a person and an organization; *PART-WHOLE*, the relation between an entity and its components; *PER-SOC*, the social relation between two persons; *PHYS*, the spatial relation between two entities; *GEN-AFF*, affiliations not captured by any of the previous relations; and *METONYMY*.

With the exception of *METONYMY*, the relation *Type* is further refined by a *Subtype* attribute: *ART* is refined into User, Owner, Inventor or Manufacturer; *GEN-AFF* is split into two subtypes: Citizen-Resident-Religion-Ethnicity, and Organization-location; *ORG-AFF* has several subtypes: Employment, Founder, Ownership, Student-Alumn, Sports-Affiliation, Investor-Shareholder, Membership; the subtypes for *PART-WHOLE* are Artifact, Geographical, and Subsidiary; for *PER-SOC* we have Business, Family, and Lasting-Personal; and finally, the *PHYS* relation has the Located or the Near subtype. Note, incidentally, that the subtype of a relation is in many cases unambiguously derived from the relation type and the types of the entities taking part in the relation: for example, if two organizations are connected by a *PART-WHOLE* relation, the only possible subtype is Subsidiary. The full set of relation types and subtypes is summarized in Table 9.1.

Some of the ACE relations are symmetric in nature: if Bob is related to Mary by a *PER-SOC* relation having subtype Family, then Mary is related to Bob by the same relation. In other cases, the entities play specific roles in the relation, and this fact is captured by the *Role* attribute of the relation; specifically if the relation can be expressed as A Verb Relation B, then A is considered “Arg-1” of the relation and B is considered “Arg-2”.

The Modality attribute of a relation that can occur unconditionally in the real world is ASSERTED, otherwise it has value equal to OTHER. A relation can occur at a specific point in time or during a specific time interval; this is captured by the Tense attribute, that can have values equal to Past, Present, Future, and Unspecified, the latter capturing all the cases that do not squarely fall in the first three.

The ACE data also contains annotations that link together multiple mentions of the same relation, thus providing training data for both the Relation Mention Detection (finding mentions of relations in the text) and the Relation Detection and Recognition tasks (grouping together mentions of the same relation).

## 9.6 Results

Automatic relation extraction in Semic languages is an open field of research. It is illustrative to describe in some detail Nanda Kambhatla’s system [24], one of the few designed for this task. The paper describes in detail the implementation and experimental results for the ACE 2004 relation detection and recognition task, which is based on the set of relations illustrated in Table 9.1. The approach relies on a cascaded architecture consisting of a relation detection stage followed by classifiers that assign attributes to the detected relations. The relation detection stage analyzes all pairs of mentions within a sentence, and discards those for which no relation is defined in the ACE taxonomy. The remaining mention pairs are analyzed by an ensemble of maximum-entropy classifiers, trained with bootstrap replicates of the training set. These bootstrap replicates are obtained by independent sampling with replacement from the training set, and have the same number of training examples as the training set. The classifiers in the ensemble are trained to detect the existence of a relation between a pair of mentions, and produce a binary label, that is, a vote, as well as a posterior probability of detection. The votes of the classifiers in the ensemble are combined using the “At-Least-N” approach: if at least  $N$  of the votes are positive, the ensemble detects the existence of a relation.

The ACE training set consists of 511 Arabic documents containing 4,126 relation mentions, while the test set consists of 178 documents with 1,894 relation mentions. The distribution of relation types and subtypes is highly skewed: the number of relation mentions of least represented relation subtype is two orders of magnitude smaller than that of the most represented relation subtype. From this raw data, all entity mentions are extracted, and for each entity mention pair in the same sentence a variety of features are computed. These include lexical, semantic, and syntactic features (described earlier in this chapter).

The authors compare various ensemble sizes and various strategies, including majority vote, averaging the posteriors and detecting relations when the average is at least  $1/2$ , and different values of  $N$  in the “At-Least-N” approach. They conclude that F-measure is maximized with less than 10 bags, and that the “At-Least-2” strategy outperforms other choices of  $N$ , majority vote, and summing, as

**Table 9.2** Results of best classifiers on ACE 2004 Arabic Relations. For each approach, the number of bags yielding the highest F-measure is selected, and the rounded results reported. The highest values in each column are highlighted in boldface

Approach	Best Nr bags	Precision	Recall	F-measure
Single classifier	–	0.39	0.21	0.27
Majority	7	<b>0.43</b>	0.22	0.29
Summing	24	<b>0.43</b>	0.22	0.29
At least 1	9	0.32	<b>0.28</b>	0.295
At least 5	24	0.36	0.25	0.295
<b>At least 2</b>	9	0.36	<b>0.28</b>	<b>0.305</b>

summarized in Table 9.2. The authors also remark that the “At-Least-2” strategy significantly outperforms the best classifier of the ensemble (the one with highest performance on the test data) in terms of the ACE value (the offical metric of the evaluation).

## 9.7 Summary

We presented in this chapter the primary ways in which relations are extracted from text. We described how relations form the higher-level components in an information extraction toolkit, in that they rely on features derived from lower-level components, such as part-of-speech taggers, parsers, a semantic-role labeling system, a mention detection component and a coreference resolution system. Relation extraction systems typically rely on lexical features, which provide a rich but potentially sparse form of usefully discriminative information, and features based on paths in a parse tree, which can often generalize better and can deal with long-distance dependencies. Furthermore, relation extraction systems typically rely on mention detection as a crucial way to identify the participants in relations and provide features that generalize well, since there are only a small, fixed number of mention types. It is also possible for these systems to construct features that capture semantic connections between the mentions by using semantic role labeling systems that decorate the parse-tree constituents with semantic roles. One of the primary goals of relation extraction is the structured representation of information in text, such that it may be entered into a database and searched with greater ease and more utility than, say, simple keyword search.

## References

1. Abney S (1991) Parsing by chunks. In: Principle-based parsing. Kluwer, Dordrecht
2. Berger A, Della Pietra S, Della Pietra V (1996) A maximum entropy approach to natural language processing. Comput Linguist 22(1):39–71

3. Bollacker K, Cook R, Tufts P (2007) Freebase: a shared database of structured general human knowledge. In: Proceedings of the 22nd national conference on artificial intelligence – volume 2, AAAI'07, Vancouver. AAAI, pp 1962–1963. <http://dl.acm.org/citation.cfm?id=1619797.1619981>
4. Bunescu RC, Mooney RJ (2005) A shortest path dependency kernel for relation extraction. In: Proceedings of the conference on human language technology and empirical methods in natural language processing, HLT'05, Vancouver. Association for Computational Linguistics, Stroudsburg, pp 724–731. doi:10.3115/1220575.1220666, <http://dx.doi.org/10.3115/1220575.1220666>
5. Bunescu RC, Mooney RJ (2006) Subsequence kernels for relation extraction. In: Weiss Y, Schölkopf B, Platt J (eds) Advances in neural information processing systems 18. MIT Press, London/Cambridge
6. Bunescu RC, Mooney RJ (2007) Learning to extract relations from the web using minimal supervision. In: Proceedings of the 45th annual meeting of the Association for Computational Linguistics (ACL'07), Prague. <http://www.cs.utexas.edu/users/ai-lab/pub-view.php?PubID=126761>
7. Caruana R (1997) Multitask learning. *Mach Learn* 28(1):41–75. doi:10.1023/A:1007379606734, <http://dx.doi.org/10.1023/A:1007379606734>
8. Chieu HL, Lee WS, Jiang J (2011) Transfer learning for adaptive relation extraction. Technical report, DSO National Laboratories, Singapore
9. Collins M, Duffy N (2001) Convolution kernels for natural language. In: Advances in neural information processing systems 14. MIT Press, London/Cambridge, pp 625–632
10. Cortes C, Vapnik V (1995) Support-vector networks. *Mach Learn* 20:273–297
11. Culotta A, Sorensen J (2004) Dependency tree kernels for relation extraction. In: Proceedings of the 42nd annual meeting on Association for Computational Linguistics, ACL'04, Barcelona. Association for Computational Linguistics, Stroudsburg. doi:10.3115/1218955.1219009, <http://dx.doi.org/10.3115/1218955.1219009>
12. Dietterich TG, Lathrop RH, Lozano-Pérez T (1997) Solving the multiple instance problem with axis-parallel rectangles. *Artif Intell* 89(1–2):31–71. doi:10.1016/S0004-3702(96)00034-3, [http://dx.doi.org/10.1016/S0004-3702\(96\)00034-3](http://dx.doi.org/10.1016/S0004-3702(96)00034-3)
13. Freund Y, Schapire R (1998) Large margin classification using the perceptron algorithm. *Mach Learn* 37(3):277–296
14. Gildea D, Jurafsky D (2002) Automatic labeling of semantic roles. *Comput Linguist* 28(3):245–288. doi:10.1162/089120102760275983, <http://dx.doi.org/10.1162/089120102760275983>
15. Glass M, Barker K (2011) Bootstrapping relation extraction using parallel news articles. In: Proceedings of the IJCAI workshop on learning by reading and its applications in intelligent question-answering, Barcelona
16. Hachey B (2006) Comparison of similarity models for the relation discovery task. In: Proceedings of the workshop on linguistic distances, LD'06, Sydney. Association for Computational Linguistics, Stroudsburg, pp 25–34. <http://dl.acm.org/citation.cfm?id=1641976.1641981>
17. Haussler D (1999) Convolution kernels on discrete structures. Technical report, UCSC-CRL-99-10, University of California at Santa Cruz
18. Hirano T, Matsuo Y, Kikui G (2007) Detecting semantic relations between named entities in text using contextual features. In: Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions, ACL'07, Prague. Association for Computational Linguistics, Stroudsburg, pp 157–160. <http://dl.acm.org/citation.cfm?id=1557769.1557815>
19. Hoffmann R, Zhang C, Weld DS (2010) Learning 5000 relational extractors. In: Proceedings of the 48th annual meeting of the Association for Computational Linguistics, ACL'10, Uppsala. Association for Computational Linguistics, Stroudsburg, pp 286–295. <http://dl.acm.org/citation.cfm?id=1858681.1858711>
20. Jiang J (2009) Multi-task transfer learning for weakly-supervised relation extraction. In: Proceedings of the joint conference of the 47th annual meeting of the ACL and the 4th international joint conference on natural language processing of the AFNLP: vol 2, ACL'09,

- Singapore. Association for Computational Linguistics, Stroudsburg, pp 1012–1020. <http://dl.acm.org/citation.cfm?id=1690219.1690288>
21. Jiang J, Zhai C (2007) A systematic exploration of the feature space for relation extraction. In: Proceedings of human language technologies: the conference of the North American chapter of the Association for Computational Linguistics (NAACL-HLT'07), Rochester, pp 113–120
  22. Jing H, Florian R, Luo X, Zhang T, Ittycheriah A (2003) HowtogetaChineseName(Entity): segmentation and combination issues. In: Collins M, Steedman M (eds) Proceedings of the 2003 conference on empirical methods in natural language processing, Sapporo, pp 200–207. <http://www.aclweb.org/anthology/W03-1026.pdf>
  23. Kambhatla N (2004) Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In: Proceedings of the ACL 2004 on interactive poster and demonstration sessions, Barcelona. Association for Computational Linguistics, Morristown, p 22. <http://dx.doi.org/10.3115/1219044.1219066>
  24. Kambhatla N (2006) Minority vote: at-least-N voting improves recall for extracting relations. In: Proceedings of the COLING/ACL on main conference poster sessions, COLING-ACL'06, Sydney. Association for Computational Linguistics, Stroudsburg, pp 460–466. <http://dl.acm.org/citation.cfm?id=1273073.1273133>
  25. Khayyamian M, Mirroshandel SA, Abolhassani H (2009) Syntactic tree-based relation extraction using a generalization of Collins and Duffy convolution tree kernel. In: Sarkar A, Rose CP, Stoyanchev S, Germann U, Shah C (eds) HLT-NAACL (student research workshop and doctoral consortium), Boulder. Association for Computational Linguistics, pp 66–71. <http://dblp.uni-trier.de/db/conf/naacl/naacl2009d.html#KhayyamianMA09>
  26. Lafferty J, McCallum A, Pereira F (2001) Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of ICML2001, Williamstown
  27. Luo X, Zitouni I (2005) Multi-lingual coreference resolution with syntactic features. In: Proceedings of the conference on human language technology and empirical methods in natural language processing, HLT'05, Vancouver. Association for Computational Linguistics, Stroudsburg, pp 660–667. doi:10.3115/1220575.1220658. <http://dx.doi.org/10.3115/1220575.1220658>
  28. McCallum A, Freitag D, Pereira FCN (2000) Maximum entropy Markov models for information extraction and segmentation. In: Proceedings of the seventeenth international conference on machine learning, ICML'00, Stanford. Morgan Kaufmann, San Francisco, pp 591–598. <http://dl.acm.org/citation.cfm?id=645529.658277>
  29. Mintz M, Bills S, Snow R, Jurafsky D (2009) Distant supervision for relation extraction without labeled data. In: Proceedings of the joint conference of the 47th annual meeting of the ACL and the 4th international joint conference on natural language processing of the AFNLP: volume 2, ACL'09, Singapore. Association for Computational Linguistics, Stroudsburg, pp 1003–1011. <http://dl.acm.org/citation.cfm?id=1690219.1690287>
  30. Nguyen TVT, Moschitti A (2011) Joint distant and direct supervision for relation extraction. In: Proceedings of 5th international joint conference on natural language processing, Chiang Mai, pp 732–740. <http://www.aclweb.org/anthology/I11-1082>
  31. NIST (2008) ACE (automatic content extraction) English annotation guidelines for relations. [http://projects.ldc.upenn.edu/ace/docs/English-Relations-Guidelines\\_v6.2.pdf](http://projects.ldc.upenn.edu/ace/docs/English-Relations-Guidelines_v6.2.pdf)
  32. Pan SJ, Yang Q (2010) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359. doi:10.1109/TKDE.2009.191. <http://dx.doi.org/10.1109/TKDE.2009.191>
  33. Rosenblatt F (1988) The perception: a probabilistic model for information storage and organization in the brain. In: Anderson JA, Rosenfeld E (eds) Neurocomputing: foundations of research. MIT, Cambridge, pp 89–114. <http://dl.acm.org/citation.cfm?id=65669.104386>
  34. Thrun S (1996) Is learning the n-th thing any easier than learning the first? In: Advances in neural information processing systems. MIT Press, London/Cambridge, pp 640–646
  35. Wu F, Weld DS (2010) Open information extraction using Wikipedia. In: Proceedings of the 48th annual meeting of the Association for Computational Linguistics, ACL'10, Uppsala. Association for Computational Linguistics, Stroudsburg, pp 118–127. <http://dl.acm.org/citation.cfm?id=1858681.1858694>

36. Zelenko D, Aone C, Richardella A (2003) Kernel methods for relation extraction. *J Mach Learn Res* 3:1083–1106. <http://dl.acm.org/citation.cfm?id=944919.944964>
37. Zhang M, Zhang J, Su J (2006) Exploring syntactic features for relation extraction using a convolution tree kernel. In: Proceedings of the main conference on human language technology conference of the North American chapter of the Association of Computational Linguistics, HLT-NAACL'06, New York. Association for Computational Linguistics, Stroudsburg, pp 288–295. doi:10.3115/1220835.1220872. <http://dx.doi.org/10.3115/1220835.1220872>

# Chapter 10

## Information Retrieval

Kareem Darwish

### 10.1 Introduction

In the past several years, some aspects of Semitic language, primarily Arabic, information retrieval (IR) have garnered a significant amount of attention. The main research interests have focused on retrieval of formal language, mostly in the news domain, with ad hoc retrieval, OCR document retrieval, and cross-language retrieval. The literature on other aspects of retrieval continues to be sparse or non-existent, though some of these aspects have been investigated by industry. The two main aspects where literature is lacking are web search and social search.

This survey will cover the two main areas: (1) a significant part of the literature pertaining to language-specific issues that affect retrieval; and (2) specialized retrieval problems, namely document image retrieval, cross-language search, web search, and social search.

### 10.2 The Information Retrieval Task

Information retrieval (IR) is concerned with the task of finding relevant documents in response to a user's information need. This information need can be expressed in a variety of ways, the most popular of which are text queries. Other expressions of information need can be in the form of:

1. An explicit indication by a user that (s)he likes a particular document or a set of documents. This is typical of document filtering.

---

K. Darwish (✉)  
Qatar Computing Research Institute, Doha, Qatar  
e-mail: [kdarwish@qf.org.qa](mailto:kdarwish@qf.org.qa)

2. A user's historical document viewing patterns. This is typical of recommender systems.
3. A find "more like this item" statement of need. The item in the query could be a text document or snippet, a picture, a video, a melody, etc.

Documents in information retrieval are containers of information that a user is interested in. Documents can be constituted of: text, such as news articles, reports, tweets, or webpages; images, such as photographs or document images; videos; or a combination of text, images, and videos. Information retrieval is mostly concerned with making the process of finding documents that match a user's information need more "effective" i.e. to find relevant, high-quality results.

Finding more relevant documents typically entails handling one or a combination of the following:

1. Appropriate language handling. Some languages require extra processing to ensure proper mapping between query words and document words. For example:
  - (a) Morphologically rich languages such as Arabic, Hebrew, and Hungarian require some stemming to remove attached articles such as determiners, coordinating conjunctions, and gender markers [37].
  - (b) Some language such as German and Finnish use word compounding to construct new words. These require word decompounding to extract appropriate units of meaning [21].
  - (c) Far-eastern languages such as Chinese typically don't use spaces between words and a user would have to segment words as they are reading. Appropriate word segmentation or the use of character n-grams is essential for effective retrieval in such languages [120].
2. Appropriate results ranking. Much work in IR focuses on ranking functions and algorithms. Traditional IR ranking techniques focused on developing ranking functions that rely on collection and document level term statistics. The main statistics include term frequency, which is the number times a term is mentioned in a document; document frequency, which is the number of documents in which a term appears; and document length, which is the number of terms in a document. One of the most popular formulas is the OKAPI BM-25 formula [102]. Follow-on work involved extending the OKAPI BM-25 to handle multiple fields [103]. A few years ago, there was a shift to using machine learning-based ranking functions that are trained using hundreds or thousands of examples using tens or hundreds of features. Such functions are commonly used for web search. The appropriate area of IR research is called "learning to rank" [84].
3. Term extraction. Aside from language handling in text documents, extraction of terms, which are searchable units, is essential particularly for non-text documents such as images, audio, music, and videos. For example, in the case of document images, which are just merely scanned manuscripts, document images are typically converted to text using optical character recognition (OCR) and the subsequent text is searched. However, due to the recognition errors presented by

OCR, further processing is required to perform appropriate term selection. One of the more popular approaches involves character n-gram extraction [35, 120].

4. Query reformulation. There is often a difference between a user's information need and the expressed query. Hence, reformulating a user's query can yield improved results. Some such techniques include:

- (a) Performing spelling correction on queries. E.g., “Andriod”  $\Rightarrow$  “Android” [6]
- (b) Adding synonyms to allow for better matching. E.g. “office to let”  $\Rightarrow$  “office (to|for) (let|rent)”
- (c) Adding expansion terms to queries. E.g. “SCAF”  $\Rightarrow$  “SCAF Egyptian military rulers”
- (d) Using alternative words. E.g. “VW Beetle”  $\Rightarrow$  “Volkswagen Beetle” [64].

Such reformulations can be recommended to users or they can be performed automatically.

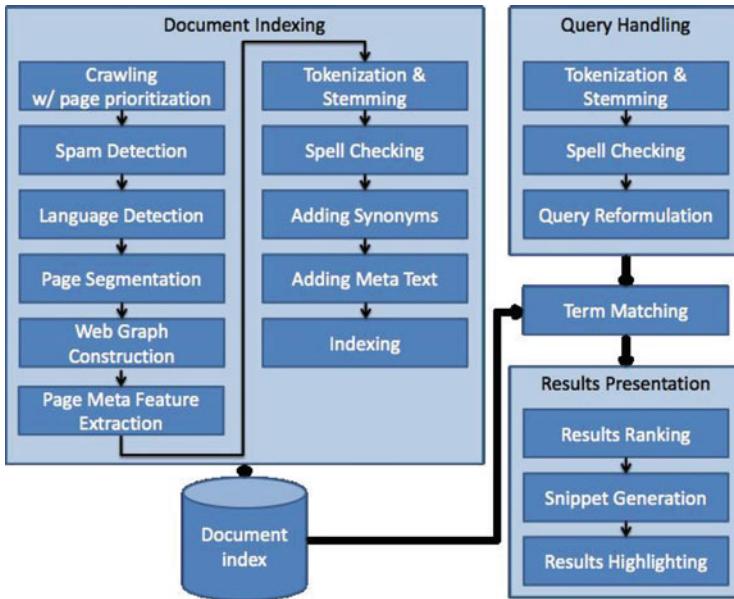
5. Specialized retrieval. This includes handling particular genres or problems that are specific to document collections, to the expression of information need, or to the mode of results delivery. Here are some examples of each:

- (a) Document collections:
  - (i) Tweets exhibit interesting properties such as the non-standard use of language, short lengths, and their often conversational nature [52].
  - (ii) Webpages are generally semi-structured with significant interconnections between webpages. These properties lend the ranking of web search results to “learning to rank” techniques [84].
- (b) Expression of need:
  - (i) Text queries are the most common form of expressed information need.
  - (ii) Non-text queries in which a user provides a picture, a video, or a sound.
  - (iii) A list of relevant or non-relevant documents.
- (c) Results delivery:
  - (i) A ranked list is the most common mode of delivering results.
  - (ii) Advertisements delivery with query results.
  - (iii) A continuously updated feed of microblogs of interest.

### **10.2.1 Task Definition**

In this survey we are concerned with two primary issues:

1. Exploring state-of-the-art language-specific techniques for retrieving documents written in Semitic languages, exclusively Arabic, Hebrew, and Amharic. Literature on the retrieval of other Semitic languages is scant at best. This includes:
  - (a) Addressing morphological and orthographic issues that affect retrieval.



**Fig. 10.1** Typical components of a search engine

- (b) Summarizing best practices.
- (c) Surveying evaluation methodologies, platforms, and campaigns.
2. Looking at retrieving Arabic documents, where Arabic is the best studied Semitic language from a retrieval perspective, in the context of the following:
  - (a) Cross-language retrieval.
  - (b) Document-image retrieval.
  - (c) Social search.
  - (d) Web search.

### 10.2.2 *The General Architecture of an IR System*

A typical Web IR system has the components shown in Fig. 10.1, where most of them are language independent. The most important language-dependent part is typically tokenization and stemming. Other components such as ranking, spell checking, and query reformulation depend somewhat on the language. Ranking may depend on the language due to the topology of the web graph for the language and the common practices of the users of a language. Spell checking and query reformulation are often affected by the morphology of the language, which is addressed

to some extent by stemming. Hence, stemming and tokenization take center stage in showing the language-specific issues that pertain to Semitic languages.

### 10.2.3 Retrieval Models

There are many popular retrieval models. These models can be categorized into four main categories. These categories are: set-based models, geometric or algebraic models, probabilistic models, and machine-learning based models. Due to the number of models that belong to these categories, we will give one or two example for each category.

#### Set-Based Models

These constitute some of the very first retrieval models that rely on matching between sets. One such model is the Boolean model that allows users to construct complex queries using the Boolean operators AND, OR, and NOT. The retrieved results would simply be documents that match the Boolean characterization without any particular order. The advantage of this model is that it is intuitive to users and it is easy to develop. However, developing complex queries can be cumbersome. Further, though all retrieved documents match the Boolean characterization, ranking results may be suboptimal. An extended Boolean model tries to overcome the ranking problem by incorporating term weights in the model [107].

#### Geometric or Algebraic Models

These models are motivated by geometric or algebraic formulations of the query–document matching problem. Some popular models that belong to this category are the vector space model and Latent Semantic Indexing (LSI).

**Vector Space Model:** As the name suggests, the vector space model represents both queries and documents as vectors whose dimensions are the words that appear in documents and queries [106]. Vector similarity functions such as cosine similarity can then be used to compute the similarity between the query vector ( $Q$ ) and document vector ( $D$ ). Cosine similarity computes the cosine of the angle  $\Theta$  between the vectors as follows:

$$\cos(\Theta) = \frac{\vec{Q} \cdot \vec{D}}{|\vec{Q}| |\vec{D}|} \quad (10.1)$$

There are three factors that contribute to the weight of terms in a document vector. They are term frequency (TF), document frequency (DF), and document length

(DL) [106]. TF is the number of time the term appears in the document. Intuitively, terms that are more representative of the content of a document, are often repeated in a document and would have high TF. Using TF improves precision and recall. DF is the number of documents in which a term appears. A term is more discriminating than another if it appears in fewer documents. Using DF improves precision.

Since documents vary in length, longer document have more terms and are more likely to have more instances of a term than shorter documents. This could lead to longer documents getting a higher score than shorter ones. Document length normalization is used to counteract this phenomenon. Cosine similarity, as in Eq. (10.1), accounts for document and query lengths by placing their lengths in the denominator.

**LSI:** It attempts to find the “latent” or hidden concepts in a document collection. Such concepts can be manifested by using multiple words with similar meanings. For example, the words “huge”, “large”, “gargantuan”, “enormous”, and “gigantic” have similar meanings and performing simple word matching would not capture such similarity. LSI constructs a matrix whose columns are the documents and whose rows are the terms. Singular value decomposition (SVD), which is a dimensionality reduction technique, is applied on the term–document matrix. Then documents and queries are mapped to the resultant lower dimensional space [43]. The advantage of LSI is that it captures hidden concepts or relationships between words. However, LSI’s main challenge is scalability in the presence of large numbers of documents and terms.

## Probabilistic Models

Probabilistic models attempt to compute the probability  $P(D|Q)$  of retrieving a document (D) given a query (Q) [101]. We will present here the probabilistic relevance model and latent Dirichlet allocation (LDA). Other popular models include inference networks and language modeling which are used in the Indri IR toolkit [94].

**Probabilistic Relevance Model:** This model attempts to estimate the similarity between a document and a query as follow:

$$\text{sim}(D|Q) = \frac{P(R|d)}{P(!R|d)} \quad (10.2)$$

where  $R$  is the set of relevant documents to a query. The model tries to estimate the probability that a document is relevant and the probability that the same document is not relevant. The most popular ranking formula based on this model is the OKAPI BM-25 formula [102]. The formula is computed as follows:

$$CFW = \log(N) - \log(DF) \quad (10.3)$$

where  $N$  is the number of documents in the collection, and  $DF$  is the document frequency.

$$\text{sim}(D, Q) = \frac{(K_1 + 1) \cdot CFW \cdot TF}{TF + K_1 \cdot \left( (1 - b) + b \cdot \frac{DL}{\text{Avg}(DL)} \right)} \quad (10.4)$$

where  $DL$  is number of terms in document  $j$ ,  $TF$  is term frequency, and  $K_1$  and  $b$  are tunable parameters.

**LDA:** LDA is a generative graphic model that assumes that latent (or hidden) topics underlie a collection of documents [20]. The model assumes that:

1. There are latent (or hidden) topics that are unobserved in the collection.
2. These latent topics can generate a set of words using a learnable distribution.
3. Documents are composed of words.
4. Documents may contain multiple latent topics.
5. The appearance of latent topics in general or in a document is governed by a learnable distribution.

Then given a query, we can infer the latent topic(s) that could have generated the words in the query. Then given the topical distribution, we can rank documents according the likelihood that they could have been generated from that distribution.

## Machine Learning-Based Models

Machine learning models or learning to rank methods involve the use of supervised or semi-supervised ranking techniques that are trained on judged query–document pairs. Such methods are popular in web search and are standard in all major web search engines. Typically query–document pairs are represented using hundreds of features and are judged for relevance on a multi-point scale. The objective function attempts to guess the proper judgment for a new query–document pair. Some commonly used machine learning techniques that use these models include RankNets [23] and SVM-Rank [66].

### 10.2.4 IR Evaluation

#### Evaluation Metrics

IR-effectiveness measures use precision and recall in different ways. For example, precision at  $n$  measures the average precision computed at every relevant document. Namely, descending from the top of a ranked list of retrieved documents, the precision is computed whenever a relevant document is found and then all precision values are divided by the maximum number of possible relevant documents that

**Table 10.1** Example output from ranked list

Rank	Relevant	Precision	Recall
1	No		0.00
2	Yes	0.50	0.50
3	No		0.50
4	No		0.50
5	Yes	0.40	1.00

could have been found when retrieving  $n$  documents [105]. This has the advantage of combining precision and recall in a single measure. For example, given a set five retrieved documents with two possible relevant documents as in Table 10.1: Average precision =  $\frac{0.5+0.4}{2} = 0.45$

Typical values of  $n$  are 1, 3, 10, and 1,000, where they respectively correspond to: the first retrieved document; documents that typically appear in the first screen of results without the need for scrolling; documents that appear in the first result page; and virtually all results that can be ever read by a user. Given multiple queries, the mean of their respective average precisions is called mean average precision (MAP). MAP computes a single value that balances between precision and recall for many topics that might vary in the number of relevant documents to estimate the expected effectiveness for unseen topics. MAP is one of the most commonly reported measures of retrieval effectiveness, particularly for ad hoc retrieval where binary relevance is assumed (i.e. documents are deemed relevant or not relevant).

In the presence of scaled relevance (as opposed to binary relevance) discounted cumulative gain (DCG) and normalized discounted cumulative gain (nDCG) are typically used [65]. The relevance judgments are often made on a 5-point scale, namely 2, 1, 0, -1, and -2 corresponding to perfect, excellent, good, poor, and not relevant respectively. nDCG measures how much information a user would gain if the user starts to read from the top of the ranked list, normalized by the maximum attainable gain. Typical reported nDCG values are nDCG@1, 3, and 10, which in web search respectively represent: the first result, which is the most likely result a user may click on; the results that typically appear on the first results screen without scrolling; and the results that appear on the first page, which users rarely go beyond.

## Evaluation Campaigns

**Overall Descriptions:** There are several IR evaluation campaigns that run periodically. Each of these campaigns is concerned with problems that are more relevant to particular parts of the world. The common goals of these campaigns are:

1. Bringing together researchers and practitioners in the area of IR.
2. Accelerating research and development in IR.
3. Providing realistic standard test collections for problems of interest to the research community and industry.
4. Developing evaluation metrics for different IR tasks.

The most popular evaluation campaigns are:

1. Text REtrieval Conference (TREC: <http://trec.nist.gov>). TREC was started in 1992 by the National Institutes of Standards and Technology (NIST) in the US. TREC has covered many IR problems such as:
  - (a) Ad hoc retrieval of English and non-English news search.
  - (b) Web search.
  - (c) Legal search.
  - (d) Biomedical search.
  - (e) Cross-language search, primarily of Arabic and Chinese using English queries.
  - (f) Enterprise search.
  - (g) Topic filtering.
  - (h) Video search.
  - (i) Social media search.
  - (j) Question answering.
2. Conference and Labs of the Evaluation Forum (CLEF).<sup>1</sup> CLEF, a European initiative, was started in 2000 and is mainly concerned with European languages. Some of the problems that have been addressed in CLEF include:
  - (a) Ad hoc search mainly for European languages.
  - (b) Image search.
  - (c) Cross-language search.
  - (d) Web search.
  - (e) Interactive search.
  - (f) People search.
3. National Institute of Informatics (NII) Test Collection for IR (NTCIR).<sup>2</sup> NTCIR was started in Japan in 1998 with a focus on Chinese, Japanese, and Korean. Some of the problems addressed by NTCIR include:
  - (a) Ad hoc search.
  - (b) Cross-language search.
  - (c) Question answering.
  - (d) Patent search.
  - (e) Web search.
  - (f) Summarization.
4. Forum for Information Retrieval Evaluation (FIRE).<sup>3</sup> FIRE was started in 2008 by the Information Retrieval Society of India to promote research in South Asian languages. Some of the problems addressed by FIRE include:

---

<sup>1</sup><http://www.clef-initiative.eu>

<sup>2</sup><http://research.nii.ac.jp/ntcir/index-en.html>

<sup>3</sup><http://www.isical.ac.in/~fire/>

- (a) Ad hoc search.
- (b) Cross-language search.
- (c) Social media search.

**Semitic Language-specific Evaluation Campaigns:** Evaluation campaigns that pertain to Semitic languages have been restricted to Arabic. There has been one primary evaluation of Arabic IR as part of TREC in 2001 and 2002 as part of the cross-language IR track. The track focused on retrieving Arabic documents using either Arabic or English queries. More details of the evaluation are provided later. Aside from the TREC IR-specific evaluation, there was a qualitative IR evaluation as part of the GALE project in the context of interactive audio and video search.

## Test Collections

Generally, evaluating retrieval effectiveness requires the availability of a test document collection, with an associated set of topics and relevance judgments. Relevance judgments are the mappings between topics and relevant documents in the collection. The cost of producing relevance judgments for a large collection is very high and dominates the cost of developing test collections [113]. There are three main methods of developing relevance judgments. The first is pooling, which involves manually assessing the relevance of the union of the top  $n$  documents from multiple retrieval systems for every topic [122]. For example, developing the relevance judgments for the 2002 Text REtrieval Conference (TREC) cross-language track involved assessing up to 4,100 documents for each of the 50 topics [97]. The second method is a manual user-guided search in which a relevance assessor manually searches and assesses documents for a topic until the assessor thinks that most relevant documents are found [124]. The third is exhaustively searching for documents that are relevant for each topic [120]. These three methods often miss relevant documents, and assessment of relevance is necessarily subjective, but studies have suggested that relevance judgments can be reliably used to correctly differentiate between retrieval systems provided that a sufficient number of queries are used [24, 104, 122]. Voorhees estimated the number of sufficient queries to be about 25 [122].

Test collections need to match the task at hand as much as possible in several aspects. Some of these aspects include:

1. Collection size: Collection size affects retrieval behavior, requiring a change in retrieval strategies. For example, performing deep morphology on Arabic to produce roots was shown to be the most effective strategy for searching Arabic on small collections of several hundred documents [12]. Later work on a large collection with 383,872 documents showed that using light stemming performed best [11]. There are indications that even using light stemming may in fact hurt retrieval effectiveness at web-scale where the document collection is in the range of hundreds of millions of documents.
2. Collection genre: Different genres exhibit different attributes that affect retrieval. For example, in searching news articles, document length normalization is

typically important. However, when searching tweets, document length normalization is less important because the variation in tweet lengths is generally small. Another example has to do with medical search where proteins and genes may have common names, scientific names, ontology identifiers, etc. and the use of synonymy becomes increasingly important [61].

3. Collection modality: Different collection modalities include text, images, videos, audio, document images, etc. An example of the effect of modality on retrieval has to do with document length normalization. Referring back to the equation for cosine similarity, a document length takes into account document frequency. Hence, documents with rare words (with low DF) are considered to be larger. When searching OCRed text, misrecognized words may have low DF causing an artificial inflation of document length. Singhal et al. [111] solved this problem for document length estimation using a document byte length. The OKAPI BM-25 formula uses a simple word count to perform normalization [102].
4. Document and collection structure: The structure of a document collection can complicate or ease retrieval. For example, news articles are typically disjoint with very few structural elements (headline, summary, article text, picture captions) that can be used to affect document ranking. Web documents on the other hand exhibit many features that can be used to enhance ranking. The most notable of these features is the existence of interlinks between pages [99].
5. Query formulation: It is important to construct queries that closely mimic queries that users actually issue. For example, web users often issue queries that contain spelling mistakes. Thus, constructing queries without spelling mistakes would hide real-life phenomena. Commercial web search engines such as Google ([www.google.com](http://www.google.com)) and Bing ([www.bing.com](http://www.bing.com)) are typically evaluated using actual queries from usage logs. Observing spelling mistakes in queries led all major commercial web search engines to include query spelling correction in the form of automatic correction or with suggested correction as in “did you mean:”.

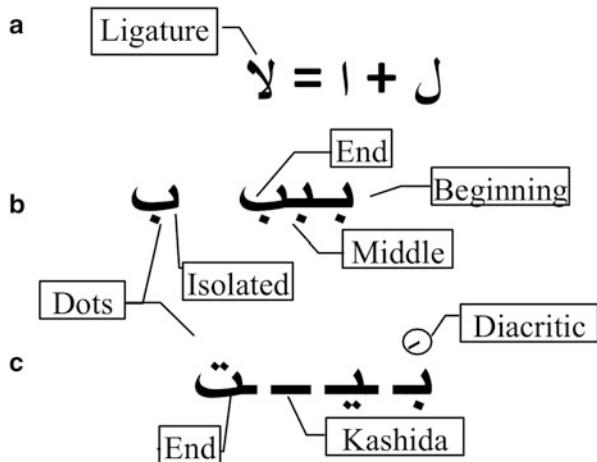
## 10.3 Semitic Language Retrieval

In this section, we look at major challenges affecting retrieving Semitic language documents while surveying a significant part of the literature pertaining to language-specific issues that affect retrieval. We also explore specialized retrieval problems, namely document image retrieval, web search, and social search.

### 10.3.1 *The Major Known Challenges*

#### Orthographic Challenges

Semitic languages exhibit some orthographic features that complicate retrieval. What follows is an exposition of some of these features.



**Fig. 10.2** (a) Example of a ligature, (b) the different shapes of the letter “ba” and (c) example of a diacritic, kashida, and three letters which are distinguishable from each other only by dots

**Arabic:** Arabic has a right-to-left connected script that uses 28 letters, which change shape depending on their positions in words. There are eight other letters (or letter forms). Fifteen of the letters contain dots to differentiate them from other letters. Letters may or may not have diacritics (short vowels), depending on the discretion of the producer of the document. Ligatures, which are special forms for some character sequences, and kashidas, which are symbols that extend the length of words, are often employed in printed text. Figure 10.2 demonstrates some of these orthographic features.

Further some letters are often used in place of others due to varying orthographic conventions, common spelling mistakes, and morphological transformations. These include<sup>4</sup>:

- ي “y” (ya) and ی “Y” (alef maqsoura).
- ہ (ha) and ۃ (ta marbouta).
- ل (alef), لـ (alef maad), لـ (alef with hamza on top), and لـ (alef with hamza on the bottom).
- و (hamza), وـ (hamza on w), and وــ (hamza on ya).

Optional diacritics, the use of kashidas and inconsistent spellings all complicate the retrieval. There are eight different diacritic marks in Arabic.

**Hebrew:** Hebrew has a right-to-left script with 22 letters. Some of the letters have special end-of-the-word forms. These letters are: ק (kaf), מ (meem), נ (nun), פ (pe), and צ (tsadi).

<sup>4</sup>Buckwalter encoding is used to Romanize Arabic text in this chapter.

The letter **וּ** is written with an optional dot on top of it to indicate whether it is pronounced as seen or sheen. Letters may or may not have diacritics (short vowels), depending on the discretion of the producer of the document. There are 13 different diacritics in Hebrew.

**Amharic:** Amharic uses syllable patterns or consonant–vowel pairs that are composed of 33 base consonants and 7 different vowels. For example, the letter that sounds like “s” in English has the following forms:

**ወ መ ንል መ ንል ሙ ዘ**

corresponding to the sounds of: ä, u, i, a, e, ə, and o respectively. Another phenomena in Amharic is the use of gemination where some letters are slightly elongated (or doubled) as in *alä* (he said) and *allä* (there is), but gemination is not indicated in writing.

## Morphological Challenges

Due to the morphological complexity of Arabic, Hebrew, and Amharic, proper handling of morphology is important for IR. In what follows, we describe peculiarities of the languages.

**Arabic:** Arabic words are divided into three types: nouns, verbs, and particles [2]. Particles are connection words such prepositions and pronouns. Arabic nouns and verbs are derived from a closed set of around 10,000 roots, which are linguistic units of meaning composed of 3, 4, or 5 letters [31]. Table 10.2 shows some of the words that can be generated from the root كتب (ktb). Arabic nouns and verbs are derived from roots by applying templates to the roots to generate stems. Applying templates often involves introducing infixes or deleting or replacing letters from the root. Table 10.3 shows some templates for three-letter roots. Further, stems may accept prefixes and/or suffixes to form words. Prefixes include coordinating conjunctions, determiners, and prepositions, and suffixes include attached pronouns and gender and number markers. Table 10.4 shows some of the possible prefixes and suffixes and their meanings. Further, plurals can be constructed using preset morphological transformations producing so-called broken plurals. Some examples of singular to broken plurals are: كتاب (ktAb) → كتب (ktb); درة (drp) → در (dr); and جامع (jAmE) → جوامع (jwAmE). The number of possible Arabic words is estimated to be  $6 \times 10^{10}$  words [5], with an estimated one million possible stems, and less than 10,000 roots. Stems are typically the units of meaning in Arabic, and hence are very important. Arabic words may have multiple valid analyses, only one of which is typically correct in context. For example the word وليد (wlyd) could be the proper name “Waleed” or may mean “and to hand” و ل يد + ل + يد (w+l+yd).

**Table 10.2** Some of the words that can be derived from the root form كتب “ktb”

كتب “ktb”	He wrote	يكتب “yktb”	He is writing	أكتب “lktb”	I write
كاتب “kAtb”	Writer	كتاب “ktAb”	Book	كتابه “ktAbh”	His book
وكتابه “wktAbh”	And his book	كتابهم “ktAbhm”	Their book	كتب “ktb”	Books

**Table 10.3** Some templates to generate stems from roots with examples from the root كتب “ktb”). “C” stands for the letters that are part of the root

CCC فعل	كتب “ktb”	فعال CCAC	كتاب “ktAb”	فاعل CACC	كاتب “kAtb”
(wrote)	(book)	(writer)			

mCCwC مفعول	مكتوب “mktwb”	فاعيل CCACyC	كتاتيب “ktAtyb”	فועל CCwC	كاتب “ktwb”
(something written)	(Quran schools)	(skilled writer)			

**Table 10.4** Some examples of prefixes and suffixes and their meanings

Examples of prefixes					
و “w”	And	ف “f”	Then	ال “Al”	The
ك “k”	Like	ل “l”	To	وال “wAl”	And the
Examples of suffixes					
هـ “h”	His	همـ “hm”	Their	هاـ “hA”	Her
كـ “k”	Your (singular)	كمـ “km”	Your (plural)	يـ “y”	My

**Table 10.5** Some of the words that can be derived from the root form כתב “ktb”

כתב “ktb”	write	כותבת “kwtbt”	I write (fm.)	כותבי “ktbty”	I wrote
-----------	-------	---------------	---------------	---------------	---------

**Hebrew:** Hebrew morphology is very similar to Arabic morphology. Words are typically derived from roots that are composed of 2, 3, or 4 letters, with 3-letter roots being the most common. Words are constructed from roots by inserting vowels, adding prefixes and suffixes, or doubling constants. Nouns can be singular or plural (or dual in classic Hebrew), and masculine or feminine. Nouns can accept a variety of affixes such as coordinating conjunctions, prepositions, determiners, pronouns, etc. Hebrew verbs may have up to seven conjugations (binyanim), four different tenses, and may be singular/plural or feminine/masculine [27]. Hebrew morphology is ambiguous with 55 % of Hebrew words having more than one analysis and 33 % having more than two analyses [25]. Table 10.5 shows some of the words derived from the root “ktb”. Table 10.6 shows some example Hebrew prefixes and suffixes.

**Amharic:** Like Arabic and Hebrew, words are derived from a set of roots. Stems are constructed using template and vowels patterns that may involve the insertion of vowels, adding prefixes and suffixes, or doubling constants. Stems may accept the attachment of prefixes and suffixes that include pronouns, determiners, plural and gender markers.

**Table 10.6** Some examples of prefixes and suffixes and their meanings

Examples of prefixes					
↑ “w”	And	↳ “t”	To	↗ “h”	The
Examples of suffixes					
↓ “k”	your (singular)	↔ “km”	your (plural)	↘ “y”	My

### **10.3.2 Survey of Existing Literature**

## Arabic Preprocessing

**Handling Orthography:** Prior to retrieval, the following features need to be handled: diacritics, kashidas, ligatures, and common spelling mistakes.

**Handling Diacritics:** Diacritics help disambiguate the meaning of words. For example, the two words علم Ealam (meaning flag) and علم Eelm (meaning knowledge) share the same letters “Elm” but differ in diacritics. One possible solution is to perform diacritic recovery. However, this approach has many problems, namely: the accuracy of state-of-the-art Arabic diacritizers on open domain text is typically below 90% [51]; diacritization is computationally expensive, often causing indexing of large amounts of text to be prohibitive; diacritization of previously unseen words is generally intractable; and word sense disambiguation, which is akin to diacritization, has been shown not to benefit retrieval [108]. The more widely adopted approach is to remove all diacritics. Though this increases ambiguity, retrieval is generally tolerant of ambiguity [108]. Further, this approach is computationally very efficient.

**Handling Kashidas and Ligatures:** Since kashidas are mere word elongation characters, they are typically removed. As for ligatures that are encoded as single characters in the codepage, they are transformed with the constituent letters. For example, the ligature ﴿ is transformed to l + ۚ.

**Common Spelling Mistakes:** For the case of the varying forms of alef, ya and alef maqsoura, and ha and ta marbouata, it is possible to build a system that would correct these common mistakes with about 99 % accuracy [44]. However, for reasons similar to those associated with diacritic recovery, doing so is not preferred. The most commonly used approach is letter normalization where:

- ي "y" and ئ "Y" are mapped to ي "y".
  - ة "h" and ؤ "p" are mapped to ه "f".
  - ا "A", ئ "I", ؎ ">", and ؋ "≤" are mapped to ا "A".
  - ، "&", ؎ "'", and ؊ "}" are mapped to ؎ " " [32].

The rationale for performing letter normalization is the same as that for diacritic removal.

**Handling Morphology:** Due to the morphological complexity of the Arabic language, some morphological processing would help recover the main units

of meaning, namely stems (or perhaps roots). Most early Arabic morphological analyzers generally used finite state transducers [18, 19, 71]. Their use is problematic for two reasons. First, they were designed to produce as many analyses as possible without indicating which analysis is most likely. This property of the analyzers complicates retrieval, because it introduces ambiguity in the indexing phase as well as the search phase of retrieval. Second, the use of finite state transducers inherently limits coverage, which is the number of words that the analyzer can analyze, to the cases programmed into the transducers. Other similar approaches attempt to find all possible prefix and suffix combinations in a word and then try to match the remaining stem to a list of possible stems [70, 87]. This approach has the same shortcomings as the finite transducer approach. Another approach to morphology is so-called light stemming. In this approach, leading and trailing letters in a word are removed if they match entries in lists of common prefixes and suffixes respectively. The advantage of this approach is that it requires no morphological processing and is hence very efficient. However, incorrect prefixes and suffixes are routinely removed. This approach was used to develop Arabic stemmers by Aljlayl et al. [11], Darwish and Oard [36], and Larkey et al. [78].

More recent analyzers can statistically perform word stemming. For example, Darwish attempted to solve this problem by developing a statistical morphological analyzer for Arabic called Sebawai that attempts to rank possible analyses to pick the most likely one [30]. Lee et al. [79] developed IBM-LM, which adopted a trigram language model (LM) trained on a portion of the manually segmented LDC Arabic Treebank in developing an Arabic morphology system, which attempts to improve the coverage and linguistic correctness over existing statistical analyzers such as Sebawai [30]. IBM-LM's analyzer combined the trigram LM (to analyze a word within its context in the sentence) with a prefix-suffix filter (to eliminate illegal prefix-suffix combinations, hence improving correctness) and unsupervised stem acquisition (to improve coverage). Lee et al. report a 2.9 % error rate in analysis compared to 7.3 % error reported by Darwish for Sebawai [79]. Diab [39] used an SVM classifier to ascertain the optimal segmentation for a word in context. The classifier was trained on the Arabic Penn Treebank data. Essentially, she treated the problem as a sequence labeling problem. She reports a stemming error rate of about 1 %. Although consistency is more important for IR applications than linguistic correctness, perhaps improved correctness would naturally yield great consistency. Follow-on work by Darwish and Ali [32] attempted to address shortcomings of existing stemmers that merely remove prefixes and suffixes. These shortcomings have to do with: (a) words that are typically borrowed from other languages that do not have standard stem forms; and (b) broken plurals. They used a generative character model to produce related stems and broken plurals, leading to significant improvements in retrieval effectiveness.

## Hebrew Preprocessing

Some of the earliest work on Hebrew retrieval started in the mid-1960s with the Responsa project [27]. The project was concerned with indexing more than 500,000 religious documents that included questions and answers contained in mainly Hebrew and partially Aramaic texts spanning 1,400 years. Due to the limited computing resources that were available at the beginning of the project, not all documents were indexed. By 1968 and 1977, researchers were able to index 518 and 24,000 documents respectively. Some research focused on manual versus automatic indexing and there were initial attempts to solve morphologically related problems in search.

**Handling Orthography:** Prior to retrieval, the following orthographic properties need to be handled: diacritics and letter normalization. Like in Arabic, diacritics help disambiguate the meaning of words. Like Arabic, it would be better to remove all diacritics. Though this increases ambiguity, retrieval is generally tolerant of ambiguity [108]. Further, this approach is computationally very efficient.

Concerning letter normalization, it is important to normalize letters that have special end-of-word forms. To show the importance of such normalization, consider the Hebrew word for “table” שולחן (sholchan) and its plural form שולחנות (sholhanot), which is simply the singular form שולחן (notice the change in the form of the first letter) plus insertion in the beginning of the plural form indicator וּ.

**Handling Morphology:** Due to the morphological complexity of Hebrew, some morphological processing would help recover the main units of meaning, namely stems (or perhaps roots). In the early work on the aforementioned Responsa Project, morphological equivalence classes were created from all the words in the document collection. In doing so, morphologically related words were heuristically produced from base noun and verb forms. When the user issued a query, the user was required to issue the query words in some standard base form, and the system produced morphologically equivalent words. Then the user was quizzed to choose which forms to include in their query. A morphological disambiguator, called KEDMA, was integrated into the Responsa Project to build concordances and perform morphological analysis [15].

Carmel and Maarek [25] developed a morphological disambiguator called Hemed to pick the most likely morphological analysis of a word – for cases where multiple analyses were possible. In their work, they used a morphological analyzer to produce all valid analyses, and then the analyses were ranked based on the likelihood of the underlying analysis patterns. They applied different thresholds on the allowed likelihood values of the disambiguator to conflate morphologically similar words. They showed significant improvements in recall at the expense of precision compared to not using morphological analysis at all, with an overall drop in F-measure. Their experiments were conducted on a small collection of 900 documents with 22 queries. Retrieval results on such small collections may not be indicative of behavior on larger collections.

Szpektor et al. [114] used a Hebrew morphological analyzer in the context of English–Hebrew cross-language IR. They used a finite state machine based Hebrew morphological analyzer that was developed by Yona and Wintner [126]. However, there were no reported treatment of Hebrew IR with comparison of different morphological techniques on a standard Hebrew test collection. We suspect that Hebrew would behave in a manner similar to Arabic where using roots for small collections is preferred, while the use of light stemming would be more preferred for larger collections. There is also a Hebrew analyzer for Lucene, a popular open source search engine, called HebMorph. More information on HebMorph and a treatment of possible variations in implementation are provided at <http://www.code972.com/blog/2010/06/open-source-hebrew-information-retrieval/>.

## Amharic Preprocessing

Some studies have looked at Amharic monolingual retrieval and Amharic–English and Amharic–French cross-language retrieval. The most notable work on Amharic monolingual retrieval is that of Alemayehu [8]. Alemayehu used a stemmer in which known prefixes and suffixes were iteratively removed. There were indications that using stemming or perhaps going to the root improved recall [9], but there were no reports of precision. Further, the collection used for evaluation had 548 documents with an associated set of 40 queries. Such small collections may lead to inconclusive results. Work has been done on improving matching against dictionary entries in cross-language retrieval between Amharic and English [14]. A stemmer was developed that attempts to generate all possible segmentations of a word, then attempts to match stems against entries in a dictionary. If multiple possible stems of a word matched dictionary entries, then the most frequent stem was chosen.

### **10.3.3 Best Arabic Index Terms**

#### Basic Preprocessing

In most Arabic IR work, the following steps have become standard [32, 78]:

- Removing diacritics and kashidas.
- Performing the aforementioned normalizations. These normalizations are performed in some web search engines.
- Replacing single character ligatures with constituent letters.
- Removing stop words (post morphological processing) [26, 45].

**Using Morphology:** Due to the morphological complexity of the Arabic language, much research has focused on the effect of morphology on Arabic IR. The goal of morphology in IR is to conflate words of similar or related meanings. Several early studies suggested that indexing Arabic text using roots significantly increases retrieval effectiveness over the use of words or stems [4, 12, 62]. However, all

the studies used small test collections of only hundreds of documents and the morphology in many of the studies was done manually.

A study by Aljlayl et al. [11] on a large Arabic collection of 383,872 documents suggested that lightly stemmed words, where only common prefixes and suffixes are stripped from them, were perhaps better index terms for Arabic. Similar studies by Darwish and Oard [35] and Larkey et al. [78] also suggested that light stemming is indeed superior to morphological analysis in the context of IR. Darwish compared light stemming to using Sebawai [30] and Larkey et al. [78] compared to using the Buckwalter morphological analyzer. The reported shortcomings of morphology might be attributed to issues of coverage and correctness. Concerning coverage, analyzers typically fail to analyze Arabized or transliterated words, which may have prefixes and suffixes attached to them and are typically valuable in IR. As for correctness, the presence (or absence) of a prefix or suffix may significantly alter the analysis of a word. For example, the word “Alksyr” is unambiguously analyzed to the root “ksr” and stem “ksyr.” However, removing the prefix “Al” introduces an additional analysis, namely to the root “syr” and the stem “syr.” Perhaps such ambiguity can be reduced by using the context in which the word is mentioned. For example, the word “ksyr” in the sentence “sAr ksyr” (and he walked like), the letter “k” is likely to be a prefix. The problem of coverage is practically eliminated by light stemming. However, light stemming yields greater consistency without regard to correctness.

However, a later study by Darwish et al. [37] suggested that using IBM-LM [79] statistically significantly improved retrieval effectiveness over using light stemming and other morphological analyzers. This is most likely due to the broad coverage of IBM-LM and the ability to rank the most likely analysis. Other work by Darwish and Ali [32] suggests that using AMIRA [39] and generating “similar” stems and broken plurals further improves retrieval effectiveness beyond other approaches due to the lower stemming error rate and broad coverage.

**Using Character N-grams:** The use of character 3- and 4-grams has been shown to be very effective in Arabic IR [35, 92]. The estimated average length of an Arabic stem is about 3.6 characters. Darwish and Oard [35] showed that character n-grams in fact outperform the use of light stemming. The effectiveness of character n-grams is perhaps due to:

- They consistently match stems of words.
- They are not bound by a preset vocabulary like morphological analysis.
- N-grams that include prefixes and suffixes appear more often than n-grams that include stems, and hence the use of inverse document frequency (IDF) would automatically demote the weight of n-grams that have prefixes and suffixes and promote the weight of n-grams that include stems.

The use of character n-grams should be done in conjunction with kashida and diacritic removal and performing proper letter normalization. The major downside of character n-grams is the increased processing of text and increased space storage requirement, where a six-letter word is replaced with four tokens (if character trigrams are used).

### ***10.3.4 Best Hebrew Index Terms***

#### **Basic Preprocessing**

Like Arabic, the following preprocessing would be appropriate:

- Removing diacritics.
- Performing the aforementioned normalizations of letters with different end-of-word forms.
- Removing stop words (post-morphological processing).

**Using Morphology:** Due to the morphological complexity of Hebrew, though the studies in the literature are inconclusive, performing morphological analysis with associated morphological disambiguation to ascertain the best analysis of a word would likely produce the best retrieval results [25, 114].

**Using Character N-grams:** The appropriate n-gram length depends on the average length of meaning-bearing units, typically stems, in a language. We suspect that character 3-, 4-, or 5-grams would be most appropriate to Hebrew. However, this requires experimentation to make a definitive conclusion.

### ***10.3.5 Best Amharic Index Terms***

Limited studies have suggested that stemming helps recall, but none reports clear numbers on the effect of stemming on precision. More work is required for Amharic.

## **10.4 Available IR Test Collections**

### ***10.4.1 Arabic***

There are several Arabic text collections that have been used for IR evaluations and can be readily obtained. These collections are:

**The TREC 2001/2002 Cross-language IR Track Collection:** Most recent studies on Arabic retrieval have been based on this collection [49, 97]. For brevity, the collection is referred to as the TREC collection. The collection contains 383,872 articles from the Agence France Press (AFP) Arabic newswire. Twenty-five topics were developed for the 2001 evaluation and an additional fifty topics were developed for 2002. Relevance judgments were developed at the LDC by manually judging a pool of documents obtained from combining the top 100 documents from all the runs submitted by the participating teams to TREC's cross-language track in 2001 and 2002. The 2001 topics and their relevance judgments are suspect due to the large number of relevant documents being contributed to the pool by only one of

the participating teams [49]. For the 2002 topics, the number of known relevant documents ranges from 10 to 523, with an average of 118 relevant documents per topic [97]. This is presently the best available large Arabic information retrieval test collection. The TREC topics include a title field that briefly names the topic, a description field that usually consists of a single sentence description, and a narrative field that is intended to contain any information that would be needed by a human judge to accurately assess the relevance of a document [49].

**Zad Al-Mead Collection:** This collection was built from Zad Al-Mead, a medieval book that is free of copyright restrictions and for which a free electronic copy is available. The book, written in the fourteenth century by a Muslim theologian, consists of 2,730 separate documents that address a variety of topics such as mannerisms, history, jurisprudence and medicine. Darwish [31], a native speaker of Arabic, developed 25 topics and exhaustively searched the collection for relevant documents. The number of relevant documents per topic ranges from zero (for one topic) to 72, averaging 18. The average query length is 5.5 words.

Other non-publicly available collections were created by different commercial entities to gauge the effectiveness of their IR systems. For example, Hefny et al. [60] reported on Arabic–English cross-language web retrieval results using a set of 200 cross-language queries that were run against Bing.

#### 10.4.2 Hebrew

The Responsa collection is perhaps the largest collection reported in the literature with 500,000 documents [27]. Carmel and Maarek [25] developed a collection of 900 documents with 22 associated queries. Szpektor et al. [114] used a collection of descriptions of museum artifacts to construct a document collection. Their study was preliminary and only used five queries, which would not be sufficient to make conclusive conclusions.

#### 10.4.3 Amharic

The only collection reported on in the literature has 548 documents with an associated set of 40 queries [9]. The collection was obtained from the Institute of Ethiopian Studies and includes documents from 1974 to 1991 on anti-government propaganda. Such small collections may lead to inclusive results.

### 10.5 Domain-Specific IR

In what preceded, we surveyed linguistic challenges associated with ad hoc retrieval of Arabic, Hebrew, and Amharic. In the following, we explore IR applications for different domains. Though we restrict the treatment of domain-specific IR

applications to Arabic, the treatment can be instructive in addressing similar applications in other Semitic languages. We address four different applications. Two of these, namely cross-language IR (CLIR) and document image retrieval, are fairly well studied. For the other two, namely web search and social search, the literature is fairly scant, but we will explore some of their challenges.

### **10.5.1 Arabic–English CLIR**

CLIR is the process of finding documents in one language based on queries in a different language [96]. One of the central issues in this work pertains to query translation. Query translation has been explored extensively in the context of cross-language information retrieval, where a query is supplied in a source language to retrieve results in a target language. Two of the most popular query translation approaches are dictionary based translation (DBT) methods [81] and machine translation (MT) [125]. DBT methods usually involve replacing each of the source language words with equivalent target language word(s). Since a source language word may have multiple translations, optionally the most popular translation or  $n$  best translations are used. Since web search engines typically use an AND operator by default, using multiple translations may cause translated queries to return no results. Another alternative is to use a synonym operator, which has been shown to be effective in CLIR [81, 100]. A synonym operator can be approximated in web search by using an OR operator between different translations. Some online search engines have synonym operators, such as the “word” operator in Bing. However, the use of a weighted synonym operator, where each translation is assigned a confidence score, is not supported in popular web search engines, though it has been shown to be effective in cross-language search [123]. MT has been widely used for query translation [121, 125]. Wu et al. [125] claim that MT outperforms DBT. Their claim is sensible in the context of web search for two reasons: (a) MT attempts to optimally reorder words after translation and web search engines are typically sensitive to word order; and (b) MT produces a single translation without any synonyms or OR operators, for which the rankers of web search engines are not tuned.

Another approach of interest here is the so-called cross-lingual query suggestion [46, 48]. This approach involves finding related translations for a source language query in a large web query log in the target language. Gao et al. [46] proposed a cross-language query suggestion framework that used a discriminative model trained on cross-language query similarity, cross-language word co-occurrence in snippets of search results, co-clicks from both queries to the same URL, and monolingual query suggestion. Recent work by Hefny et al. [60] extends the work of Gao et al. [46] by using alternative features, such as phonetic similarity, relative query lengths, and cross-language coverage.

The second issue involves merging multilingual results, where results from multiple languages that the user may speak are combined into a single ranked list. For results merging in general, there are several simple techniques in the literature

such as score-based merging, round-robin merging, and normalized score-based merging [83]. Score-based merging assumes that scores in different ranked lists are comparable, which cannot be guaranteed. This can be solved by normalizing scores from different ranked lists. Round-robin merging assumes that different ranked lists have a comparable number of relevant documents [83]. Si and Callan [110] used a logistic regression based model to combine results from different multilingual ranked lists in the context of ad hoc search. Tsai et al. [119] also addressed the problem in the context of ad hoc search. They used document, query, and translation based word-level features to train an FRank-based ranker whose score was then linearly combined with the BM-25 score of each document. Rankers of web search engines typically consider many more features such as link-graph, document structure, and log based features. Gao et al. [47] used a Boltzman machine to learn a merging model that takes cross-language relations between retrieved documents into account. They tested their approach on ad hoc as well as web search. In the context of web search, they evaluated their approach using queries that have equivalents in the query log of the target language, which means that these queries would likely benefit from cross-language results merging. Hefny et al. [60] employed a supervised learning rank model, namely SVMRank, to merge multiple ranked lists into a single list. SVMRank was trained using the relevance judgments for query–result pairs which were used to extract pairwise order constraints.

The third central issue is ascertaining when cross-language web search would yield good results. The literature is relatively sparse on this issue. Kishida [72] examined “ease of search” and translation quality as means to perform cross-language query prediction. Another less related work examined when and when not to translate query words [80]. Hefny et al. [60] proposed the use of query logs, through so-called query picking, to determine if cross-language search would be effective or not. Essentially, given a source language query, if an equivalent query is found in a large query-log, then the query would likely produce good results.

Most of the work on Arabic–English CLIR was conducted as part of the TREC 2001 and 2002 CLIR track [49,97]. The track-associated work was conducted on the aforementioned collection of 383,872 news articles from the AFP Arabic newswire associated with 75 topics and relevance judgments, and it focused exclusively on searching an Arabic corpus using English queries. Studied on this collection focused on a variety of issues such as transliterating named entities [3], stemming [10,26], combining multiple transliteration resources [36], combining multiple translations [123], interactive retrieval [59], and blind relevance feedback [93].

Hefny et al. in [60] focused on searching the English web using Arabic queries. Their work addressed other problems relating to combining multilingual results and cross-language query performance prediction.

### 10.5.2 Arabic OCR Text Retrieval

This refers to searching scanned manuscripts. The most notable method for searching such scanned manuscripts involves performing OCR and then searching the resultant text. Although OCR is fast, OCR output typically contains errors. The errors are even more pronounced in OCRed Arabic text due to Arabic's orthographic and morphological properties. The introduced errors adversely affect linguistic processing and retrieval of OCRed documents.

#### Arabic OCR

The goal of OCR is to transform a document image into character-coded text. The usual process is to automatically segment a document image into character images in the proper reading order using image analysis heuristics, apply an automatic classifier to determine the character codes that most likely correspond to each character image, and then exploit sequential context (e.g., preceding and following characters and a list of possible words) to select the most likely character in each position. The character error rate can be influenced by reproduction quality (e.g., original documents are typically better than photocopies), the resolution at which a document was scanned, and any mismatch between the instances on which the character image classifier was trained and the rendering of the characters in the printed document. Arabic OCR presents several challenges, including:

- Arabic's cursive script in which most characters are connected and their shapes vary with position in the word. Further, multiple connected characters may resemble other single characters or combinations of characters. For example, the letter ش (\$) may resemble نت (nt).
- The optional use of word elongations and ligatures, which are special forms of certain letter sequences.
- The presence of dots in 15 of the 28 to distinguish between different letters and the optional use of diacritic which can be confused with dirt, dust, and speckle [35]. The orthographic features of Arabic lead to some characters being more prone to OCR errors than others.
- The morphological complexity of Arabic, which results in an estimated 60 billion possible surface forms, complicates dictionary-based error correction.

There are a number of commercial Arabic OCR systems, with Sakhr's Automatic Reader and Shonut's OmniPage being perhaps the most widely used [75, 76]. Most Arabic OCR systems segment characters [50, 56, 57, 75], while a few opted to recognize words without segmenting characters [13, 86]. A system developed by BBN avoids character segmentation by dividing lines into slender vertical frames (and frames into cells) and uses an HMM recognizer to recognize character sequences [86].

## OCR Degraded Text Retrieval

Retrieval of OCR degraded text documents has been reported on for many languages, including English [53, 68, 115, 117]; Chinese [120]; and Arabic [30].

For English, Doermann [40] reports that retrieval effectiveness decreases significantly for OCRed documents with an error rate at some point between 5 and 20 %. Taghva reported experiments which involved using English collections with 204–674 documents that were about 38 pages long on average [116, 117]. The documents were scanned and OCRed. His results show negligible decline in retrieval effectiveness due to OCR errors. Taghva's work was criticized for being done on very small collections of very long documents [120]. Small collections might not behave like larger ones, and thus they might not be reflective of real-life applications in which retrieval from a large number of documents is required [54]. Similar results for English were reported by Smith [112] in which he reported no significant drop in retrieval effectiveness with the introduction of simulated OCR degradation in which characters were randomly replaced by a symbol indicating failure to recognize. These results contradict other studies in which retrieval effectiveness deteriorated dramatically with the increase in degradation. Hawking reported a significant drop in retrieval effectiveness at a 5 % character error rate on the TREC-4 “confusion track” [58]. In the TREC-4 confusion track, approximately 50,000 English documents from the federal registry were degraded by applying random edit operations to random characters in the documents [68]. The contradiction might be due to the degradation method, the size of the collection, the size of the documents, or a combination of these factors. In general retrieval effectiveness is adversely affected by the increase in degradation and decrease in redundancy of search terms in the documents [40].

Several studies reported the results of using n-grams. A study by Harding et al. [53] compared the use of different length n-grams to words on four English collections, in which errors were artificially introduced. The documents were degraded iteratively using a model of OCR degradation until the retrieval effectiveness of using words as index terms started to significantly deteriorate. The error rate in the documents was unknown. For n-grams, a combination of 2- and 3-grams and a combination of 2-, 3-, 4-, and 5-grams were compared to words. Their results show that n-gram indexing consistently outperformed word indexing, and combining more n-grams was better than combining fewer. In another study by Tseng and Oard, they experimented with different combinations of n-grams on a Chinese collection of 8,438 document images and 30 Chinese queries [120]. Although ground truth was not available for the image collection to conclude the effect of degradation on retrieval effectiveness, the effectiveness of different index terms were compared. They experimented with unigrams, bigrams, and a combination of both. Chinese words were not segmented and bigrams crossed word boundaries. The results of the experiments show that a combination of unigrams and bigrams consistently and significantly outperforms character bigrams, which in turn consistently and significantly outperform character unigrams.

For Arabic, Darwish and Oard [35] reported that character 3- and 4-grams were the best index terms for searching OCR degraded text. They conducted their experiments on a small collection of 2,730 scanned documents. In general, blind relevance feedback does not help for the retrieval of OCR degraded documents [33, 77, 120].

## Building an OCR Degraded Collection

To build an OCR degraded test collection, there are three common approaches:

- **Printed Document Domain:** which involves building a collection by scanning printed documents and performing OCR. This approach is most desirable because the errors in the text are due to real OCR degradation and not a model of the degradation. However, building large test collections of several hundred thousand documents with a set of topics and relevance judgments can be very expensive. Therefore, the collections reported in the literature were all small. One such collection is a Chinese collection of 8,438 documents which was developed by Tseng and Oard [120]. The documents in Tseng's collection varied widely in their degradation level and there was no accurately character-coded version (OCR ground truth) for the collection. Abdelsapor et al. [1] developed a collection of Arabic OCRed document images by randomly picking approximately 25 pages from 1,378 Arabic books from Bibliotheca Alexandrina (BA) forming a set of 34,651 printed documents. Associated with the collection is a set of 25 topics that were developed using an iterative search and judge method [109]. The books cover a variety of topics including historical, philosophical, cultural, and political subjects and the printing dates of the books range from the early 1920s to the present. Again, no ground truth is available for the collection. Having ground truth helps show the effect of degradation on retrieval. Developing OCR ground truth is typically laborious, involving either correction of OCR errors in the OCRed version of the collection or manual reentry of the collection's text. Lam-Adesina and Jones [77] reported on a collection that they developed from the Spoken Document Retrieval (SDR) track collection. The stories in the collection was printed using different formats and fonts, and the resulting hardcopies were scanned and OCRed. Associated with the collection of 21,759 news stories are rough or closed-caption quality transcripts and 50 topics that were developed for the SDR track [77]. Darwish [31] reported on a small collection of 2,730 documents of scanned and OCRed document images for which ground truth exists.
- **Image Domain:** which involves building a collection by synthesizing document images from a preexisting non-degraded collection, degrading the document images, and performing OCR on them. Synthesizing document images is done by typesetting the text into an image [41]. To degrade document images, different document degradation models were developed [16, 17, 41, 74]. The models parameterize different aspects of the document images such as font size, page skew, horizontal and vertical offset, horizontal and vertical scaling, blur,

resolution, pixel jitter, and sensitivity. With degradation modeling, document image collections of varying degradation levels with corresponding ground truth can be developed automatically. To verify suitability of the generated document image collections for further OCR research, tests were developed. It is claimed that a degradation model is valid if the confusion matrices that result from automatically degraded documents are similar to the ones that result from real documents [73, 82, 85]. The advantage of this approach for creating OCR-degraded collections is that it is inexpensive, the degradation level can be tuned, and OCR ground truth is automatically available. Although OCR researchers prefer real document images and real OCR output [120], this approach might be suitable for IR experimentation, but its suitability for IR needs to be verified.

- **Text Domain:** building a collection by synthesizing OCR degradation. This approach has the advantage of being able to use a preexisting non-degraded collection with its topics and relevance judgments to rapidly build a new degraded collection. This approach was used in developing many degraded text collections [29, 53, 55, 112]. The degradation models ranged between ones that attempted to accurately model OCR degradation [53] to ones that randomly introduced errors [112]. Mittendorf and Schuble [95] argued that using synthetic OCR degradation does not lead to the variations of recognition probabilities, which affect ranking permutations the most, that are observed in real OCR degradation. Darwish [31] introduced formal tests to verify that the modeled OCR degradation has similar effect on retrieval as real OCR degradation.

## OCR Error Correction

Much research has been done to correct recognition errors in OCR degraded collections. Reducing the number of errors in the text may lead to improved IR effectiveness. There are two main approaches to error correction, namely, word level and passage level. Some of the kinds of word-level post-processing include the use of dictionary lookup [22, 28, 63, 67], character [86, 116] and word n-grams [63, 88], frequency analysis, and morphological analysis [42, 98]. Passage-level post-processing techniques include the use of word n-grams [88], word collocations [63], grammar [7], conceptual closeness [63], passage-level word clustering [116] (which requires handling of affixes for Arabic [38]), and linguistic and visual context [63]. Dictionary lookup is used to compare recognized words with words in a lexicon [22, 28, 63, 67]. Finding the closest matches to every OCRed word in the dictionary is attempted, and the matches are then ordered using a character-level error model in conjunction with either a unigram probability of the matches in the text [67] or an n-gram language model [88, 118]. Another approach suggested by Magdy et al. [89] involves the use of multi-OCR output fusion. In this approach multiple OCR systems, which typically have different classification engines with different training data, are used to recognize the same text. The output of the different OCR systems is then fused by picking the most likely recognized sequence of tokens using language modeling [89].

## Other Approaches

Two other techniques are worth mentioning here. The first is query garbling in which an error model is used to generate garbled versions of query words [34]. Query garbling is akin to translating queries to the document space and much of the work on CLIR would apply. Another approach does not involve using OCR at all. The basic idea is that similar connected elements in printed documents are clustered and represented with IDs, which are then used to generate equivalent textual representations. The resultant representations are indexed using an IR engine and searched using the equivalent IDs of the connected elements in queries [90].

### 10.5.3 Arabic Social Search

Though most of the 350 million natives of the Middle East can read and understand Modern Standard Arabic (MSA), they generally use dramatically different languages in daily interactions. With the spread of online social interaction in the Arab world, these dialects started finding their way into online social interaction. There are six dominant dialects, which are Egyptian (85+ million speakers), Moroccan (75+ million), Levantine (35+ million), Iraqi (25 million), Gulf (25+ million), and Yemini (20+ million).<sup>5</sup> Aside from those, there are tens of different Arabic dialects along with variations within the dialects.

There are several factors that make dialects different. Different dialects make different lexical choices to express certain concepts, though in many cases the lexical choices have proper Arabic roots. For example, the concept corresponding to “I want” is expressed as “EAwz” عاوز in Egyptian, “Abgy” ابغى in Gulf, “Aby” ابى in Iraqi, and “Bdy” بدې in Levantine. The most popular MSA form is “Aryd” أريد. In certain situations, some words are used to mean different or completely opposite things in different dialects.

The pronunciations of different letters are often different from one dialect to another. One of the letters with most variations in pronunciation is the letter “q” (ق). In MSA, it is typically pronounced similar to the English “q” as in the word “quote”. However, it is pronounced as an “a” in Egyptian and Levantine (as in “Alpine”), as a “ga” in Gulf (as in “gavel”), and as a “gh” in Sudanese (as in “Ghana”). Another is the letter “j” (ج) which is pronounced as a soft “g” like in “gavel” in Egyptian and Yemini dialects and as “j” like in “John” in most other dialects. Differing pronunciations reflect on the way people spell dialectic text.

As for the influence of foreign languages, different countries have had strong interactions with other societies that speak different languages. Some of the interactions have been the result of geographic juxtaposition or military conflict. For example, Lebanon was occupied by France, Egypt was occupied by Britain,

---

<sup>5</sup>[http://en.wikipedia.org/wiki/Varieties\\_of\\_Arabic](http://en.wikipedia.org/wiki/Varieties_of_Arabic)

Iraq has a large Kurdish minority and borders Iran, and Algeria has a large Berber population. Such interaction has caused an influx of foreign words into the dialects of different countries. For example, in some of the Gulf countries you may hear the phrase “shebb el-lait” (شب الليت) meaning “turn on the light”. In this particular example “shebb” is Arabic and “lait” is a deformed form of the word “light”. In more extreme situations as in countries like Algeria, people mix Arabic, French, and Berber together, while using grammatical rules of one language, typically Arabic, to construct sentences.

All these factors have complicated interactions between people from different parts of the Arab world. Social media platforms have caused large portions of populations to express themselves in writing. Though MSA was the traditional de facto modus operandi for writing Arabic, people became more inclined to use their dialectic Arabic in their written daily interactions on social media sites. Some notable trends started appearing in text used on social platforms, such as:

1. The writing of Arabic words using phonetically equivalent Latin characters. These tendencies have given rise to truly interesting language use. Consider the following Facebook post “Ana nazel el goma3a el gayah el tahrir inshAllah” which corresponds to “I am going next Friday to Tahrir – God willing.” In the post, dialectic Egyptian Arabic is written using Latin letters with the corresponding Arabic is نازل يوم الجمعة إن شاء الله أانا. Notice also, that the word “goma3a”, which corresponds to Friday in English, starts with the letter “jeem” (ج) but is actually written in Latin letters to correspond to the Egyptian pronunciation of the letter. There is an opposite phenomena where English words are written using Arabic letters. There are several commercial programs that aim to convert from Romanized Arabic text to Arabic such as Yamli ([www.yamli.com](http://www.yamli.com)), Google Ta3reeb (<http://www.google.com/ta3reeb/>), and Microsoft Maren ([www.getMaren.com](http://www.getMaren.com)). Such Romanized Arabic text requires specialized language detection to properly identify it in text. Darwish [69] adapted transliteration work with language modeling to perform offline conversion of Arabizi to Arabic.
2. The mixed language text where many of the social platform users may speak multiple languages, mainly Arabic and English or Arabic and French. Consider the following tweet: “yeaaa aham 7aga do everything with pride” where the Arabic is mixed with English to say “yes, the most important thing: do everything with pride.”
3. The use of dialectic words that may have different morphological forms than MSA. For example, Egyptian Arabic uses a negation construct similar to the “ne pas” negation construction in French. Consider the Egyptian Arabic word ملعبيش (mlEbt\$) which means “I did not play” and is composed of “m+I+Ebt+\$”. Such constructs are not handled by existing MSA morphological analyzers.
4. The use of phonetic transcription of words to match how words are pronounced in dialects. For example, the word صدق (Sdq) meaning “truth” or “honestly” is often written as صخ (Sj) to match the Gulf dialect.
5. Creative spellings, spelling mistakes, and word elongations are ubiquitous in social text.

**Table 10.7** Arabic processing in Google and Bing

Feature	Google	Bing
Diacritics removal	✓	✓
Kashida removal	✓	✓
Letter normalization	Partial	✓
Light stemming	X	X
Stop word removal	X	X

6. The use of new words that don't exist in the language such as words expressing laughter (e.g. لول (lwl) corresponding to Laugh Out Loud (LOL)) and using Arabic words to mean new things (e.g. طحن (THn) meaning "grinding" in MSA, but intended to mean "very").

All these factors complicate Arabic social text retrieval. There is a recent preliminary paper by Magdy et al. [91] that tries to address some of these issues, but many more issues are yet to be tackled.

#### 10.5.4 Arabic Web Search

To the best of our knowledge, there is no publicly published work on Arabic web search. Google and Bing have been working on improving Arabic web search for several years, but the vast majority of their work is unpublished. Google and Bing claim to index roughly 3 billion and 210 million Arabic pages respectively. To ascertain the relative number of indexed Arabic pages, two Arabic stop words (في-in and من-from) were used as queries to search in the two search engines. There are several challenges that need to be addressed for Arabic search, namely:

1. Efficiency: Due to the large scale of the web, all processing must be very efficient. Hence, performing complex processing such morphological analysis and latent semantic indexing becomes prohibitive.
2. Language handling: Due to efficiency related constraints, minimalist language processing is done. Table 10.7 summarizes current support in Google and Bing. This information was obtained by searching using different queries that were designed to test the specific features in the engines. Both search engines emulate stemming using query alteration. Alteration may involve expanding query words using their synonyms and/or morphological variants.<sup>6</sup>
3. Ranking: There are fundamental differences between the English and Arabic webs. Some of these differences are:
  - (a) Unlike the English web, the Arabic web is sparsely connected. This makes features such as PageRank less useful.

---

<sup>6</sup>This is based on communication with people working on different web search engines.

- (b) The relative size of the Arabic forum content is disproportionately larger compared to the English forum content. English forum content is often considered of lower quality. However, such content is often of high-quality in Arabic. Considering that most Arab companies and institutions lack web presence, many entities resort to forums to post information.
  - (c) The Arabic web is significantly smaller than the English web, and the size of available Arabic resources such as Wikipedia is dwarfed by those available for English. To contrast both, Arabic Wikipedia has slightly more than 259,000 articles compared to more than three million English articles.
4. Index coverage: Index coverage of good quality pages is critical to the effectiveness of web search. Much Arabic content is nestled inside large portals such as YouTube, WordPress, Facebook, etc. Short of indexing everything on the web in general and on such sites in specific, indexing the Arabic specifically may be challenging. There are other issues that are not necessarily unique to Arabic but would need handling nonetheless such as spam detection, query spell checking, web page segmentation, query reformulation, and results presentation.

## 10.6 Summary

This chapter presents a treatment of some of the language-specific issues that affect the retrieval of three Semitic languages, namely Arabic, Hebrew, and Amharic. Literature on the retrieval of other Semitic languages is limited or non-existent. The chapter covers morphological as well as orthographic features of the different languages and state-of-the-art methods to handle these features. Further, the chapter addresses certain domain-specific IR problems, namely cross-language IR, OCR text retrieval, social search, and web search. The literature on the two latter problems is scant at best, and much interesting research is required to address them.

## References

1. Abdelsapor A, Adly N, Darwish K, Emam O, Magdy W, Nagi M (2006) Building a heterogeneous information retrieval collection of printed Arabic documents. In: LREC 2006, Genoa
2. Abdul-Al-Aal A (1987) An-Nahw Ashamil. Maktabat Annahda Al-Masriya, Cairo
3. AbdulJaleel N, Larkey LS (2003) Statistical transliteration for English–Arabic cross language information retrieval. In: CIKM’03, New Orleans, 3–8 Nov 2003
4. Abu-Salem H, Al-Omari M, Evens M (1999) Stemming methodologies over individual query words for Arabic information retrieval. JASIS 50(6):524–529
5. Ahmed M (2000) A large-scale computational processor of the Arabic morphology, and applications. Faculty of Engineering, Cairo University, Cairo
6. Ahmad F, Kondrak G (2005) Learning a spelling error model from search query logs. In: Proceedings of HLT-2005, Vancouver

7. Agirre E, Gojenola K, Sarasola K, Voutilainen A (1998) Towards a single proposal in spelling correction. In: Proceedings of COLING-ACL'98, San Francisco, pp 22–28
8. Alemayehu N (1999) Development of a stemming algorithm for Amharic language text retrieval. Ph.D. thesis, Dept. of Information Studies, University of Sheffield, Sheffield
9. Alemayehu N, Willett P (2003) The effectiveness of stemming for information retrieval in Amharic. *Electron Libr Inf Syst* 37(4):254–259
10. Aljail M, Frieder O (2002) On Arabic search: improving the retrieval effectiveness via a light stemming approach. In: CIKM'02, McLean
11. Aljail M, Beitzel S, Jensen E, Chowdhury A, Holmes D, Lee M, Grossman D, Frieder O (2001) IIT at TREC-10. In: TREC 2001, Gaithersburg
12. Al-Kharashi I, Evens M (1994) Comparing words, stems, and roots as index terms in an Arabic information retrieval system. *JASIS* 45(8):548–560
13. Allam M (1995) Segmentation versus segmentation-free for recognizing Arabic text. *Proc SPIE* 2422:228–235
14. Argaw AA, Asker L (2007) An Amharic stemmer: reducing words to their citation forms. In: Proceedings of the 5th workshop on important unresolved matters, ACL-2007, Prague, pp 104–110
15. Attar R, Choueka Y, Dershowitz N, Fraenkel AS (1978) KEDMA – linguistic tools for retrieval systems. *J Assoc Comput Mach* 25(1):52–66
16. Baird H (1990) Document image defect models. In: IAPR workshop on syntactic and structural pattern recognition, Murray Hill, pp 38–46
17. Baird H (1993) Document image defects models and their uses. In: Second international conference on document analysis and recognition (ICDAR), Tsukuba City, pp 62–67
18. Beesley K (1996) Arabic finite-state morphological analysis and generation. In: COLING-96, Copenhagen
19. Beesley K, Buckwalter T, Newton S (1989) Two-level finite-state analysis of Arabic morphology. In: Proceedings of the seminar on bilingual computing in Arabic and English, Cambridge
20. Blei D, Ng A, Jordan M (2003) Latent Dirichlet allocation. *J Mach Learn Res* 3(4–5):993–1022
21. Braschler M, Ripplinger B (2004) How effective is stemming and decompounding for German text retrieval? *Inf Retr J* 7(3–4):291–316
22. Brill E, Moore R (2000) An improved error model for noisy channel spelling correction. In: Proceedings of the 38th annual meeting of the association for computational linguistics, ACL'00, Hong Kong, pp 286–293
23. Burges C, Shaked T, Renshaw E, Lazier A, Deeds M, Hamilton N, Hullender G (2005) Learning to rank using gradient descent. In: Proceedings of the 22nd international conference on machine learning, Bonn
24. Burgin B (1992) Variations in relevance judgments and the evaluation of retrieval performance. *Inf Process Manage* 28(5):619–627
25. Carmel D, Maarek YS (1999) Morphological disambiguation for Hebrew search systems. In: NGITS-99, Zikhron-Yaakov
26. Chenm A, Gey F (2002) Building an Arabic stemmer for information retrieval. In: TREC-2002, Gaithersburg
27. Choueka Y (1980) Computerized full-text retrieval systems and research in the humanities: the Responsa project. *Comput Hum* 14:153–169. North-Holland
28. Church K, Gale W (1991) Probability scoring for spelling correction. *Stat Comput* 1:93–103
29. Croft WB, Harding S, Taghva K, Andborsak J (1994) An evaluation of information retrieval accuracy with simulated OCR output. In: Proceedings of the 3rd annual symposium on document analysis and information retrieval, University of Nevada, Las Vegas, pp 115–126
30. Darwish K (2002) Building a shallow morphological analyzer in one day. In: ACL workshop on computational approaches to Semitic languages, Philadelphia
31. Darwish K (2003) Probabilistic methods for searching OCR-degraded Arabic text. Ph.D. thesis, Electrical and Computer Engineering Department, University of Maryland, College Park

32. Darwish K, Ali A (2012) Arabic retrieval revisited: morphological hole filling. In: Proceedings of the 50th annual meeting of the Association for Computational Linguistics: short papers-volume 2, Jeju Island. ACL, pp 218–222
33. Darwish K, Emam O (2005) The effect of blind relevance feedback on a new Arabic OCR degraded text collection. In: International conference on machine intelligence: special session on Arabic document image analysis, Tozeur, 5–7 Nov 2005
34. Darwish K, Magdy W (2007) Error correction vs. query garbling for Arabic OCR document retrieval. ACM Trans Inf Syst (TOIS) 26(1):5
35. Darwish K, Oard DW (2002) Term selection for searching printed Arabic. In: Proceedings of the 25th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR'02), Tampere, pp 261–268
36. Darwish K, Oard D (2002) CLIR experiments at Maryland for TREC 2002: evidence combination for Arabic–English retrieval. In: Text retrieval conference (TREC'02), Gaithersburg
37. Darwish K, Hassan H, Emam O (2005) Examining the effect of improved context sensitive morphology on Arabic information retrieval. In: Proceedings of the ACL workshop on computational approaches to Semitic languages, Ann Arbor, pp 25–30
38. De Roeck A, El-Fares W (2000) A morphologically sensitive clustering algorithm for identifying Arabic roots. In: 38th Annual meeting of the ACL, Hong Kong, pp 199–206
39. Diab M (2009) Second generation tools (AMIRA 2.0): fast and robust tokenization, POS tagging, and Base phrase chunking. In: 2nd international conference on Arabic language resources and tools, Cairo
40. Doermann D (1998) The indexing and retrieval of document images: a survey. Comput Vis Image Underst 70(3):287–298
41. Doermann D, Yao S (1995) Generating synthetic data for text analysis systems. In: Symposium on document analysis and information retrieval, Las Vegas, pp 449–467
42. Domeij R, Hollman J, Kann V (1994) Detection of spelling errors in Swedish not using a Word List en Clair. J Quant Linguist 1:195–201
43. Dumais ST, Furnas GW, Landauer TK, Deerwester S, Harshman R (1988) Using latent semantic analysis to improve access to textual information. In: CHI'88 proceedings of the SIGCHI conference on human factors in computing systems, Washington, DC
44. El-Kholi A, Habash N (2010) Techniques for Arabic morphological detokenization and orthographic denormalization. In: Proceedings of language resources and evaluation conference (LREC), Valletta
45. Fraser A, Xu J, Weischedel R (2002) TREC 2002 cross-lingual retrieval at BBN. In: TREC-2002, Gaithersburg
46. Gao W, Niu C, Nie J-Y, Zhou M, J Hu, Wong K-F, Hon H-W (2007) Cross-lingual query suggestion using query logs of different languages, SIGIR-2007, Amsterdam, pp 463–470
47. Gao W, Niu C, Zhou M, Wong KF (2009) Joint ranking for multilingual web search. In: ECIR 2009, pp 114–125
48. Gao W, Niu C, Nie J-Y, Zhou M, Wong K-F, Hon H-W (2010) Exploiting query logs for cross-lingual query suggestions. ACM Trans Inf Syst 28:1–33
49. Gey F, Oard D (2011) The TREC-2001 cross-language information retrieval track: searching Arabic using English, French or Arabic queries. In: TREC 2001, Gaithersburg, pp 16–23
50. Gillies A, Erlandson E, Trenkle J, Schlosser S (1997) Arabic text recognition system. In: The symposium on document image understanding technology, Annapolis
51. Habash N, Rambow O (2007) Arabic diacritization through full morphological tagging. In: Proceedings of NAACL HLT 2007, Rochester, Companion volume, pp 53–56
52. Han B, Baldwin T (2011) Lexical normalisation of short text messages: makn sens a #twitter. In: Proceedings of the 49th annual meeting of the Association for Computational Linguistics: human language technologies-volume 1, Portland. ACL, pp 368–378
53. Harding S, Croft W, Weir C (1997) Probabilistic retrieval of OCR-degraded text using N-grams. In: European conference on digital libraries, Pisa. Research and advanced technology for digital libraries. Springer, Berlin/Heidelberg, pp 345–359
54. Harman D (1992) Overview of the first Text REtrieval conference, Gaithersburg, TREC-1992

55. Harman D (1995) Overview of the fourth Text REtrieval conference, Gaithersburg, TREC-4, p 1
56. Hassibi K (1994) Machine printed Arabic OCR. In: 22nd AIPR workshop: interdisciplinary computer vision, SPIE Proceedings, Washington, DC
57. Hassibi K (1994) Machine printed Arabic OCR using neural networks. In: 4th international conference on multi-lingual computing, London
58. Hawking D (1996) Document retrieval in OCR-scanned text. In: 6th parallel computing workshop, Kawasaki
59. He D, Oard DW, Wang J, Luo J, Demner-Fushman D, Darwish K, Resnik P, Khudanpur S, Nossal M, Subotin M, Leuski A (2003) Making MIRACLEs: interactive translingual search for Cebuano and Hindi. *ACM Trans Asian Lang Inf Process (TALIP)* 2(3):219–244
60. Hefny A, Darwish K, Alkahky A (2011) Is a query worth translating: ask the users! In: ECIR 2011, Dublin, pp 238–250
61. Hersh WR, Bhuptiraju RT, Ross L, Cohen AM, Kraemer DF, Johnson P (2004) TREC 2004 genomics track overview (TREC-2004), Gaithersburg
62. Hmeidi I, Kanaan G, Evens M (1997) Design and implementation of automatic indexing for information retrieval with Arabic documents. *JASIS* 48(10):867–881
63. Hong T (1995) Degraded text recognition using visual and linguistic context. Ph.D. thesis, Computer Science Department, SUNY Buffalo, Buffalo
64. Huang J, Eftimiadis EN (2009) Analyzing and evaluating query reformulation strategies in web search logs. In: CIKM'09, Hong Kong, 2–6 Nov 2009
65. Jarvelin K, Kekalainen J (2002) Cumulated gain-based evaluation of IR techniques. *ACM Trans Inf Syst* 20(4):422–446
66. Joachims T (2006) Training linear SVMs in linear time. In: Proceedings of the ACM conference on knowledge discovery and data mining (KDD), Philadelphia
67. Jurafsky D, Martin J (2000) Speech and language processing. Prentice Hall, Upper Saddle River
68. Kantor P, Voorhees E (1996) Report on the TREC-5 confusion track. In: TREC-1996, Gaithersburg
69. Kareem Darwish (2013) Arabizi detection and conversion to Arabic. *CoRR* abs/1306.6755
70. Khoja S, Garside R (2001) Automatic tagging of an Arabic corpus using APT. In: The Arabic linguistic symposium (ALS), University of Utah, Salt Lake City
71. Kiraz G (1998) Arabic computation morphology in the west. In: 6th international conference and exhibition on multi-lingual computing, Cambridge
72. Kishida K (2008) Prediction of performance of cross-language information retrieval using automatic evaluation of translation. *Libr Inf Sci Res* 30(2):138–144
73. Kanungo T, Haralick R (1998) An automatic closed-loop methodology for generating character ground-truth for scanned documents. *IEEE Trans Pattern Anal Mach Intell* 21(2):179–183
74. Kanungo T, Haralick R, Phillips I (1993) Global and local document degradation models. In: 2nd international conference on document analysis and recognition (ICDAR'93), Tsukuba City, pp 730–734
75. Kanungo T, Bulbul O, Marton G, Kim D (1997) Arabic OCR systems: state of the art. In: Symposium on document image understanding technology, Annapolis
76. Kanungo T, Marton G, Bulbul O (1999) OmniPage vs. Sakhr: paired model evaluation of two Arabic OCR products. In: SPIE conference on document recognition and retrieval (VI), San Jose
77. Lam-Adesina AM, Jones GJF (2006) Examining and improving the effectiveness of relevance feedback for retrieval of scanned text documents. *Inf Process Manage* 42(3):633–649
78. Larkey LS, Ballesteros L, Connell ME (2002) Improving stemming for Arabic information retrieval: light stemming and co-occurrence analysis. Research and development in information retrieval – SIGIR-2002, Tampere, pp 275–282
79. Lee Y, Papineni K, Roukos S, Emam O, Hassan H (2003) Language model based Arabic word segmentation. In: Proceedings of the 41st annual meeting of the association for computational linguistics, Sapporo, July 2003, pp 399–406

80. Lee CJ, Chen CH, Kao SH, Cheng PJ (2010) To translate or not to translate? In: SIGIR-2010, Geneva
81. Levow GA, Oard DW, Resnik P (2005) Dictionary-based techniques for cross-language information retrieval. *Inf Process Manage J* 41(3):523–547
82. Li Y, Lopresti D, Tomkins A (1997) Validation of document defect models. *IEEE Trans Pattern Anal Mach Intell* 18:99–107
83. Lin WC, Chen HH (2003) Merging mechanisms in multilingual information retrieval. CLEF 2002, LNCS 2785. Springer, Berlin/New York, pp 175–186
84. Liu T-Y (2009) Learning to rank for information retrieval. *Found Trends Inf Retr* 3(3):225–331
85. Lopresti D, Zhou J (1994) Using consensus sequence voting to correct OCR errors. In: IAPR workshop on document analysis systems, Kaiserslautern, pp 191–202
86. Lu Z, Bazzi I, Kornai A, Makhoul J, Natarajan P, Schwartz R (1999) A robust, language-independent OCR system. In: 27th AIPR workshop: advances in computer assisted recognition, Washington, DC. SPIE
87. Maamouri M, Graff D, Bouziri B, Krouna S, Bies A, Kulick S (2010) LDC standard Arabic morphological analyzer (SAMA) version 3.1. Linguistics Data Consortium, Catalog No. LDC2010L01
88. Magdy W, Darwish K (2006) Arabic OCR error correction using character segment correction, language modeling, and shallow morphology. In: Empirical methods in natural language processing (EMNLP'06), Sydney, pp 408–414
89. Magdy W, Darwish K, Rashwan M (2007) Fusion of multiple corrupted transmissions and its effect on information retrieval. In: ESOLE 2007, Cairo
90. Magdy W, Darwish K, El-Saban M (2009) Efficient language-independent retrieval of printed documents without OCR. In: SPIRE 2009, Saariselkä
91. Magdy W, Darwish K, Mourad A (2012) Language processing for Arabic microblog retrieval. In: CIKM, Maui
92. Mayfield J, McNamee P, Costello C, Piatko C, Banerjee A (2001) JHU/APL at TREC 2001: experiments in filtering and in Arabic, video, and web retrieval. In: Text retrieval conference (TREC'01), Gaithersburg
93. McNamee P, Mayfield J (2002) Comparing cross-language query expansion techniques by degrading translation resources. In: SIGIR'02, Tampere
94. Metzler D, Croft WB (2004) Combining the language model and inference network approaches to retrieval. *Inf Process Manage* 40(5):735–750. Special issue on Bayesian Networks and Information Retrieval
95. Mittendorf E, Schäuble P (2000) Information retrieval can cope with many errors. *Inf Retr* 3(3):189–216. Springer, Netherlands
96. Oard D, Dorr B (1996) A survey of multilingual text retrieval. UMIACS, University of Maryland, College Park
97. Oard D, Gey F (2002) The TREC 2002 Arabic/English CLIR track. In: TREC-2002, Gaithersburg
98. Oflazer K (1996) Error-tolerant finite state recognition with applications to morphological analysis and spelling correction. *Comput Linguist* 22(1):73–90
99. Page L (1998) Method for node ranking in a linked database. US patent no. 6285999
100. Pirkola A (1998) The effects of query structure and dictionary setups in dictionary-based cross-language information retrieval. In: SIGIR-1998, Melbourne, pp 55–63
101. Robertson SE, Jones KS (1976) Relevance weighting of search terms. *J Am Soc Inf Sci* 27:129–146
102. Robertson SE, Jones KS (1996) Simple, proven approaches to text-retrieval. Technical report 356, Computer Laboratory, University of Cambridge, Cambridge
103. Robertson SE, Zaragoza H (2009) The probabilistic relevance framework: BM25 and beyond. *Found Trends Inf Retr* 3(4):333–389
104. Salton G, Lesk M (1969) Relevance assessments and retrieval system evaluation. *Inf Storage Retr* 4:343–359

105. Salton G, McGill M (1983) *Introduction to modern information retrieval*. McGraw-Hill, New York
106. Salton G, Wong A, Yang CS (1975) A vector space model for automatic indexing. *Commun ACM* 18(11):613–620
107. Salton G, Fox EA, Wu H (1983) Extended Boolean information retrieval. *Commun ACM* 26(11):1022–1036
108. Sanderson M (1994) Word sense disambiguation and information retrieval. In: SIGIR'94, Dublin, pp 142–151
109. Sanderson M, Joho H (2004) Forming test collections with no system pooling. In: SIGIR'04, Sheffield, 25–29 July 2004
110. Si L, Callan J (2005) CLEF 2005: multilingual retrieval by combining multiple multilingual ranked lists. In: Sixth workshop of the cross-language evaluation forum, CLEF, Vienna
111. Singhal A, Salton G, Buckley C (1996) Length normalization in degraded text collections. In: 5th annual symposium on document analysis and information retrieval, Las Vegas
112. Smith S (1990) An analysis of the effects of data corruption on text retrieval performance. Thinking Machines Corp, Cambridge
113. Soboroff I, Nicholas C, Cahan P (2001) Ranking retrieval systems without relevance judgments. In: SIGIR, New Orleans
114. Szpektor I, Dagan I, Lavie A, Shacham D, Wintner S (2007) Cross lingual and semantic retrieval for cultural heritage appreciation. In: Proceedings of the workshop on language technology for cultural heritage data, Prague
115. Taghva K, Borasack J, Condit A, Gilbreth J (1994) Results and implications of the noisy data projects, 1994. Information Science Research Institute, University of Nevada, Las Vegas
116. Taghva K, Borsack J, Condit A (1994) An expert system for automatically correcting OCR output. In: SPIE-document recognition, San Jose
117. Taghva K, Borasack J, Condit A, Inaparthy P (1995) Querying short OCR'd documents. Information Science Research Institute, University of Nevada, Las Vegas
118. Tillenius M (1996) Efficient generation and ranking of spelling error corrections. NADA technical report TRITA-NA-E9621
119. Tsai MF, Wang YT, Chen HH (2008) A study of learning a merge model for multilingual information retrieval. In: SIGIR, Singapore
120. Tseng Y, Oard DW (2001) Document image retrieval techniques for Chinese. In: Symposium on document image understanding technology (SDIUT), Columbia, pp 151–158
121. Udupa R, Saravanan K, Bakalov A, Bhole A (2009) “They Are Out There, If You Know Where to Look”: mining transliterations of OOV query terms for cross-language information retrieval. In: ECIR, Toulouse. LNCS, vol 5478, pp 437–448
122. Voorhees E (1998) Variations in relevance judgments and the measurement of retrieval effectiveness. In: SIGIR, Melbourne
123. Wang J, Oard DW (2006) Combining bidirectional translation and synonymy for cross-language information retrieval. In: SIGIR, Seattle, pp 202–209
124. Wayne C (1998) Detection & tracking: a case study in corpus creation & evaluation methodologies. Language resources and evaluation conference, Granada
125. Wu D, He D, Ji H, Grishman R (2008) A study of using an out-of-box commercial MT system for query translation in CLIR. In: Workshop on improving non-English web searching, CIKM, Napa Valley
126. Yona S, Wintner S (2008) A finite-state morphological grammar of Hebrew. In: Proceedings of the ACL-2005 workshop on computational approaches to Semitic languages, Ann Arbor, June 2005

# Chapter 11

## Question Answering

**Yassine Benajiba, Paolo Rosso, Lahsen Abouenour, Omar Trigui,  
Karim Bouzoubaa, and Lamia Belguith**

### 11.1 Introduction

The question answering (QA) task has been created to satisfy a specific need of information requested by users who are looking to answer a specific question. The final goal is the ability to automatically parse the available data, extract and validate the potential answers regardless of whether the question is simple, such as:

*“To what family of languages does Hebrew belong?”*

or one that needs a deeper analysis of the data, such as:

*“What political events succeeded the Tunisian revolution?”*

It is important to note that one of the most important features of QA systems is their ability to extract the necessary information from natural language documents. This feature lowers the cost significantly because the creation and update of databases that encompass all the knowledge in a structured fashion has proved to be practically impossible. The use of natural language processing (NLP) techniques does not come at no cost because they:

---

Y. Benajiba (✉)

Thomson Reuters, 3 Times Square, New York, NY, USA

e-mail: [Yassine.benajiba@thomsonreuters.com](mailto:Yassine.benajiba@thomsonreuters.com)

P. Rosso

Pattern Recognition and Human Language Technology (PRHLT) Research Center, Universitat Politècnica de València, Valencia, Spain

e-mail: [pross@prhl.upv.es](mailto:pross@prhl.upv.es)

L. Abouenour • K. Bouzoubaa

Mohamed V-Agdal University, Rabat, Morocco

e-mail: [abouenour@yahoo.fr](mailto:abouenour@yahoo.fr); [karim.bouzoubaa@emi.ac.ma](mailto:karim.bouzoubaa@emi.ac.ma)

O. Trigui • L. Belguith

ANLP Research Group-MIRACL Laboratory, University of Sfax, Sfax, Tunisia

e-mail: [omar.trigui@fsegs.rnu.tn](mailto:omar.trigui@fsegs.rnu.tn); [l.belguith@fsegs.rnu.tn](mailto:l.belguith@fsegs.rnu.tn)

- Come with an intrinsic error penalty as they almost always rely on statistical models and rule-based modules that never perform at 100 %;
- Require a significant amount of training data to build the underlying statistical models; and
- Tend to resort to language-dependant resources and modules which need to be built from scratch for each different language.

In this chapter, we are concerned with the performance of such a system when dealing with a Semitic language. Such a language, as introduced in Chap. 1, exhibits a set of morphological and syntactic properties that need to be taken into consideration and which we will discuss in this chapter. The interest of the NLP research community in QA for Semitic languages is very recent. In CLEF 2012, the QA4MRE task was the first competition to include a Semitic language, i.e. Arabic. Therefore, there is not enough literature on Semitic languages QA per se and we will borrow some insights from other NLP tasks that, we know, can significantly help to understand the intricacies of Semitic languages QA. Thereafter, we will look into QA research works that have been conducted on Arabic to further understand what is required to build a successful QA system for our languages of interest.

In order to achieve this endeavor, we have prepared a chapter that covers almost all the necessary topics to be self-contained. It has been organized as follows. Section 11.2 provides a more detailed description of the generic architecture of a QA system and also takes a deep dive into approaches for answering definition questions and query expansion techniques for QA. Section 11.3 briefly enumerates the most relevant characteristics of the Semitic languages to the QA task. It follows up by a traversal of the relevant NLP tasks to QA and summarizes the relevant literature. Section 11.4 focuses on two specific Arabic QA research works to study some of the techniques for answering definition questions and enriching a query, and reports their results and conclusions. Finally, in Sect. 11.5 we provide a summary to draw the most insightful conclusions.

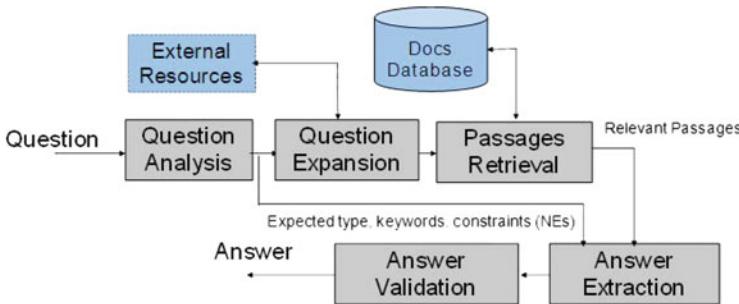
## 11.2 The Question Answering Task

### 11.2.1 Task Definition

As we have previously mentioned the main goal of the QA task is to help users with an urgent need of answering a specific question. However, a more formal definition is needed at this stage to give more details about the task and introduce some of the relevant terminology for the rest of the chapter.

A QA system has two major items at its input (see Fig. 11.1), namely:

1. User questions: that consists in a set of questions expressed in natural language.
2. A collection of documents: these are unstructured documents written in natural language where the answers to the user question can be fetched.



**Fig. 11.1** Generic architecture of a QA system

The system generates at its output the set of relevant answers sorted according to their probability of correctness. The general internal architecture of a QA system is discussed in detail below. However, it is important to note here that this architecture is closely related to the type of questions we want our QA system to be able to process. The type of the question not only is a strong hint for the system to filter out the information found in the document collection, it also defined the type of preprocessing technology required, such as part-of speech (POS) tagging, named entity recognition (NER), coreference resolution (coreference for short), etc., to build a robust system. Consequently, in order to better define the QA task it is necessary to discuss the types of questions involved.

The types of questions that have been enumerated and used in the major evaluation campaigns, i.e. TREC and CLEF, are the following:

1. Factoid questions: generally, the main purpose of the user when she asks a factoid question is to find out about a specific name of a person, place, organization, etc. An example of a factoid question is:

“*What is the capital of Morocco?*”

Or it can be a little bit more complicated, such as:

“*What is the biggest city in Italy?*”

This type of question is considered to be the least complicated because they do not require any type of inference or expensive post-processing.

2. List questions: similar to the previous type of question but the expected answer is a list of items instead of one item. For instance, the following question:

“*What are the countries that have geographic boundaries with Spain?*”

expects for an answer a list of countries. Also the question:

“*Who are the most famous quantum physicists that have been part of the Manhattan project?*”

is a more complicated example because it is probable that the list of physicists' names is scattered over different documents.

3. Definition questions: which will be discussed in more detail in Sect. 11.4.1, are the opposite of factoid questions. Whereas factoid questions provide information

and expect a name, definition questions provide the name and request the QA system to provide information. For instance:

“Who is Moncef Marzouki?”

4. Date and quantity: as the name clearly indicates, the expected answer is a date or a quantity. For instance, the following questions fall under this category:

“When was Richard Feynman born?”

and,

“What age was Mozart when he composed his first symphony?”

5. Other more general types: CLEF and TREC have created a separate set of questions for the most advanced systems. This type of question is kept open and the participating system need to perform more processing and inference than in the previous categories to figure out the right answer. An example of this category of questions would be a question like:

“What were the major consequences of the Arab Spring revolutions over the world economy?”

By defining this set of question types the CLEF and TREC evaluation campaigns have elegantly established different domains of QA expertise where each type of question reveals the performance of the participating systems at specific tasks. In other words, each type of question raises a different type of challenge. In the next subsection we want to take a first glance at these challenges. By doing so we make it very easy to understand afterwards the reason behind choosing a specific architecture for a QA system and at the same time we prepare the ground to explain how exactly building a QA system for a Semitic language requires investigating additional aspects.

### ***11.2.2 The Major Known Challenges***

As we have attempted to show in the previous subsection, understanding the challenges posed by the QA task is key to understanding the contribution of the seminal QA research works (described in Sect. 11.2.3) and the role played by each module in a QA system. The goal of this subsection is to shed light on these challenges. We consider that the major ones are the following:

1. Lack of context: as mentioned in the introduction of this chapter, the input of a QA system is the user’s question and a collection of documents. However, only the former element is relevant for the system to understand the user’s need. These questions are usually short and concise which raises a quintessential challenge to the QA task, i.e. there is not enough material (or context) to restrict the search space or to disambiguate in case some question terms are polysemous. Readers who have approached the information retrieval (IR) task should find this challenge familiar.
2. Language dependency: whether to build a language-specific QA system or not has always been a key decision for most of the researchers in this domain.

The main reason is that this decision drives the technologies that are going to be used to build the system. The language-independent QA systems, from a general viewpoint, tend to use more frequency-based approaches and try to limit, as much as possible, the use of any NLP using language-specific rules or probabilistic models. On the other hand, the language-specific ones, which we are most interested in here, use more NLP tools and models.

3. Required technologies: because a language-specific QA system is itself a federate system built upon a set of sub-modules, its overall performance is strongly proportional to theirs. The challenge is to build such sub-modules in a way that they perform well enough to make an accurate QA system. Achieving this endeavor might be stymied by lack of data and/or adequate techniques. For instance, some of the modules that we frequently see employed in QA systems are the following:

- *Preprocessing tools*: these prepare the ground for the rest of the modules by cleaning the text and make it more “suitable” for the NLP tools to operate on. We have highlighted the word suitable because it is idiosyncratic and it is better understood when the type of data and the nature of the target language are specified. For the Semitic languages, as we will see in more detail in Sect. 11.3, the major preprocessing step consists of word-segmentation, also called tokenization. This consists of breaking each word into the morphemes composing it in order to decrease the sparseness of the text.
- *Information extraction (IE) tools*: their main role is to highlight specific information, relevant to the QA system, in the text to be used when listing the potential answers. The named entity recognition (NER) systems are amongst the most used IE tools in QA. However, the most advanced systems tend to use also coreference and relation extraction systems to overcome the problems encountered when different parts of the goal information are dispersed in the text.
- *Semantic resources*: in order to perform query expansion (see Sect. 11.2.3) it is recommended to use semantic resources such as WordNet. In the case of Semitic languages these resources are either absent for some of them or, for others, require a significant enrichment to reach a level where they can be used for successful results (see Sect. 11.4.2).

Other challenges may arise when we get into the details. However, what we believe is important to point out here is that even though we cannot find enough literature to show the challenging aspects of building a QA system oriented to a Semitic language, looking at the published work investigating the NLP sub-fields which we have enumerated is relevant to understand the difficulty of the QA task.

### 11.2.3 *The General Architecture of a QA System*

According to the literature, a standard QA system is composed of at least three modules: question analysis, document or passage retrieval, and answer extraction.

Further modules for query expansion, to enrich the original question with related terms, and answer validation, to validate the answer before returning, can be added in the QA pipeline of Fig. 11.1.

Given as input the question, typed or via a voice interface [55], the question analysis module determines the type of the given question (question classification), in order to pass the information about the expected type of answer to the answer extraction module, together with the question keywords (used by the passage retrieval module as a query) and the named entities appearing in the question (which are very essential to validate the candidate answers). For instance, if the question is “How many inhabitants were in Rabat in 1956?” the expected answer is a quantity (the target of the question is the number of inhabitants) but this value has to accomplish the contextual constraints of Rabat (i.e., we are not interested in the number of inhabitants of another town) and of 1956 (i.e., we are not interested in any other date). Question analysis is probably the most crucial step of a QA system, since, in the answer extraction module, the extraction strategy of the answer completely depends on the correct classification of the question: a proper classification of the question will allow to restrict the candidate answers to be considered. Moldovan et al. report in [45] that more than 36 % of errors in QA are due to mistakes of question classification. The approaches for question classification can be pattern-based or machine learning-based and generally use a QA typology made up of question types. Most of these approaches use named entities (NEs) tagging for question (and answer) typing, typically classifying a question as requiring a particular NE type as the answer. Therefore, the recognition of NEs is crucial throughout the QA process, from question analysis to answer extraction, and even more for a morphologically-rich language such as Arabic [10] and [11].

Document or passage retrieval is the core module of QA systems and it retrieves the passages of documents that are relevant to contain the answer. Usually document retrieval systems supply a set of ranked documents based on a distance function between the question and the documents, and use classical weighting schemes that are based on term frequency, such as tf-idf, or on statistical analysis, such as BM25 [52]. Most QA systems are based on IR methods that have been adapted to work on passages instead of the whole document. The main problem with these QA systems is that they are the adaptations of classical document retrieval systems that are not specifically oriented to the QA task, and often fail to find documents containing the answer when presented with natural language questions.

The passage retrieval (PR) module returns instead pieces of text (passages) that are relevant to the user questions instead of returning a ranked list of documents like the IR systems do. In fact, it is important to consider the differences between traditional document retrieval and QA-oriented passage retrieval: in the first case, the greatest effort is made to retrieve documents about the topic of the query, while in the second case, the aim is to retrieve pieces of text that contain the answer to a given question. Various methods have been proposed to determine the similarity between the passage and the question [59].

Also, it is interesting to mention that in the PR module the NE recognition is very important. Chu-Carroll et al., in [18] and [17], investigate the impact that

NE recognition may have on document and/or passage results. In order to study the significant degradation in search performance with imperfect NEs, the authors applied different error models to the gold standard annotations in order to simulate errors made by automatic recognizers. A significant document retrieval gain was achieved when adopting NE recognizers (15.7% improvement in precision).

As stated by Abney et al. in [1] and Clarke et al. in [19], the good performance of the document or PR module is a critical aspect for the answer extraction module that is responsible for extracting the final answer from the retrieved passages or documents. Every piece of information extracted during the previous step (e.g. the words of the contextual constraints of the question) is important in order to determine the right answer. The answer is searched for within the retrieved passages, using the information obtained from the question analysis about the focus and the target of the question that need to be extracted by the answer extraction module among the list of candidate answers. Finally the answer extraction module extracts a list of candidate answers from the relevant passages and returns the one that it is most confident about. In order to estimate for each of the candidate answers the probability of correctness and to rank them from the most to the least probable correct one, some QA systems have a final answer validation module in the pipeline of Fig. 11.1 [53].

In this section we described the main characteristics of a generic architecture of a monolingual QA system whose aim is mainly answering factoid questions on objective facts or definitions. In case of cross-lingual QA a module to deal with the translation of the question has to be added at the beginning of the pipeline of Fig. 11.1. Moreover, when we want to process more difficult questions, such as list questions, further modules need to be included to both add another layer of information, i.e. anaphora resolution, and deal with the high level of noise usually encountered in microblogs and tweets.

#### ***11.2.4 Answering Definition Questions and Query Expansion Techniques***

Initial researchers in the definition question answering area have proposed the use of models such as tf-idf, bag of words and centroid vector based on a numeric approach for the identification of answers [66], and on surface patterns based on a symbolic approach for the extraction of answers [28, 51]. Other methods have been proposed based on the combination of numeric and symbolic approaches. These methods are frequently based on machine learning algorithms (i.e., decision trees, Hidden Markov Models) [15, 20, 26, 65], linguistic approaches involving deeper linguistic tools and resources (i.e., ontologies, syntactic and semantic analyzers, named entity recognizer) [38–40]. Moreover, Web knowledge mining methods have been investigated [67]. Cui et al. have proposed in [21] a soft matching method for answer extraction that is based on machine learning, web mining and a probabilistic approach. Moldovan et al., in [46], have proposed to transform

questions and answer passages into logic representations and world knowledge axioms as well as linguistic axioms to supply the logical prover, in order to render a deep understanding of the relationship between questions, passages and answers. Also, Denicia-Carral et al. have proposed in [23] a method for answering definition questions that is based on the use of surface text patterns, a sequence-mining algorithm, and a knowledge-based approach.

Other researchers, such as Miliaraki and Androutsopoulos in [44], have used a mining external definitions method to rank the definitions with machine learning approaches. Their methods consist in calculating the similarity between candidate answers and definitions from external Web knowledge resources such as Wikipedia and biography dictionaries. Harabagiu et al. in [56] and Moldovan et al. in [46] have both proposed to identify answers by combining extraction techniques with abductive reasoning. Blair-Goldensohn et al. [14] have proposed to combine their knowledge-based method with a statistical approach to answer definition question. Finally, Tanev et al. have proposed in [58] to combine linguistic processing with a Web knowledge mining method.

The above research works have given a new advance to answering definition questions not only on the basis of symbolic and numeric approaches but also with hybrid approaches. In order to address the problem of answering definition question in Arabic, in Sect. 11.4 we present DefArabicQA, a QA system based on both linguistic and frequency-based approaches. DefArabicQA uses a surface patterns technique similar to the one developed by Ravichandran and Hovy in [51].

With respect to query expansion (QE), many techniques have been investigated by researchers in the IR field. The basic ones are those targeting to fix spelling errors by searching for the corrected form of the words, such as the work presented by Pinto, in [50]. Other QE processes rely on morphological relations and reformulate the user query by adding the different variations that are generated from a keywords stem [42]. Although this QE technique produces higher recall [12, 47] it is difficult to assert that it improves the precision. This is why researchers have investigated other QE techniques such as those using semantic relations. Generally, a semantic QE process is performed by considering the synonyms of the query keywords. A thesaurus can be used as a base for such a process [48]. However, the use of a thesaurus, which is generally built on the basis of statistical techniques, presents many disadvantages. Indeed, building a thesaurus is a time-consuming task since a great amount of data has to be processed. Moreover, the precision of thesaurus-based QE in term of semantic distance has to be proved.

In the work of Bilotti et al. presented in [13], a special focus was put on QA. Indeed, in that work, the authors quantitatively compare two different approaches to handling term variation: applying a stemming algorithm at indexing time, and performing morphological query expansion at retrieval time. Conducted experiments show that morphological expansion yields higher recall.

In Sect. 11.4 we describe our QE approach for Arabic QA. The approach uses both morphological variations as in [12, 13] and semantic relations. In comparison with the work of Bilotti et al. in [13], our approach expands at retrieval time. It aims at reaching higher recall by considering semantically related terms extracted from

the Arabic WordNet. This technique has also been used in other works, such as Hsu in [36] and Gong et al. in [33]. With respect to the latter works, our approach is different in:

1. Considering many levels in the hyponymy hierarchy of the Arabic WordNet;
2. Taking into account relations with the super merged ontology (SUMO);
3. Improving precision in the context of QA using a distance density N-gram PR module;
4. Not considering terms weighting.

### ***11.2.5 How to Benchmark QA System Performance: Evaluation Measure for QA***

In order to be able to compare the overall performance of a QA system and its sub-modules, different measures have been defined. In this subsection we want to enumerate the most used ones and describe their formulas.

**F-measure:** this measure is used in QA especially to evaluate the performance of the PR module. It combines precision and recall in one measure. Precision is an indicator of how many of the retrieved items are relevant. Thus precision can be expressed as follows:

$$\text{Precision} = \frac{\text{Number of retrieved relevant items}}{\text{Number of retrieved items}}$$

Recall, on the other hand, indicates how many of all the relevant items have been retrieved and can be expressed as follows:

$$\text{Recall} = \frac{\text{Number of retrieved relevant items}}{\text{Number of relevant items}}$$

F-measure combined both in the following formula:

$$\text{F-measure} = 2 \cdot \text{Precision} \cdot \text{recall} / (\text{Precision} + \text{recall})$$

**Accuracy:** this measure is used to evaluate the quality of the overall QA system that provides one potential answer. Accuracy is a number between 0 and 1 that indicates the probability of the QA system to provide the correct answer on average. It is expressed as following:

$$\text{Accuracy} = \frac{\text{Number of correct answers}}{\text{Number of questions}}$$

**Mean reciprocal rank (MRR):** this measure is used to evaluate the quality of an overall QA system that provides a sorted list of  $n$  potential answers. MRR is a number between 0 and 1 that indicates how many times the QA system ranks the correct answer as first. The formula to compute MRR is:

$$MRR = (1/|Q|) \cdot \sum_{i=0}^{|Q|} (1/rank(i)) \quad (11.1)$$

**Answered questions (AQ):** is another measure for QA systems providing a sorted list of  $n$  potential answers. AQ is a number between 0 and 1 that indicates the probability of the QA system providing a correct answer in its sorted list of  $n$  potential answers. AQ is expressed as follows:

$$AQ = \frac{\text{Number of answer lists that contain the correct answer}}{\text{Number of questions}}$$

This measure represents a relaxed version of both the accuracy and the MRR. That is because it does not penalize the system when it does not rank the correct answer as first.

## 11.3 The Case of Semitic Languages

### 11.3.1 NLP for Semitic Languages

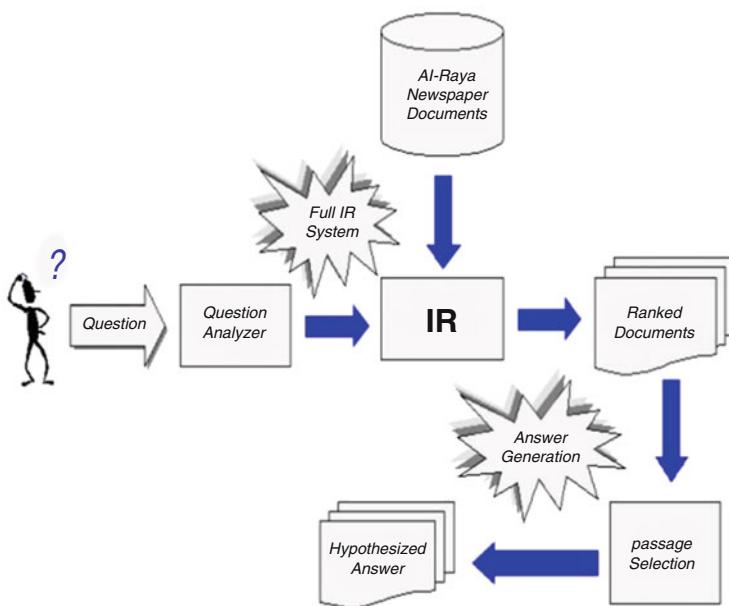
In the first chapter of this book, the authors give a detailed characterization of the Semitic languages. In this section, however, we want to merely recall two characteristics that have been central to almost all NLP tasks, namely:

1. Lack of short vowels: in their classical form, all the Semitic languages use two types of vowels. The long ones are represented as letters in the scripture and the short ones as diacritics. In the modern Semitic languages, however, the diacritics are omitted, primarily in all news-wire texts, because the reader can still mentally add the diacritics while reading the text. The machines on the other hand are confronted with an increase in ambiguity because when the short vowels are, completely or partially, omitted new homographs are created.
2. Agglutinating word forming strategy: the Semitic languages use both derivation and inflection to form the words. Consequently, Semitic languages exhibit a high morpheme-per-word ratio that results in a sparseness of data.

According to the literature, building a robust NLP system for any Semitic language means accounting for these characteristics and their implications.

Another challenging aspect of building efficient NLP systems oriented to a Semitic language is the availability of resources and annotated data. Across the board, publications show that some Semitic languages have access to more resources than others.

For instance, Itai and Wintner in [37] report Hebrew NLP resources that are available both for research and commercial purposes. These resources include POS-tagging, morphological disambiguation and other lexicons. For Arabic, we find among the most used corpora the Arabic TreeBank [43], the Arabic WordNet [25] and the data used for the automatic content extraction (ACE) competition [24]. Also, an Amharic POS-tagging corpus is described by Demeke and Getachew in [22].



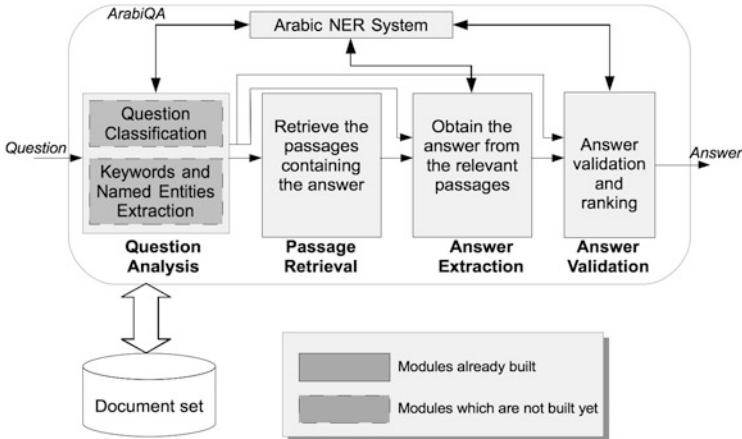
**Fig. 11.2** Architecture of the QARAB system as illustrated in the original paper

### 11.3.2 QA for Semitic Languages

In our description of the major challenges of the QA task in this chapter (see Sect. 11.2.2) we have mentioned how, due to the nature of the task, QA systems are federate systems relying on many components. Building these components requires a set of resources and annotated data which might not be available for the Semitic languages. As we show in Sect. 11.2.1, it is still possible to employ language-independent techniques to successfully build a QA system for a specific type of questions. However, that does not deny the fact that one of the factors that discourages conducting more advanced research in QA for our languages of interest is the scarcity of resources.

In the literature, the only published QA research works for Semitic languages report studies using only Arabic (including the ones we report in Sect. 11.4). Those works are the following:

Hammou et al. report in [35] their work to build a QA system, QARAB, fully oriented to the Arabic language. The architecture of QARAB is illustrated in Fig. 11.2. As reported in their paper, their system relies on a tokenizer and POS-tagger to preprocess the documents. This is necessary when dealing with any Semitic language since the data tends to be very sparse in its raw form as mentioned



**Fig. 11.3** ArabiQA system architecture

in Sect. 11.3.1. After retrieving the candidate sentences that may contain the answer using an IR system, they are processed using a set of keywords to find the answer.

The test-bed of the second QA system was composed of a set of articles from the Raya newspaper. In the evaluation process four Arabic native speakers were asked to give the system 113 questions and judge manually the correctness of its answers. The reported results of precision and recall are of 97.3 %. These (possibly biased) results seem to be very high if compared with those obtained before for other languages (cf. Sect. 11.2.3). Unfortunately, the test-bed that was used by the authors is not publicly available to be used for comparison.

Benajiba et al., in [10], have followed CLEF guidelines to build an evaluation corpus. Their research study, however, targeted factoid questions only. The resulting QA system, ArabiQA (cf. Fig. 11.3), employs JIRS for passage retrieval. The same authors showed in a previous publication [9] that JIRS performs significantly better when the documents are preprocessed with a stemmer first.

In their paper, the authors report that an answer extraction (AE) module reached an accuracy of 83.3 %. This AE system relies heavily on their NER system.

Other works that have significantly contributed to the Arabic QA research are Abouenour et al. in [5] and Trigui et al. in [62]. In Sect. 11.4 we give a detailed description of their main contributions.

For Semitic languages other than Arabic, we did not find any published works on QA or any field closely related to it. However, we believe that most of the works done on Arabic report significant insights can be easily extrapolated to the other Semitic languages because of the characteristics they share.

In the next section we describe in detail two Arabic QA systems in which we invite the reader to see more details of the intricacies involved in building a QA system for a Semitic language.

## 11.4 Building Arabic QA Specific Modules

### 11.4.1 Answering Definition Questions in Arabic

In this section, we detail the approach and the architecture of an Arabic definition QA system entitled ‘DefArabicQA’ [61].

As mentioned in Sect. 11.1, definition questions are questions asking about interesting information concepts, organizations or persons such as “*What is Subliminal Stimuli?*”, “*What is the Muslim Brotherhood?*” or “*Who is Tawakel Karman?*” Figure 11.4 illustrates an example of an Arabic definition question with its expected answer.

This section describes DefArabicQA, an open-domain definition question answering system for Arabic language. This system is based on a hybrid approach constituted of superficial linguistic and statistical methods. The statistical methods are based on external knowledge, Web mining for QA, frequency and bag-of-words ranking. These methods are used mainly for both passage retrieval and answer selection modules. The superficial linguistic methods are based essentially on lexical pattern, lexical constraint pattern and lexical analyzer. These methods are used in both question analysis and answer extraction modules.

#### DefArabicQA System

To perform question answering, as discussed in Sect. 11.2, questions are first analyzed and a list of predictions is made such as the type of answer expected by the user. Secondly, a search of a list of relevant documents is performed. And then, the top relevant passages that probably contain the expected answer are identified. A list of candidate answers is extracted from these passages. Finally, the top selected candidate answers are reported.

The architecture of the DefArabicQA system consists of four modules which can be described as follows: questions analysis, passage retrieval, answers extraction and answer selection (Fig. 11.5).

#### Question Analysis

The first module of DefArabicQA is question analysis. A set of lexical question patterns built by an expert are used in this module to identify the key characteristics of the question which are: question class, expected answer type, and question topic. These patterns are illustrated in Fig. 11.6. They cover the most standard linguistic styles used to ask a definition question in Arabic language about a concept, an organization, or a person. These question patterns cover respectively in English the typical form of definition questions started by the interrogative pronouns: WHO and WHAT used in the data set of the conference TREC [64].

السؤال التعرفي	من هم الاخوان المسلمين؟
الاجابة التعريفية	الاخوان المسلمين هي جماعة إسلامية، تصف نفسها بأنها «اصلاحية شاملة». تعتبر اكبر حركة معارضة سياسية في كثير من الدول العربية، خاصة في مصر، أتها حسن البنا في مصر في مارس عام ١٩٢٨ مكررة اسلامية وسرعان ما انتشر فكر هذه الجماعة، فنشأت جماعات اخرى تحمل فكر الاخوان في العديد من الدول، ووصلت الان الى ٢٢ دولة. انضم كل الدول العربية ودول اسلامية وغير اسلامية في القارات السنت.
The definition question	What is the Muslim Brotherhood?
The definition answer	The Muslim Brotherhood, is the world's most influential and one of the largest Islamist movements, and is the largest political opposition organization in many Arab states. Founded in Egypt in 1928 as a Pan-Islamic, religious, political, and social movement by the Islamic scholar and schoolteacher Hassan al-Banna. <sup>2</sup>

Fig. 11.4 An Arabic definition question and its expected answer (with its English version)

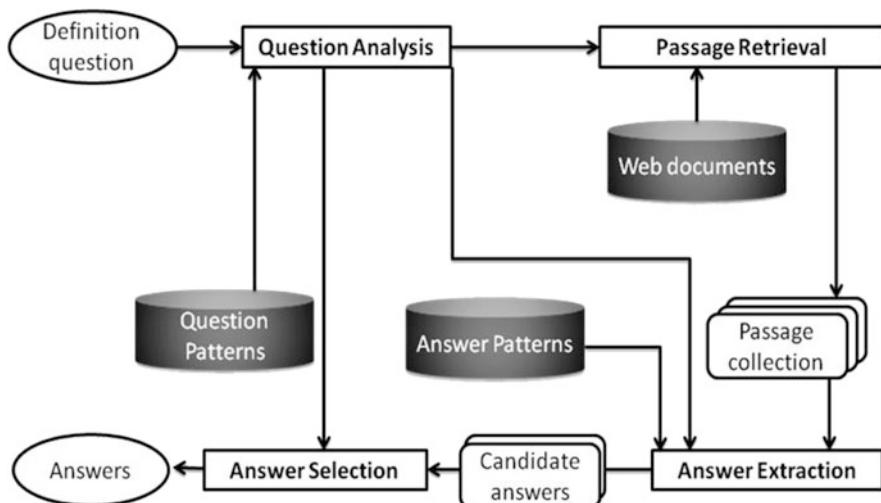


Fig. 11.5 DefArabicQA system architecture

Definition question patterns	Expected answer types
? + من هو/هي «الموضوع» ?	Who+be+ a question topic +? An interesting information about a person
? + من هم «الموضوع» ?	Who+be+ a question topic +? An interesting information about a group of persons with common criteria or an organization
? ما هو/هي «الموضوع» +	What+be+ a question topic +? An interesting information about an organization or a concept

Fig. 11.6 A list of the Arabic definition lexical question patterns used by DefArabicQA with their English version

The result of the question analysis module for the definition question: “ما هي الاخوان المسلمين؟” (What is the Muslim Brotherhood?), shown in Fig. 11.4, is as follows: the question class is “definition”, the expected answer type is “a definition of a group of persons sharing common criteria or a definition of an

organization”, the question topic is “الإخوان المسلمين” (“Muslim Brotherhood”), and the interrogative pronoun is “ما هي” (“who is the”).

The key characteristics of the question are essential for the process of the passage retrieval, answer extraction and answer selection modules.

## Passage Retrieval

This is the second module of DefArabicQA. In this module, the search of the top relevant passages is performed to look for pertinent passages from structured and unstructured Web documents. The key characteristics of the question (i.e., the question class, the question topic and the interrogative pronoun) are used to build the respective search queries. The structured Web documents are collected from the online encyclopedia “Wikipedia”, while the unstructured Web documents are collected from the result of the Arabic version of the Google search engine. All these Web documents are collected according to the respective question search queries.

The search results of the passages retrieved are normalized by replacing some Arabic characters by standard ones (e.g. the Arabic letter “ج” is normalized to “ج”). Then, a list of constraint rules are applied to verify the minimum length of passages and their language, and the existence of the totality of the words of the question topic when it is composed of multi-words. The passages satisfying these constraint rules are kept and used as input to the answer extraction module.

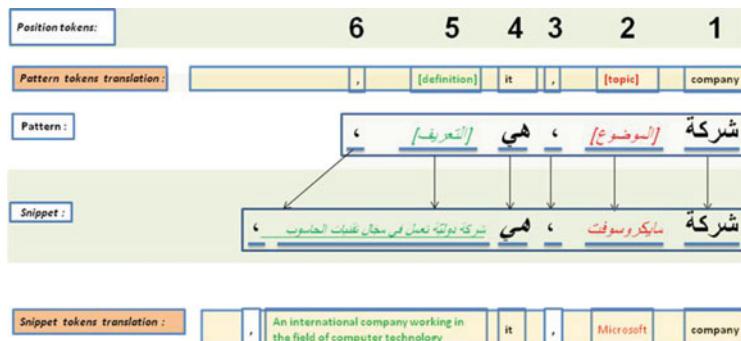
## Answer Extraction

This is the third module of DefArabicQA. This module is based on statistical and surface pattern approaches. A set of 100 definition lexical patterns are built semi-automatically following a method inspired by the work reported by Ravichandran and Hovy in [51]. These lexical answer patterns are built to extract precise definition information (answers) from the collected passages. These patterns are built to deal with various linguistic styles used in Arabic documents to introduce definition information. The method used to construct these patterns can be described in four steps: (i) a set of seeds (i.e., a seed is constituted of a named entity and its definition) are collected from an annotated corpus, (ii) a set of passages containing both elements of the respective seed are retrieved, (iii) the lexical answer patterns are generated from these passages by replacing the seed elements (i.e., named entity and its respective definition) by these tags: [named entity] and [definition], (iv) an Arabic expert verifies these lexical answer patterns by removing the tokens that are not useful and attributing a pertinence score to each pattern. Figure 11.7 illustrates these steps with an example.

Therefore, the alignment process based on the soft matching method can deal with non-exceptioned tokens in a passage [62]. Figures 11.8 and 11.9 (translation provided in Figure 11.10) illustrates an example of this variation language (i.e., the

Example of a seed: Named entity/definition	السيليمinal / البرجة اللاشعورية أو تحت الشعورية باختصار شديد Subliminal stimuli/ are any sensory stimuli below an individual's threshold for conscious perception
Example of a passage containing the elements of the respective seed	بناء على طلبك في التعريف بهذا الموضوع السيليمinal هو البرجة اللاشعورية أو تحت الشعورية باختصار شديد Based on your request in the definition of the subject Subliminal stimuli/ are any sensory stimuli below an individual's threshold for conscious perception
Definition lexical answer pattern generated	[Definition] على طلبك في التعريف بهذا الموضوع [Named Entity] هو [Topic] Based on your request in the definition of the subject [Named entity] are [Definition].

**Fig. 11.7** Example of the process of the main steps to build a lexical answer pattern



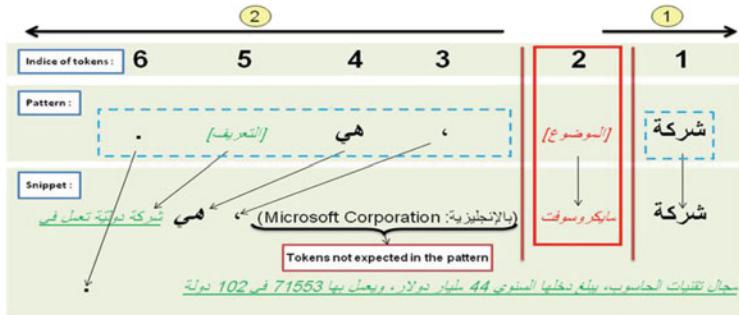
**Fig. 11.8** An example of a successful alignment process based on hard-matching method

tokens “Microsoft Corporation”: are not expected by the pattern at position 3) and how the alignment process has succeeded in dealing with it.

The soft-matching method clearly helps when the correct information (the expected answer) is not frequent in the retrieved passages, because in case of non-redundancy, any missed information means probably an unanswered question. The flexibility criterion of the alignment process based on soft-matching method resolves this problem and the infrequent information is identified. Despite this advantage, the negative consequence of using the soft-matching approach in the alignment process is the identification of noisy information as candidate answers in some case. This drawback has its effect on the last answer selection module of DefArabicQA cases and, therefore, on the overall results [62].

## Answer Selection

The fourth module of DefArabicQA employs a statistical approach. The candidate answers identified by the answer extraction module are classified as appropriate and inappropriate candidate answers. The criterion used for this classification is based on the hypothesis that information is correct if it is redundant. A candidate



**Fig. 11.9** An example of a successful alignment process based on soft-matching method

Token (in English)	Token (in Arabic)
Company	شركة
Topic	الموضوع
Microsoft	مايكروسوفت
She	هي
(In English: Microsoft Corporation)	(بالإنجليزية: Microsoft Corporation)
Definition	التعریف
an international company working in the field of computer technology, with an annual income of 44 billion \$, and employs 71,553 in 102 countries	شركة دولية تعمل في مجال تقنيات الحاسوب، يبلغ دخلها السنوي ٤٤ مليار دولار، ويعمل بها ٧١٥٥٣ في ١٠٢ دولة.

**Fig. 11.10** Tokens of the examples in Figs. 11.8 and 11.9 translated from Arabic to English

answer is classified as appropriate if it is constituted by frequency clauses above a given threshold, otherwise it is classified as inappropriate. The duplicate and near duplicate candidate answers that are classified as appropriate are moved to the inappropriate ones. Only the appropriate candidate answers that are different are kept for the ranking step.

The candidate answers classified as appropriate are ranked according to criteria related to special and global features of the candidate answers. As described by Trigui et al. [60], these criteria are: pattern weight criterion, snippet position criterion, frequent word criterion, and lexical coverage criterion. The top ranked candidate answers are selected to be reported.

## Implementation

The lightweight definition of the QA system for DefArabicQA is developed by the Anlp-RG research group.<sup>1</sup> It is based on tools which are not costly in complexity and

<sup>1</sup><https://sites.google.com/site/anlprg/>



**Fig. 11.11** An example of an output of DefArabicQA according to an Arabic definition question

**Table 11.1** Results according to the answer expected type

Nature of the expected answers for the questions	Correct	False	Total
A definition answer	79 (89 %)	10 (11 %)	89 (100 %)
A “Nil” answer	20 (51 %)	19 (49 %)	39 (100 %)

time such as Alkhalil morpho sys, reported by Boudlal et al. in [16]. It uses the Web resource as a corpus. A list of patterns are used both in the question analysis and the answer extraction modules. These patterns are three definition lexical question patterns that were built manually by an expert and 100 definition lexical answer patterns built semi-automatically using 230 seeds.

Figure 11.11 shows an example of the main interface of DefArabicQA. The question asked is “من هي توكل كرمان؟” (in English “Who is Tawakel Karman?”). The returned answers by the DefArabicQA are annotated by an Arabic expert as appropriate answers.

## Evaluation

DefArabicQA was evaluated on 128 Arabic definition questions. These questions were manually annotated based on whether or not they have answers. As result of the annotation, 89 questions were considered as definition questions and 39 were not (i.e., ‘Nil’ answer expected). The overall results are shown in Table 11.1. The results are divided according to the annotation of the questions: question expecting a

definition answer or not. Eighty-nine percent of the questions expecting a definition answer have obtained correct answers. While 51 % of the questions expecting no answer have obtained correctly ‘Nil’. The overall accuracy for these 128 definition questions is 0.77, while MRR is equal to 0.74. This score indicates the performance of DefArabicQA.

### ***11.4.2 Query Expansion for Arabic QA***

As mentioned in Sect. 11.2.3, the Passage Retrieval module plays a key role in a QA system. So the enhancement of such a module is an important subject for QA research. One possibility is integrating a Query Expansion module on top of PR module. Let us recall that a Query Expansion process consists in adding new terms to the bag-of-words in the user query and generating new queries that are used at the document or passage retrieval stage. Another possibility is using N-gram models in order to improve the quality of passage ranking that is provided the PR module.

The current section focuses on how QE and N-gram models can be developed and combined in the context of QA especially for Arabic as one of the Semitic languages, and discusses the required linguistic resources and datasets as well as the impact on performances measures.

Recently, Abouenour et al. proposed in [3] an approach that aims at enhancing performance of PR leveraging semantic QE. This work is not only concerned with the development of modules but also by resource enrichment, applying and evaluating statistical approaches to Arabic QA through well-known measures and an evaluation dataset that was built in the framework of this work.

Actually, in this approach, a question, after being analyzed (keywords extraction, stopwords removal, question classification, etc.), is processed through two levels: keyword-based level and structure-based level. The former integrates a semantic QE module that generates semantically related terms for each question keyword. The latter applies a PR system that is based on the Distance Density N-gram Model and is adapted to the Arabic language. Both levels aim at enhancing the quality of passage ranking.

The proposed approach was tested by the same authors (see [6]) using a dataset which is composed of: (i) 2,264 TREC and CLEF questions<sup>2</sup>(with their answers) translated into Arabic and covering different question types and domains (see Table 11.2 and Fig. 11.12); (ii) a document collection which is automatically built from Web snippets returned by the Yahoo API.<sup>3</sup> These snippets correspond to queries generated from the 2,264 questions of the dataset. On average, each question is concerned by 8 queries and each query provides 30 snippets; so, the collection

---

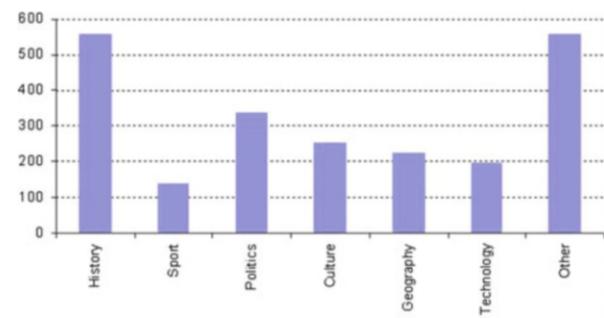
<sup>2</sup>Can be downloaded from: <http://sibawayh.emi.ac.ma/ArabicQA/resources/TrecClefQuestionsArabic.xls>

<sup>3</sup><http://developer.yahoo.com/>

**Table 11.2** TREC and CLEF questions distribution over categories

Type	CLEF	TREC	Collection
Abbreviation	34	133	167
Count	89	106	195
Location	123	307	430
Measure	16	56	72
Object	26	29	55
Organization	63	57	120
Other	120	340	460
Person	194	258	452
Time	93	208	301
List	6	6	12
Overall	764	1,500	2,264

**Fig. 11.12** TREC and CLEF question distribution over domains



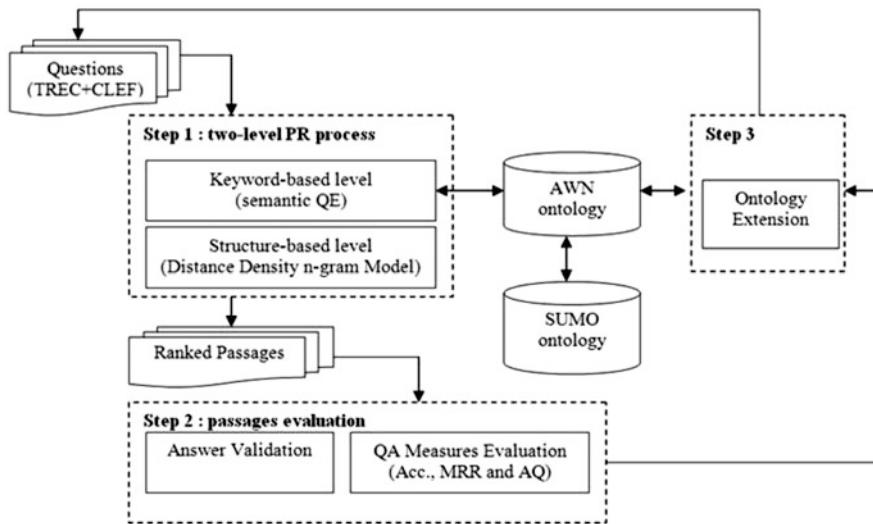
contains around 556,816 documents (a document per snippet). These documents have been converted into the SGML format that is used by the PR system integrated in the structure-based level.

The objective behind building and using this dataset is multi-fold. Firstly, it fills in a gap in the evaluation material for Arabic QA. In fact, the QA4MRE task organized on behalf of CLEF 2012 was among the rare campaigns that provided a test-set and collections that can be used in evaluation of Arabic QA systems. Secondly, the dataset of 2,264 questions can present complexity that is similar to that of CLEF and TREC in terms of number and structure. Thirdly, since the main objective of the evaluation is comparing results before and after using the proposed approach, we can afford to build the document collection from question queries. Finally, using CLEF and TREC questions as evaluation test beds gives us an idea about the difference in Arabic QA performances with respect to different levels of complexity such as question culture (European and American respectively) and structure (short and long respectively).

The experimental process followed the cycle shown in Fig. 11.13.

This experimental process integrates three main steps:

- Step 1: concerns the PR process which implements the previously mentioned two levels.



**Fig. 11.13** Experimental process for the Arabic QA approach evaluation

- Step 2: the passages resulting from the PR process (composed of the two levels) are considered relevant if they contain the expected answer. The answer validation task allows evaluating the process on the basis of three well-known measures namely accuracy, mean reciprocal rank and number of answered questions.
- Step 3: after analysis of the questions for which our process fails to retrieve passages containing the right answer, the ontology used as a resource is extended and the PR process is refined in order to overcome the identified weaknesses and, therefore, increasing passage relevance.

The evaluation of the proposed approach has a two-fold objective: (i) testing the usefulness of this approach in a challenging context such as the Web, and (ii) considering a large number of open-domain questions that have a significant representativeness of users' needs.

In the following subsections, we begin by giving an overview of the semantic QE process integrated in the keyword-based level. After that, we describe the use of the Java information retrieval system<sup>4</sup> (JIRS) as an implementation of the Distance Density N-gram Model adopted at the structure-based level. We show by example how this system can decrease the negative effect of the high recall generated by QE. Finally in this section, the obtained results as well as their analysis are presented and discussed.

<sup>4</sup><http://sourceforge.net/projects/jirs>

## Semantic Query Expansion Based on Arabic WordNet

### Overview of the QE Process

The first objective of a QE process is reformulating the user query in such a way as to consider other related terms. At the retrieval stage, these new terms may help in forming new queries that can reveal documents with high relevance for the user even if they do not contain any term of the original query. Generally, light stemming is applied to each keyword in order to enrich the query by terms sharing the same stem. In the QA task, the meaning of the user question has to be maintained since the finality of the system is providing a precise answer rather than listing related documents. Hence, a semantic QE process is more suitable in this case. Such a process consists in considering also terms that are nearer in terms of meaning even if they do not share the same stem. The semantic QE process proposed by Abouenour et al. in [3] is based on the content and semantic relations in the Arabic WordNet (AWN) ontology [25] which presents the following advantages:

- The AWN ontology is a free resource for modern standard Arabic (MSA).
- AWN is based on the design and the content of Princeton WordNet (PWN) described by Fellbaum in [27] and other WordNets. These WNs have been used in different NLP fields and have proven significantly useful in many applications;
- AWN has a structure which is similar to WordNets existing for approximately 40 languages. Therefore, cross-language processes could be considered later as an enhancement of the present work.
- AWN is also connected to the suggested upper merged ontology (SUMO) [49]. Let us recall briefly that SUMO is an upper level ontology which provides definitions for general-purpose terms and acts as a foundation for more specific domain ontologies. The connection of AWN with SUMO allows these formal definitions to be used.

The proposed QE process uses four of the semantic relations connecting AWN synsets. Let us recall that in WordNet, synonyms are grouped into synsets. Therefore, in addition to the stem-based QE, a question keyword is expanded by its hyponyms, hypernyms, synonyms and related SUMO concepts. The first three relations have been used in QE for QA for other languages such as English [34]. The current work proposes for Arabic an approach which is inspired by what was done for English with two major contributions:

- Considering also SUMO concepts that are connected with AWN synsets;
- Performing a recursive QE process where terms that are generated from the four relations (i.e., synonymy, hypernymy, hyponymy and SUMO) are again inputs for the same process. Figure 11.14 illustrates an example where the four kinds of QE have been applied [5].

Figure 11.14 illustrates the process applied in the keyword-based level. Two stages are depicted: the preprocessing stage (Node with label 1) and the keywords expansion stage (Node with label 2). Each stage is composed of elementary tasks.

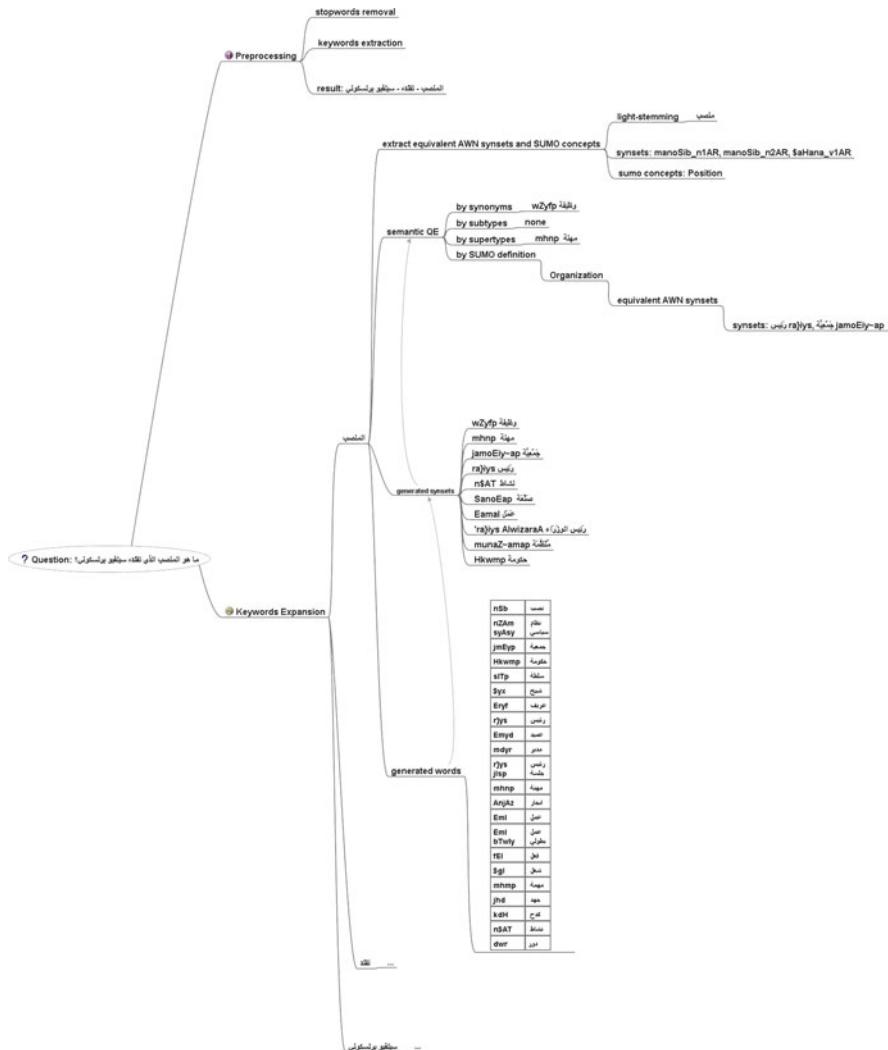


Fig. 11.14 Example using the proposed semantic QE

These elementary tasks are illustrated by subnodes that are placed in the figure from the top down according to their rank in the processing.

As shown in Fig. 11.14, the preprocessing stage (Node 1) which precedes the keywords expansion stage (Node 2) consists in extracting the most relevant keywords from the question by removing stopwords. Given the example of the

question<sup>5</sup> “ما هو المنصب الذي تقلده سيلفيو برلسكوني؟” (What is the position held by Silvio Berlusconi?), the set of keywords to be considered at the keywords expansion stage (Node 2) is formed of the following terms: تقلده (position), المنصب (that was held), سيلفيو برلسكوني (Silvio Berlusconi). Figure 11.14 depicts results after applying this semantic stage on the keyword “المنصب”. The process starts by extracting the AWN synsets that are equivalent to the stem of the keyword concerned.

Therefore, three synsets are found: manoSib\_n1AR, manoSib\_n2AR and \$aHana\_v1AR. After that, the process looks for the SUMO concepts linked to the three AWN synsets. “Position” is the only concept that is linked to these synsets.

Using the three AWN synsets and the SUMO concept, the process can then move to the next step of the semantic QE. This step consists in extracting the AWN synsets that are synonyms, supertypes (hypernym), subtypes (hyponym) and SUMO definitions related synsets of the three synsets. For instance, the synset “manoSib\_n1AR” has the following related synsets:

- One synonym synset which is وظيفة (wZyfp : job);
- A supertype which is مهنة (mhnp : profession) and no subtypes. These related synsets are linked to the original synset “manoSib\_n1AR” using the “hyponym” relation in AWN;
- The concept “Position” which is linked to the considered original synset has the following definition in SUMO: “A formal position of responsibility within an &%Organization. Examples of Positions include president, laboratory director, senior researcher, sales representative, etc.” Given that in SUMO definitions, concepts are preceded by the symbols “&%” and “?”, the process can identify the SUMO concept “Organization” as being related to the “Position” concept. This new concept is linked to the AWN synset “جعية” (jamoEiy~ap : association). The neighborhood (supertypes and subtypes) of this new synset allows us to reach new related synsets such as: “منظمة” (munaZ{ amap : organization), “جماعة” (jamaAEap : community), “حكومة” (Hkwmp : government) and “نظام سياسي” (niZaAm siyaAsiy : political system). The SUMO concept “Organization” is also linked to other synsets such as “رئيس” (ra}iys : Chairman). Therefore, new terms could be reached in the neighborhood of this other synset such as “ملك” (malik : king), “وزير” (ra}iys AlwizaraA’ : prime minister) and “رئيس الدولة” (ra}iys Ald awolap : nation president).

In Fig. 11.14, we illustrate that related synsets are extracted after applying recursively the four kinds of QE on each original and generated synset. The same figure shows the words that can replace the keyword “المنصب” in the question in order to generate new queries to be passed to the used Search Engine (SE). These words are those belonging to the generated synsets. In the case of named entities

---

<sup>5</sup>Example from the CLEF 2005 questions set.

(NEs), the keyword is preceded in the question by its supertype rather than replaced by this supertype.

## Improving the QE Resource

The described semantic QE process is applied on each question keyword. Hence, if none of the question keywords is covered by AWN the question cannot be processed. A first analysis of the TREC and CLEF questions shows that the use of the current release of AWN allows applying the process on only 19.9 % of the CLEF questions and 42.8 % of the TREC questions. In addition, the average of related terms that can be generated per question is very low (three terms per question in the case of extension by synonymy). Therefore, an extension of the AWN resource was necessary in order to achieve a twofold goal: (i) increasing the number of questions to be considered in the experiments and (ii) enlarging the list of generated terms per question.

In the NLP community, building large-scale language resources is devoted more attention as a prerequisite to any NLP application. In the community of WordNets (WNs), there are many such resources having similar structures. However, differences in terms of coverage are observed between European WNs and Semitic WNs. This is mainly due to the fact that the former was and is still concerned by a large number of extension works and approaches. The extension of Semitic WordNets such as AWN and Hebrew WN can profit from these European languages in terms of approaches and experiences with a special focus on the improvement of Semitic WN's usability. In this perspective and in order to reach the previously mentioned twofold goal, the AWN extension has been investigated in the framework of existing European WN approaches (such as English and Spanish) and for Arabic QA usability. Obviously, there are many lines that can be followed and therefore priorities have to be defined. In order to define priorities in the framework of Arabic QA, preliminary experiments were conducted and analysis of failed questions (i.e. questions for the QE process fail to improve the quality of returned passages and to get the expected answer in the first five ranks) was done. Hence, three main priorities were figured out: (i) adding new NEs in AWN, (ii) adding new verbs and (iii) refining synsets hierarchy for nouns. Following these three lines, AWN was firstly extended by 300,000 NEs extracted from the YAGO<sup>6</sup> ontology [57]. These NEs were semi-automatically translated (into Arabic) and mapped with existing synsets by means of WordNet-based and heuristic-based techniques [6]. After that, verbs extracted from the English VerbNet<sup>7</sup> as well as the Unified Verb Index<sup>8</sup> (UVI)

---

<sup>6</sup>YAGO is an ontology with large number of entities (around three million NEs) and related facts written in English. It is built from WordNet and Wikipedia and is connected with the SUMO ontology. It is available at <http://www.mpi-inf.mpg.de/YAGO-naga/YAGO/downloads.html>

<sup>7</sup><http://verbs.colorado.edu/~mpalmer/projects/verbnet.html>

<sup>8</sup><http://verbs.colorado.edu/verb-index/>

were translated into Arabic in order to increase the coverage of verbs in AWN. This translation resulted in around 6,600 verbs that were considered in the process of mapping verbs with their corresponding AWN synsets through three of the eight existing mapping heuristics. Note that these heuristics were previously used in many WordNet building and enrichment projects, e.g. the work presented by Rodríguez et al., in [54] and that for the sake of accuracy the other five heuristics were not considered (these five heuristics generated many new but irrelevant verb/synset connections). The third line that was considered for AWN extension consisted in refining the hierarchy of nouns in the AWN ontology. The objective behind this refinement is to be able to generate more terms using QE by subtypes and supertypes. In this third line, a two-stage technique was adopted:

- Stage 1: Identifying hyponymy patterns over Web snippets. A seed list of AWN hypernym/hyponym pairs helped in generating Web queries while the Maximal Frequent Sequences algorithm [29] allowed extracting patterns from the resulting snippets;
- Stage 2: Instantiating the identified patterns. The instantiation is performed by searching for hypernym/X and X/hyponym pairs which match the given pattern where X stands for the searched part of the pair.

The different techniques used in the three AWN extension lines are characterized by their independence from the set of questions that have been analyzed for the definition of AWN extension priorities. In fact, the extension processes do not start from the keywords of these questions but rather from other resources. These processes allowed adding new instances (around 433,333 NEs), new verbs (+122 % of what exist in the original AWN) and new hypernym/hyponym associations (+5 %).

The whole AWN extension can be explored using an online Web interface<sup>9</sup> which allows identifying original and new information in AWN. Figure 11.15 illustrates a sample entry from this interface.

As depicted in Fig. 11.15, a user can explore the hierarchy of original and extended AWN synsets through the tree displayed on the left side of the interface. In our example, the focus is on the synset شاهد (\$Ahd : to see) under the lexical node فعل (fEl : verb). On the right side, the interface shows that this original synset (we can identify that it is original i.e. it is part of the non-extended version of AWN since it does not contain the label NS) was augmented by six new word senses (those having the label NWS) such as لمح (lmH : to descry) and لاحظ (IAHZ : to take notice). Similarly, the interface helps the user in identifying new added synsets, instances (i.e. NEs of the selected synset) and word forms (for instance broken plural form of words or root form of verbs).

To sum up, the keyword-based level generates for each question keyword related terms on the basis of AWN semantic relations. In order to guarantee that question keywords are more likely covered by AWN and therefore by the proposed QE

---

<sup>9</sup>The AWN extension Web interface is available at [http://sibawayh.emi.ac.ma/awn\\_extension](http://sibawayh.emi.ac.ma/awn_extension)

**Fig. 11.15** Example of the extension of the verb synset شاهد (\$Ahd: to see)

process, AWN has been semi-automatically extended. The QE process can generate a higher number of related terms per question keyword due to the fact that process is recursive and that the extended AWN provides more related terms. This situation can result in extraction of irrelevant snippets by the PR system integrating the proposed QE process.

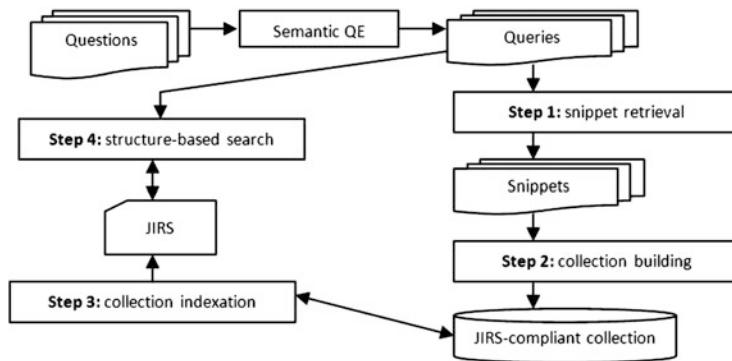
In order to limit the eventual negative impact of generated terms using recursion, preliminary experiments were conducted and showed that by ascending or descending two levels in the AWN hierarchy, the recursive QE process with subtypes and supertypes can improve results by 8 % for the accuracy and by around 4 % for the MRR [3, 4]. Thus, setting this limitation can reduce the number of generated terms without affecting performance.

In addition, the problem of generating a higher number of related terms by using the extended AWN can be resolved at the structure-based level of the approach. The following section gives an overview of this level.

### Distance Density N-Gram Model Combined with Semantic QE

The structure-based level of the current approach uses JIRS that implements the Distance Density N-gram Model [31]. This model considers a sequence of  $n$  adjacent words (n-gram) extracted from a sentence or a question. All possible n-grams of the question are searched in the collection in concern. It also assigns them a score according to the n-grams and weight that appear in the retrieved passages.

The JIRS system has the advantage of being language-independent because it processes the question and the passages without using any language knowledge, lexicon or syntax [30]. However, some adaptations were added to the system in order to be used in the context of the Arabic language [9]. The main modifications



**Fig. 11.16** Steps of the JIRS integration in the structure-based level

were made on the Arabic language-related files (text encoding, stop-words, list of characters for text normalization, Arabic special characters, question words, etc.).

In the same work, the authors conducted preliminary experiments based on a test-bed developed for Arabic with the same characteristics of the CLEF 2006 collection. These experiments show that JIRS can retrieve relevant passages also in Arabic reaching a coverage up to 69 % and a redundancy up to 3.28 when light-stemming is also processed. It was reported by Gómez et al. in [31] that this QA-oriented system improves the coverage measure. The same authors in [32], prove empirically that the JIRS system enhances the MRR. Both works used a Spanish collection and 200 CLEF questions. Balahur et al., in [8], also used the system in increasing the precision of a process that combines opinion analysis with other challenges, such as the ones related to English QA.

With respect to these encouraging experiments, it makes sense to use JIRS in the structure-based level of this approach in order to assign a higher relevance to passages that present a higher probability to contain the expected answer. Often, the expected answer appears in a passage with the same question structure. For instance, in our previous example, the expected structure is **النصب الذي تقلده سيلفيو بيرلسكوني** (the position held by Silvio Berlusconi). Thus passages that contain this structure will be assigned the highest relevance measured by JIRS score.

The structure-based level integrates JIRS according to four steps as illustrated in Fig. 11.16. We can summarize these steps as follows:

- Step 1 “snippets retrieval”: using the Yahoo API, 1,000 distinct snippets corresponding to each question query (generated using the semantic QE described in the previous section) are extracted from the Web. This number was set on the basis of previous experiments where it has been checked that the optimal value is between 800 and 1,000 snippets for the Spanish CLEF document collection [31];
- Step 2 “collection building”: a collection is built using the extracted snippets in the format of JIRS files;

**Table 11.3** Questions covered by semantic QE using the extended AWN

	By synonyms	By subtypes	By supertypes	By SUMO def.	Semantic QE
CLEF	412	304	337	244	412
%	53.9	39.8	44.1	31.9	53.9
TREC	1,135	660	694	704	1,135
%	75.7	44	46.3	46.9	75.7
Total	1,547	964	1,031	948	1,547
%	68.3	42.6	45.5	41.9	68.3

- Step 3 “collection indexation”: the built collection is indexed using the indexation process provided by the JIRS API;
- Step 4 “structure-based search”: each question query is searched again but now against the generated index and relying on the Distance Density N-gram Model implemented in JIRS.

After step 4, each query generated from the question results in five ranked passages. This passage ranking is based on the similarity score calculated by the JIRS system according to the Distance Density N-gram Model. For instance, if we have 20 queries generated from the question in concern, then 20 sets of 5 ranked passages are retrieved using the JIRS index corresponding to the collection built in step 2. From these sets, the five passages with the best JIRS scores are considered as the most relevant passages for the question.

Using the keyword-based and structure-based levels described in Sects. 11.1 and 11.2, experiments were carried out on the set of 2,264 Arabic TREC and CLEF questions. The obtained results and the improvements made are discussed in the following section.

### Experiment Results, Analysis of Questions and Performance Improvement

The conducted experiments allow investigating three main issues: (i) the ability of the used resource (original and extended AWN) to support the proposed QE process, i.e. what is the percentage of questions that can be expanded per QE type (by synonyms, by supertypes, etc.); (ii) the ability of the used resource to generate a high number of new related terms per question; (iii) the impact of the keyword and structure based levels on performances in terms of accuracy, MRR and answered questions. Concerning the first two issues, experiments show that using the extended AWN a significant improvement is reached in terms of covering the questions with the QE process and generating higher number of related terms. Tables 11.3 and 11.4 show the obtained results.

As Tables 11.3 and 11.4 reveal, the semantic QE process could be applied on around 68 % of the TREC and CLEF questions thanks to the extended release of AWN (vs 35 % with the standard release). The other advantage of the proposed AWN extension is generating a higher number of terms per question. In fact, the average of generated terms has been significantly improved for the QE by

**Table 11.4** Comparison between standard and extended AWN in terms of generated terms per question

QE type	Avg. generated terms per question	
	Original AWN	Extended AWN
By synonyms	3	18
By subtypes	1	15
By supertypes	2	23
By definition	1	1

**Table 11.5** Results before and after AWN enrichment

Measures	Without the two-level approach	Using the approach (original AWN)	Using the approach (extended AWN)
Acc@G (%)	9.66	17.49	26.76
MRR@G	3.41	7.98	11.58
AQ@G (%)	20.27	23.15	35.94

Synonyms, by Subtypes and by Supertypes. Since the AWN extension did not focus on the improvement of SUMO concepts/AWN synsets connections, the average of QE by Definition remained the same.

As regards the third issue investigated by the conducted experiment, let us recall that terms generated by the semantic QE process are used then to formulate new Web queries. These queries result in a number of snippets that are processed by JIRS following the steps described above. The answer of each question is then checked in the first five snippets. In order to evaluate performances we use the QA measures describe in Sect. 11.2.5.

Table 11.5 shows the results obtained for these measures with respect to three cases [2]: (i) using only the SE, (ii) using the two-level approach relying on the standard release of AWN and (iii) using the same approach but relying on the extended AWN.

From Table 11.5, it is clear that the proposed approach which is based on the combination of the keyword-based and structure-based levels has a significant impact on performances. Indeed, the accuracy reached 17.49 % (vs 9.66 % without the two-level approach), the global MRR passed from 3.41 to 7.98 and the number of answered questions increased by around 3 % (from 20.27 to 23.15 %). Table 11.5 also highlights the prominent role of extending AWN which resulted in semantically expanding a higher number of questions. In fact, this role is raised by roughly 27 % of accuracy, 12 of global MRR and 36 % of answered questions. The statistical significance of these results was prooved by a t-test [5].

The above results show that both retrieval performance (measured by means of accuracy and number of answered questions) and passage ranking quality (measured by MRR) are enhanced after using the proposed approach. These results are encouraging due to the following facts:

- The obtained results can be compared to those of QA systems for English. Indeed, Kim et al. report in [41] a 38.6 MRR and 51.8 % of answered questions

with respect to lenient evaluation. Aktolga et al. report in [7] that they reached a 26.1 % accuracy, a 36.3 MRR and 36.4 % of answered questions. Let us recall that their experiments consider 622 questions from TREC 2000 and TREC 2001 and that their approach also integrates a QE process based on only WordNet synonymy relations. As we can see, the results in terms of accuracy and number of answered questions are approximately similar.

- Most of the processed Arabic questions contain NEs. This problem was slightly resolved by integrating new entities in AWN from YAGO. However, many NEs in Arabic can have different spelling variants.<sup>10</sup> If such variants are not recognized then the semantic QE approach cannot be applied.
- In the current work, the Web has been queried instead of a document collection. This is a double-edged sword. While the high redundancy of the Web allows a better recall, some expected answers cannot be reached due to information depreciation. This problem concerns particular information about persons. If we take the previous example, results in the Web for the position held by “Silvio Berlusconi” will emphasize on current positions rather than those held in previous periods. For instance, if the question looks for his position in 1990 the answer may not appear in the first 1,000 snippets. This concerns many questions since the TREC and CLEF set of questions belong mostly to the year 2004.
- For languages such as Spanish and English, previous experiments with JIRS report that this system is more likely to retrieve relevant passages when high redundancy is guaranteed. Unfortunately, the Web does not provide enough redundancy for many Arabic TREC and CLEF questions. This is mainly due to the fact that these translated questions come from the European and the American cultures and are therefore not covered by Arabic Web content.
- The snippets are too small to contain both question keywords and the expected answer. For example, in a document which contains the structure “المصب الذي” “رئيـس الـوزراء الـاـيـطـالـي” “تقـلـدـه سـيلـفـيو بـرـلـسـكـوـنـي”, the expected answer “رئـيـس الـوزـراء الـاـيـطـالـي” may not occur in a snippet composed of 20 words if the distance between this answer and that structure is higher than 20.

The present work has contributed in the area of Semitic QA systems from different perspectives through: (i) the development of new modules (semantic QE process), resources (extended AWN) and approaches (statistical approach based on the combination of QE and QA-oriented PR systems); (ii) building evaluation datasets for Arabic QA evaluation which is lacking in the community (in fact, the list of Arabic TREC and CLEF questions can be reused by other researchers); (iii) conducting new experiments with language-independent QA tools (i.e. JIRS) and measures which allow benchmarking with existing campaigns so far not concerned with Semitic languages.

---

<sup>10</sup>The NE resource available at <http://langtech.jrc.it/JRC-Names.html> claims integrating up to hundreds of variants for a single person.

This work can be followed by other Semitic QA projects since the resource namely AWN exists for these languages with similar structure and semantic relations and the used statistical approach supported by JIRS is language-independent.

As future works, the proposed AWN extension can be a start point towards a more refined release to be integrated in the semantic QE process. In this direction, other AWN extension lines can be considered in particular with the addition of new semantic relations such as meronymy (i.e., is part-of relation) and the enrichment of SUMO concept and AWN synsets connections.

## 11.5 Summary

In this chapter we have described some key concepts to be taken into consideration when building a QA system for a Semitic language. Two major aspects have been described:

1. The QA task: that is unique in its goal, i.e. fetch an answer for a specific question in a pool of documents written in natural language. It is also unique in that the best QA systems are built not only by investigating the appropriate techniques but also by putting together several building blocks where each one plays a specific role.
2. The characteristics of the Semitic languages: for which we have tried to show that there are intrinsic characteristics, such as the lack of short vowels and the sparseness of data, that can stymie building robust statistical models for the different building blocks of the systems. But also the scarcity of resources and annotated data might be a challenging obstacle towards obtaining the desired performance.

In Sect. 11.4 we have described two research works, both participating systems in the Arabic QA task in the QA4MRE track at CLEF 2012. In each one of them specific modules of QA have been developed for Arabic. In the first one, the main aim of the research work was to investigate an efficient technique to answer Arabic definition questions. In order to achieve this endeavor, the DefArabicQA system was developed. This system is based on a linguistic knowledge approach. Although this approach has been widely used in the state of the arts, Trigui et al. in [63] have focused on its impact on the Arabic QA, and its benefits when it is applied to the Web as a source of descriptive information instead of a collection of news documents. The experimental results achieved show clearly the efficiency of the proposed technique. The second research work os aimed at improving the performances of Arabic QA systems that target open and dynamic domains. To do so, the focus was placed on enhancing the QE. Expertise learnt from this work after building the whole system is threefold:

- When working in dynamic domains such as the Web where the collection of documents grows over time, the F-measure cannot be applied because the recall

and precision cannot be calculated. It is important also to mention that the three other measures (MRR, accuracy and AQ) have the same increasing trend.

- Concerning the QE module, it is worth extending the question in order to get more opportunity to answer it from documents. The extension has to be done using the closest semantic neighborhood of question keywords. For this purpose, the semantic resource to rely on (AWN in our case) should be as accurate as possible and as wide as possible in terms of domain/language coverage.
- Finally, attention should also be devoted to PR. When using the Distance Density N-gram Model, PR has the effect of decreasing the negative effect of QE noise (if any) and improving the quality (MRR) of returned passages.

Our main goal was to offer a chapter that could help researchers, engineers and practitioners better understand the task and the intricacies involved in building them especially when it comes to morphologically rich/complex languages such as the Semitic languages.

## References

1. Abney S, Collins M, Singhal A (2000) Answer extraction. In: Proceedings of the 6th applied natural language processing conference, Seattle, pp 296–301
2. Abouenour L (2011) On the improvement of passage retrieval in Arabic question/answering (Q/A) systems. In: Mueoz R et al (eds) NLDB'11, Alicante. Lecture notes in computer science, vol 6716/2011. Springer, Berlin/Heidelberg, pp 336–341. doi:10.1007/978-3-642-22327-3\_50
3. Abouenour L, Bouzoubaa K, Rosso P (2009) Three-level approach for passage retrieval in Arabic question/answering systems. In: Proceedings of the 3rd international conference on Arabic language processing CITALA2009, Rabat
4. Abouenour L, Bouzoubaa K, Rosso P (2009) Structure-based evaluation of an Arabic semantic query expansion using the JIRS passage retrieval system. In: Proceedings of the workshop on computational approaches to Semitic languages, E-ACL-2009, Athens, April 2009. Association for Computational Linguistics (ACL), Stroudsburg, pp 62–68
5. Abouenour L, Bouzoubaa K, Rosso P (2010) An evaluated semantic QE and structure-based approach for enhancing Arabic Q/A. In: The special issue on “advances in Arabic language processing” for the IEEE Int J Inf Commun Technol (IJICT). ISSN:0973-5836, Serial Publications
6. Abouenour L, Bouzoubaa K, Rosso P (2010) Using the Yago ontology as a resource for the enrichment of named entities in Arabic WordNet. In: Workshop on language resources (LRs) and human language technologies (HLT) for Semitic languages status, updates, and prospects, LREC’10 conference, Malta
7. Aktolga E, Allan J, Smith DA (2011) Passage reranking for question answering using syntactic structures and answer types. In: ECIR 2011, Dublin. Lecture notes in computer science, vol 6611. Springer, Berlin/Heidelberg, pp 617–628
8. Balahur A, Boldrini E, Montoyo A (2010) The OpAL system at NTCIR 8 MOAT. In: Proceedings of NTCIR-8 workshop meeting, Tokyo
9. Benajiba Y, Rosso P, Gómez JM (2007) Adapting JIRS passage retrieval system to the Arabic. In: Proceedings of the 8th International conference on computational linguistics and intelligent text processing, CICLing-2007, Mexico City. Lecture notes in computer science, vol 4394. Springer, pp 530–541
10. Benajiba Y, Rosso P, Lyhyaoui A (2007) Implementation of the ArabiQA question answering system’s components. In: Proceedings of ICTIS-2007, Fez

11. Benajiba Y, Zitouni I, Diab M, Rosso P (2010) Arabic named entity recognition: using features extracted from noisy data. In: Proceedings of the 48th annual meeting of the Association for Computational Linguistics, ACL-2010, Uppsala, 11–16 July 2010, pp 281–285
12. Bilotti M (2004) Query expansion techniques for question answering. Master's thesis, Massachusetts Institute of Technology
13. Bilotti M, Katz B, Lin J (2004) What works better for question answering: stemming or morphological query expansion?. In: Proceedings of the information retrieval for question answering (IR4QA) workshop at SIGIR 2004, Sheffield
14. Blair-Goldensohn S, McKeown K, Schlaikjer AH (2003) A hybrid approach for QA track definitional questions. In: TREC 2003, Gaithersburg, pp 185–192
15. Borg C, Rosner M, Pace G (2009) Evolutionary algorithms for definition extraction. In: The 1st workshop on definition extraction 2009, RANLP, Borovets
16. Boudlal A, Lakhouaja A, Mazroui A, Meziane A, Ould Abdallah Ould Bebah M, Shoul M (2010) Alkhali MorphoSys: a Morphosyntactic analysis system for non vocalized Arabic. In: International Arab conference on information technology (ACIT'2010), Benghazi
17. Chu-Carroll J, Prager J (2007) An experimental study of the impact of information extraction accuracy on semantic search performance. In: Proceedings of the 16th ACM conference on information and knowledge management, New York, pp 505–514
18. Chu-Carroll J, Prager J, Czuba K, Ferrucci D, Duboue P (2006) Semantic search via XML fragments: a high-precision approach to IR. In: Proceedings of the annual international ACM SIGIR conference on research and development on information retrieval, Seattle, pp 445–452
19. Clarke C, Cormack G, Lynam T (2001) Exploiting redundancy in question answering. In: Proceedings of the 24th ACM SIGIR conference, New York, pp 358–365
20. Cui H, Kan M, Chua T (2005) Generic soft pattern models for definitional question answering. In: SIGIR 2005, Salvador, pp 384–391
21. Cui H, Kan MY, Chua TS (2007) Soft pattern matching models for definitional question answering. ACM Trans Inf Syst (TOIS) 25(2):1–30
22. Demeke G, Getachew M (2006) Manual annotation of Amharic news items with part-of-speech tags and its challenges. ELRC working papers 2:1–17
23. Denicia-Carral C, Montes-y-Gómez M, Villaseñor-Pineda L, García Hernández R (2006) A text mining approach for definition question answering. In: 5th international conference on natural language processing, FinTal 2006, Turku. Lecture notes in artificial intelligence
24. Doddington G, Mitchell A, Przybocki M, Ramshaw L, Strassel S, Weischedel R (2004) ACE program – task definitions and performance measures. In: Proceedings of LREC, Lisbon, pp 837–840
25. Elkateb S, Black W, Vossen P, Farwell D, Rodríguez H, Pease A, Alkhalifa M (2006) Arabic WordNet and the challenges of Arabic. In: Proceedings of Arabic NLP/MT conference, London
26. Fahmi I, Bouma G (2006) Learning to identify definitions using syntactic features. In: Workshop of learning structured information in natural language applications, EACL, Trento
27. Fellbaum C (2000) WordNet: an electronic lexical database. MIT, Cambridge. [cogsci.princeton.edu](http://cogsci.princeton.edu), 7 Sept
28. Figueira A, Neumann G, Atkinson J (2009) Searching for definitional answers on the web using surface patterns. IEEE Comput 42(4):68–76
29. García-Blasco S, Danger R, Rosso P (2010) Drug-drug interaction detection: a new approach based on maximal frequent sequences. Sociedad Espanola para el Procesamiento del Lenguaje Natural (SEPLN) 45:263–266
30. Gómez JM, Montes-Gomez M, Sanchis E, Villasenor-Pineda L, Rosso P (2005) Language independent passage retrieval for question answering. In: Fourth Mexican international conference on artificial intelligence, MICAI 2005, Monterrey. Lecture notes in computer science. Springer, pp 816–823
31. Gómez JM, Buscaldi D, Rosso P, Sanchis E (2007) JIRS: language independent passage retrieval system: a comparative study. In: 5th international conference on natural language processing, Hyderabad, 2006

32. Gómez JM, Rosso P, Sanchis E (2007) Re-ranking of Yahoo snippets with the JIRS passage retrieval system. In: Proceedings of the workshop on cross lingual information access, CLIA-2007, 20th international joint conference on artificial intelligence, IJCAI-07, Hyderabad
33. Gong Z, Cheang CW, Hou U (2005) Web query expansion by WordNet. In: DEXA 2005, Copenhagen. Lecture notes in computer science, vol 3588, pp 166–175
34. Gong Z, Cheang C, Hou L (2006) Multi-term web query expansion by WordNet. In: Bressan S, Küng J, Wagner R (eds) Database and expert systems applications. Lecture notes in computer science, vol 4080. Springer, Berlin/Heidelberg, pp 379–388
35. Hammou B, Abu-salem H, Lytinen S, Evens M (2002) QARAB: a question answering system to support the Arabic language. In: The proceedings of the workshop on computational approaches to Semitic languages, Philadelphia. ACL, pp 55–65.
36. Hsu MH, Tsai MF, Chen HH (2008) Combining WordNet and ConceptNet for automatic query expansion: a learning approach. In: Asia information retrieval symposium, Harbin. Lecture notes in computer science, vol 4993. Springer, pp 213–224
37. Itai A, Wintner S (2008) Language resources for Hebrew. *Lang Resour Eval* 42:75–98
38. Joho H, Sanderson M (2000) Retrieving descriptive phrases from large amounts of free text. In: 9th ACM CIKM conference, McLean, Nov 2000, pp 180–186
39. Joho H, Liu YK, Sanderson M (2001) Large scale testing of a descriptive phrase finder. In: 1st human language technology conference, San Diego, pp 219–221
40. Kaisser M, Becker T (2004) Question answering by searching large corpora with linguistic methods. In: Proceedings of the thirteenth text REtrieval conference, TREC 2004, Gaithersburg, 16–19 Nov 2004
41. Kim S, Baek D, Kim S, Rim H (2000) Question answering considering semantic categories and co-occurrence density. In: TREC 2000, Gaithersburg
42. Larkey LS, Ballesteros L, Connell ME (2002) Improving stemming for Arabic information retrieval: light stemming and co-occurrence analysis. In: Proceedings of ACM SIGIR, Tampere, pp 269–274
43. Maamouri M, Bies A, Buckwalter T (2004) The Penn Arabic Treebank: building a largescale annotated Arabic corpus. In: NEMLAR conference on Arabic language resources and tools, Cairo
44. Miliaraki S, Androutsopoulos I (2004) Learning to identify single-snippet answers to definition questions. In: COLING 2004, Geneva, pp 1360–1366
45. Moldovan D, Pasca M, Harabagiu S, Surdeanu M (2003) Performance issues and error analysis in an open-domain question answering system. In: Proceedings of the 40th annual meeting Association of Computational Linguistics, New York, 2003
46. Moldovan D, Clark C, Harabagiu S (2003) COGEX: a logic prover for question answering. In: NAACL '03 proceedings of the 2003 conference of the North American chapter of the Association for Computational Linguistics on human language technology – vol 1, Edmonton, pp 87–93.
47. Monz C (2003) From document retrieval to question answering. Ph.D. dissertation, Institute for Logic, Language, and Computation, University of Amsterdam
48. Nanba H (2007) Query expansion using an automatically constructed thesaurus. In: Proceedings of NTCIR-6 workshop meeting, Tokyo, 15–18 May 2007
49. Niles I, Pease A (2003) Linking lexicons and ontologies: mapping WordNet to the suggested upper merged ontology. In: Proceedings of the 2003 international conference on information and knowledge engineering, Las Vegas
50. Pinto FJ(2008) Automatic query expansion and word sense disambiguation with long and short queries using WordNet under vector model. In: Proceedings of workshops of Jornadas de Ingeniería del Software y Bases de Datos, Gijón, vol 2(2)
51. Ravichandran D, Hovy EH (2002) Learning surface text patterns for a question answering system. In: Proceedings of ACL, Philadelphia, pp 41–47
52. Robertson E, Walker S, Beaulieu M (2000) Experimentation as a way of life: okapi at TREC. *Inf Process Manag* 36(1):95–108

53. Rodrigo A, Peñas A, Verdejo F (2010) Answer validation exercises at CLEF. In: Tufis D, Forascu C (eds) *Multilinguality and interoperability in language processing with emphasis in Romanian*. Ed. Academiei Romane, Bucureşti, pp 135–148
54. Rodríguez H, Farwell D, Farreres J, Bertran M, Alkhalifa M, Antonia Martí M, Black W, Elkateb S, Kirk J, Pease A, Vossen P, Fellbaum C (2008) Arabic WordNet: current state and future extensions. In: Proceedings of the fourth international global WordNet conference – GWC 2008, Szeged
55. Rosso P, Hurtado L, Segarra E, Sanchis E (2012) On the voice-activated question answering. *IEEE Trans Syst Man Cybern–Part C* 42(1):75–85
56. Harabagiu SM, Moldovan DI, Clark C, Bowden M, Williams J, Bensley J (2003) Answer mining by combining extraction techniques with abductive reasoning. In: TREC 2003, Gaithersburg, pp 375–382
57. Suchanek FM, Kasneci G, Weikum G (2007) YAGO: a core of semantic knowledge unifying WordNet and Wikipedia. In: Proceedings of the 16th WWW, Banff, pp 697–706
58. Tanev H, Kouylekov M, Magnini B (2004) Combining linguistic processing and Web mining for question answering: ITC-irst at TREC 2004. In: Voorhees EM, Buckland LP (eds) TREC 2004, Gaithersburg. National Institute of Standards and Technology (NIST)
59. Tellez S, Katz B, Lin J, Fernandes A, Marton G (2003) Quantitative evaluation of passage retrieval algorithms for question answering. In: Proceedings of the 26th annual international ACM SIGIR conference on research and development in information retrieval, New York, 2003, pp 41–47
60. Trigui O (2011) How to extract Arabic definitions from the Web? Arabic definition question answering system. In: NLDB’11 proceedings of the 16th international conference on natural language processing and information systems, Alicante, pp 318–323
61. Trigui O, Belguith HL, Rosso P (2010) DefArabicQA: Arabic definition question answering system. In: Workshop on language resources and human language technologies for Semitic languages, 7th LREC. Valletta, pp 40–44
62. Trigui O, Belguith HL, Rosso P (2010) An automatic definition extraction in Arabic language. In: 15th international conference on applications of natural language to information systems (NLDB 2010), Cardiff. Lecture notes in computer science, vol 6177. Springer, Heidelberg, pp 240–247
63. Trigui O, Belguith HL, Rosso P (2011) ARABTEX. In: 7th international computing conference in Arabic (ICCA’11), Riyadh
64. Voorhees E (2003) Overview of the TREC 2003 question answering track. In: TREC 2003, Gaithersburg, pp 54–68
65. Westerhout E (2009) Extraction of definitions using grammar-enhanced machine learning. In: Student research workshop, EACL 2009, Athens, pp 88–96
66. Xu J, Licuanan A, Weischedel RM (2003) TREC 2003 QA at BBN: answering definitional questions. In: 12th TREC, Washington, DC, pp 98–106
67. Zhang Z, Zhou Y, Huang X, Wu L (2005) Answering definition questions using web knowledge bases. In: Dale R, Wong K-F, Su J, Kwong OY (eds) IJCNLP 2005, Jeju. Lecture notes in computer science (lecture notes in artificial intelligence), vol 3651. Springer, Heidelberg, pp 498–506

# Chapter 12

## Automatic Summarization

Lamia Hadrich Belguith, Mariem Ellouze, Mohamed Hedi Maaloul,  
Maher Jaoua, Fatma Kallel Jaoua, and Philippe Blache

### 12.1 Introduction

This chapter addresses automatic summarization of Semitic languages. After a presentation of the theoretical background and current challenges of automatic summarization, we present different approaches suggested to cope with these challenges. These approaches fall into two classes: single vs. multiple document summarization approaches. The main approaches dealing with Semitic languages (mainly Arabic, Hebrew, Maltese and Amharic) are then discussed. Finally, a case study of a specific Arabic automatic summarization system is presented. The summary section draws the most insightful conclusions and discusses some future research directions.

The *summary* notion was defined by the American National Standards Institute, in 1979 as follows: *an abbreviated, accurate representation of the content of a document preferably prepared by its author(s) for publication with it. Such abstracts are also useful in access publications and machine-readable databases.*

Text summarization is then the process to reduce the length or complexity of the original text without losing the main content, by means of different techniques such as selection and/or generalization on what is important in the source [52]. According to [75], text summarization is the process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user (or users) and task (or tasks).

---

L.H. Belguith (✉) • M. Ellouze • M. Jaoua • F.K. Jaoua  
ANLP Research Group, MIRACL Laboratory, FSEGS, University of Sfax, B.P 1088, 3018 Sfax, Tunisia  
e-mail: [l.belguith@fsegs.rnu.tn](mailto:l.belguith@fsegs.rnu.tn); [Mariem.Ellouze@planet.tn](mailto:Mariem.Ellouze@planet.tn); [Maher.Jaoua@fsegs.rnu.tn](mailto:Maher.Jaoua@fsegs.rnu.tn);  
[kalleljaouafatma@gmail.com](mailto:kalleljaouafatma@gmail.com)

M.H. Maaloul • P. Blache  
LPL Laboratory, 5 avenue Pasteur, CP 13100 Aix-en-Provence, France  
e-mail: [mohamed.maaloul@lpl-aix.fr](mailto:mohamed.maaloul@lpl-aix.fr); [blache@lpl-aix.fr](mailto:blache@lpl-aix.fr)

The input of the summarization process has traditionally been focused on text. But it can also be multimedia information, such as images, videos or audios, as well as on-line information or hypertexts. Furthermore, summarizing can be applied to multiple documents. This process is known as multi-document summarization and the source documents in this case can be in a single-language (monolingual) or in different languages (translingual or multilingual).

The output of the summarization process may be an extract (i.e. when a selection of *significant* sentences of a document is performed) or an abstract (i.e. when the summary can serve as a substitute for the original document) [29].

The above aspects tackled by the summarization task should be considered by an automatic process of this task. Automatic summarization is a very interesting and useful task to cope with the increasing availability of online information on the World Wide Web. Although attempts to generate automatic summaries started in 1950 [63], automatic summarization is still a challenging research area. Various research works related to this area have been published (details will be presented in the following sections of this chapter). However, most of these works have concentrated on indo-European languages. Works on Semitic languages are relatively few and not very advanced.

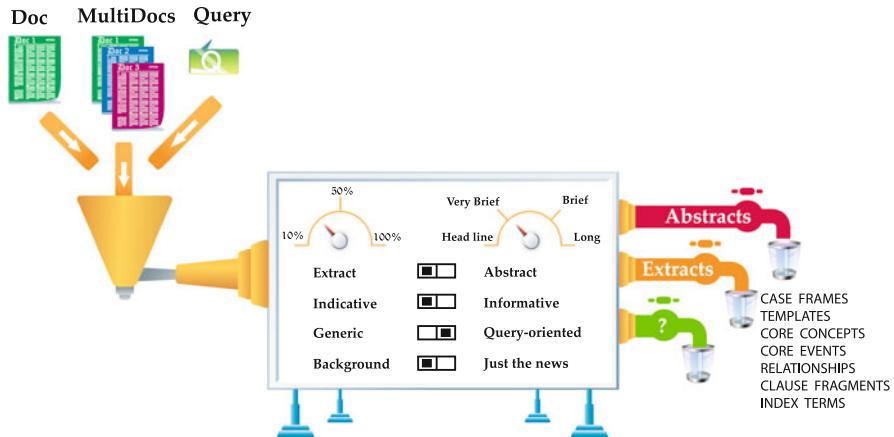
This chapter intends then to investigate the approaches and systems developed for Semitic languages in the context of automatic summarization.

The rest of the chapter is organized as follows: Sect. 12.2 describes some aspects related to text summarization. Section 12.3 presents the main evaluation campaigns and the common measures to evaluate automatic summarization systems. Sections 12.4 and 12.5 give an overview of the different approaches to generate summaries for both single and multiple documents. Summarization approaches built for Semitic languages are outlined in Sect. 12.6. A case study of an Arabic summarization system is detailed in Sect. 12.7. Finally, Sect. 12.8 concludes this chapter.

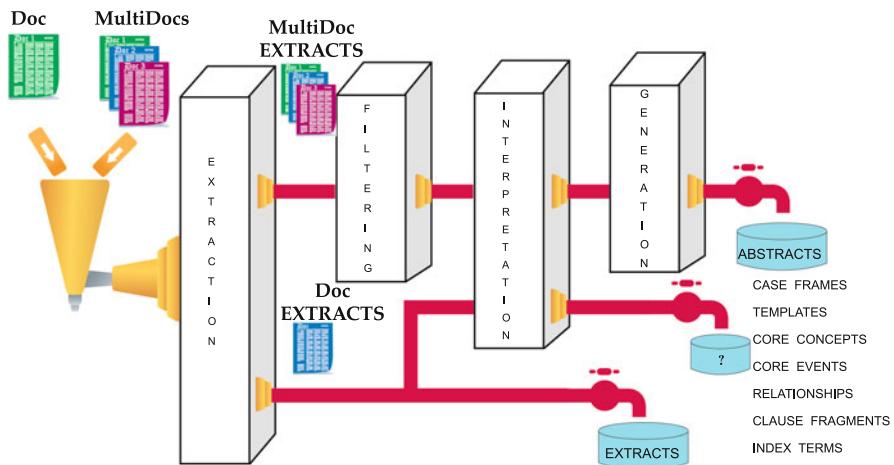
## 12.2 Text Summarization Aspects

The concepts presented throughout this section (summary types, user types, summary characteristics, etc.) should be taken into account in the automation of the summarization task. The summarization machine described by Hovy and Marcu [47] satisfies this condition. It shows that a source text could have different summaries (see details in Sect. 12.2.1). Indeed, this machine can be configured according to user preferences. The parameters taken into account could be the summary size, type, etc. Note that these parameters have been discussed by several authors [52, 70].

The summarization machine accepts, as input, a single or multiple documents with or without a user query and generates, as output, an abstract (a short well-written text), an extract (a set of pertinent sentences), a template, or a frame. Extract and abstract forms will be discussed in Sect. 12.2.2.



**Fig. 12.1** A summarization machine [47]



**Fig. 12.2** The general architecture of a summarization system [47]

Figure 12.1 shows the inputs, the outputs and the possible options (parameters) of the summarization machine. The general process recommended for a summarization system, shown in Fig. 12.2, requires three main steps: extraction (or topic identification), interpretation and generation. Note that filtering is performed when dealing with multiple document summarization. The required steps depend on the form of the final product delivered to the user.

The first stage, extraction or topic identification produces simple extracts. It filters the input to retain only the most important, central, topics. Generally, we assume that a text could have many sub-topics, and that the topic extraction process could be parameterized in at least two ways: first, by including more or fewer topics to produce longer or shorter summaries, and second, by including only topics

related to the user's expressed interests. Typically, topic identification is achieved using several complementary techniques. These techniques will be discussed in Sects. 1.4 and 1.5.

To produce abstract-type summaries, the core process is the interpretation step. In this step, two or more topics are fused together to form a third, more general one. Since the result is something new, not explicitly contained in the input, this stage requires the system to have access to knowledge different from the input. The results of interpretation are usually unreadable abstract representations. Systems therefore include a step of summary generation to produce human readable text. This step reformulates the extracted and fused material into a coherent, densely phrased, new text.

### 12.2.1 Types of Summaries

The same text may have many different summaries depending on various factors: input, output, purpose, etc. According to Spark Jones [52] and Hovy [45, 46], different summary types can be distinguished:

- Single-document vs. multi-document source: A summary can be based on one text in some cases, but it fuses together many texts in other cases. Note that a multi-document summary is a text that covers the content of more than one input text, and is usually used only when the input texts are thematically related.
- Domain-specific vs. general: A domain-specific summary derives from input text(s) which theme(s) pertain to a single restricted domain. Then, it may be appropriate to apply domain-specific summarization techniques, focus on specific content, and output specific formats. On the contrary, a general-domain summary derives from input text(s) in any domain, and can make no such assumptions [46].
- Extract vs. abstract: The output of a summary system may be an extract consisting of material entirely copied from the source (i.e. a set of *significant* sentences selected from the document). A summary can also be an abstract generated from an internal representation that results from the input analysis [29].
- Fluent vs. disfluent: A fluent summary is written in full, grammatical sentences that are related and follow a logical order, according to the rules of coherent discourse structure. A disfluent summary consists of individual words or text portions that lead to non-grammatical sentences or incoherent paragraphs [46].
- Neutral vs. biased: A neutral summary presents the essential information without providing opinions. It tries to be objective. A biased summary extracts and formulates the content from some point of view. It includes some of the system's own bias through inclusion of material with one bias and omission of material with another.
- Generic vs. query-oriented: Generic summaries can serve as substitute for the original text and try to represent all relevant material of a source text. They provide the author's view whereas query-oriented summaries reflect the user's

interest. User-focused summaries select specific themes of the original text, in response to a user's query.

- Indicative vs. informative: Indicative summaries are used to underline topics that are addressed in the source text. They can give a brief idea of what the original text is about. They provide a reference function for selecting documents for more in-depth reading. Informative summaries are intended to cover the topics in the source text at some level of detail [2, 68]. Thus, they reflect the content, and describe what is in the input text.
- Background vs. just-the-news: A background summary teaches about the topic. It assumes that the reader does not have enough knowledge of the input text(s) content, and hence includes explanatory material, such as circumstances of place, tense, and actors. A just-the-news summary conveys just the newest facts on an already known subject, assuming the reader is familiar with the topic.

### 12.2.2 Extraction vs. Abstraction

Summaries produced by automatic systems can be grouped into two families: extracts and abstracts. On one side, systems producing summaries by abstraction have to understand the document and then generate a grammatical and coherent text. On the other side, systems performing extracts should select units (words, sentences, paragraphs, etc.) that contain the essence of the document and then produce an extract by assembling them. It was observed that about 70 % of the sentences used in manually created summaries are taken from the source text without any modification [60]. As its name implies, an extract is a part taken from a source document to provide an overview (outline) of its contents.

An abstract relies on two phases: first understanding the source document and then rewriting the document in a more compact form. Mani [68] distinguishes the abstract from the extract as follows: an abstract is a text with at least a few units (paragraphs, phrases, words, etc.) which are not present in the source document.

This definition is nevertheless too restrictive. During the production of an extract, a copy-paste process can be applied to informative units such as sentence segments (e.g. nominal or verbal group). Thus, the produced extract will consist of units that do not appear in the original document.

Most of the extraction methods adopt a linear weighting model. In this model, each text unit is weighted according to some features such as: the unit's location in the source text ( $Location(U)$ ), how often it occurs in the source text ( $StatTerm(U)$ ), the appearance of cue phrases ( $CuePhrase(U)$ ), and statistical significance metrics ( $AddTerm(U)$ ). The sum of these individual weights, usually modified by specific tuning parameters attached to the weights, is the overall weight of the text unit  $U$  [41]:

$$Weight(U) := Location(U) + CuePhrase(U) + StatTerm(U) + AddTerm(U) \quad (12.1)$$

Unlike the linear model in extraction methods, abstraction requires more resources for natural language processing (NLP), including grammars and lexicons for parsing and generation. It also requires some common sense and domain-specific ontologies for reasoning during analysis and salience computation [41]. So, we can conclude that extraction is easier than abstraction.

### 12.2.3 *The Major Known Challenges*

The major challenge in summarization lies in distinguishing the most informative parts of a document from the less informative ones. Summarization approaches are often divided into two main groups, text abstraction and text extraction. Text abstraction presents more challenges: it should parse the original text in a deep linguistic way, interpret the text semantically into a formal representation, find new more concise concepts to describe the text and then generate a new shorter text, an abstract, with the same basic information content. Much knowledge in NLP is then required to perform the text abstraction.

Text extraction also faces the challenge of determining the effective features of a text summarization system that extracts the main ideas from source documents and that covers all important themes. This problem is usually addressed by text-clustering approach. For example, Carbonell and Goldstein [18] proposes a contribution to topic driven summary by introducing the Maximal Marginal Relevance (MMR) measure. The idea is to combine query relevance with information novelty. MMR rewards relevant sentences and penalizes redundant ones.

Other problems consist in resolving automatic text summarization in a language-domain-independent way. Efforts towards statistical language-independent processing have been taken by the extractive approach, so summarization steps can possibly be automated in a language-independent system. Such a promising approach has been proposed by many authors [7, 70, 85]. However, due to the lack of resources (annotated corpora, NLP tools) for Semitic languages, the goal of language-independent automatic summarization cannot be reached very soon.

## 12.3 How to Evaluate Summarization Systems

The process of summary system evaluation faces two major problems. First, researchers use different metrics when assessing their systems. Second, the evaluations are performed on different corpora. The absence of standard test corpora and metrics makes it very hard to compare different summarization systems. Thus, some evaluation campaigns have been created. These campaigns have established standard corpora and metrics firstly to encourage researchers to work on the summarization field and secondly to ensure a fair evaluation of the different systems.

### 12.3.1 Insights from the Evaluation Campaigns

DUC (Document Understanding Conference) is the most important evaluation campaign created by the community of the automatic summarization and extraction field. This conference was created by a group of researchers in March 2000 and is directed by the NIST (National Institute for Science and Technology), an organization under the defense advanced research projects agency (DARPA). In 2008, DUC joined TREC (Text Retrieval Conferences) to form only one conference namely the TAC conference (Text Analysis Conference).

One should mention that many of the actual advances in the automatic summarization field are the result of the establishment of this annual evaluation campaign (DUC/TAC). The evaluation performed by these conferences allowed the researchers to evaluate and compare their systems on a large scale and on the same corpora. New summarization tasks have been proposed by these conferences.

These tasks have different inputs (biography, blog, multilingual) and purposes (headline generation, specific). Moreover, the TAC conference proposed an evaluation task to develop new evaluating measures [23].

It should be noted that the availability of previous years' evaluation data for DUC/TAC has also made possible the exploration of supervised summarization methods. Indeed, one could use the corpora released by the conferences, the human summaries as well as the summaries produced by the participating systems.

### 12.3.2 Evaluation Measures for Summarization

Summary evaluation could be performed either automatically or manually. Manual evaluation is based on subjective criteria and produced by a human evaluator to assess the linguistic quality of a summary. Human evaluators generally attribute scores to evaluate the grammaticality of the extract, its coherence and its cohesion. Moreover, human evaluators could determine an overall responsiveness score by comparing the summary to the source text.

For automatic evaluation there are many evaluation measures developed for the evaluation campaigns. In the following, we present three main measures: Rouge, AutoSummEng and Pyramid.

- ROUGE Scores: These metrics allow to evaluate automatically an extract or a summary by comparing it to several model summaries. ROUGE scores are used by DUC/TAC<sup>1</sup> conferences to evaluate both human and system summaries [57]. The scores determined by ROUGE are based on two types of units: model units (MU) representing the n-grams of the words extracted from the human

---

<sup>1</sup><http://www.nplir.nist.gov/projects/duc/>  
<http://www.nist.gov/tac/>

summaries, used as summary models; and peer units (PU) which result from the decomposition of the summaries (or the extracts) generated automatically by the systems into n-grams of words. The ROUGE<sub>n</sub> metric uses the correspondence between the distribution of the words (n-grams) of a candidate summary (PU) and that of the whole human summary (MU). It should be noted that ROUGE<sub>n</sub> is the general formula of the ROUGE metric and ROUGE can adopt various parameters, including word stemming, stop-word removal and n-gram size. For example, ROUGE<sub>2</sub> evaluates bigram co-occurrence while ROUGE<sub>SU4</sub> evaluates skip bigrams with a maximum distance of four words. It should be also noted that there are other metrics of ROUGE called ROUGE<sub>BE</sub> that consider the minimal length of syntactically well-formed units [48, 98]

- AutoSummENG: This metric is based on the extraction of relations between n-grams, given the spatial proximity of those n-grams within a given text [37]. Then, a graph is constructed to indicate the full set of deduced relations. Such representations are extracted from both system and human summaries. The graph edges indicate the mean distance between the neighboring n-grams in all occurrences. A comparison between the graph representation of the generated summary and the summary models is established, returning a degree of similarity between the graphs.
- Pyramid: The Pyramid metric [81] is an indirect manual evaluation metric of summary content. Human assessors read each summary model and determine the semantic content units (SCU) (i.e., the ideas or statements of a text). To each pyramid SCU is attributed a score according to its occurrence in the summarizing system. The pyramid score of a system summary is given by the score of similar SCU contained in the pyramid.

## 12.4 Single Document Summarization Approaches

Automatic summarization methods are based on extraction or on abstraction approaches. The extraction methods generally use a selection algorithm to extract, from the source text, a list of salient textual units (generally sentences). The final extract is then generated from this list, by respecting the chronological order in which the units appear in the source text and ensuring the reduction threshold (i.e., a maximal number of textual units). In the abstraction methods, the computer program attempts to interpret, to some degree, the information contained in the document. The information is condensed into a few core ideas which may be presented in a readable form via natural language.

We discuss in this section methods based on extraction approaches. Extraction methods are based on numerical, symbolic knowledge, or hybrid heuristics (i.e., combining both knowledge types).

### 12.4.1 Numerical Approach

The numerical approach regroups empirical methods which are typically confined with an assignment of scores (i.e. weights) to the text words and thereafter to the text sentences. The score assignment is based on specific rules mainly relative to the identification of indicators or criteria for the importance of each source text sentence.

The final score of a sentence is thus obtained by aggregating the scores of the different sentence units. The final sentence score is considered as its importance degree. The idea is to support sentences which have the highest final scores. The final decision of a sentence belonging to the extract is also based on the reduction threshold, which will increase or decrease the chance of the sentence being selected in the extract. The final extract generation is thus reduced to the concatenation of the sentences with highest scores in the order of their occurrence in the source document.

On the practical level, the score calculation is often done by statistical and probabilistic methods. Thus, the criteria of score attribution can combine different features: word frequency ( $tf * idf$ ) [91], sentence position [29], cue phrases [86], word signature [58], words from the titles, *bonus* and *stigma* expressions [78].

In addition to statistical and probabilistic techniques, sentence selection could rely on learning methods. The general principle of the extraction techniques based on learning consists of taking advantage of the reference summaries (extracts or abstracts) produced by professional summarizers in order to build systems which learn how to classify sentences on the basis of extraction criteria used by professional summarizers [7, 55, 68].

Various learning methods have been proposed. Turney et al. used the Genetic Algorithm to learn the best combinations [99]. The Genetic Algorithm was also adopted by Jaoua et al. to simulate the mechanisms of extract generation and classification in order to determine the best mechanism which maximizes the informational quality of the extract [50].

### 12.4.2 Symbolic Approach

The symbolic approach tries to model the text's discourse structure. Initial works use a linguistic level. We cite as an example the work of Hahn [40] which is based on the identification of the relations of synonymy, generalization and specification. Barzilay proposed to detect robust lexical chains [9] for performing the summarization task.

Subsequently, the use of a symbolic approach based on discourse structure analysis has been imposed. Indeed, the purely symbolic approach is generally done by a formal representation of the knowledge contained in the documents or by paraphrasing techniques [94].

Many works on discourse analysis have been proposed. They are based on the idea that the text structure and coherence can be represented by rhetorical relations. These relations constitute valuable tools to consider the discourse under various angles: (i) they handle the text on a hierarchical basis by connecting its sentences; (ii) they produce a complement to the sentence semantics; (iii) they establish a fundamental model to represent the argumentative and/or intentional organizations which constitute a discourse [90]. The rhetorical structure of text shows then the *centrality* of textual units that reflect their importance. In this context, several symbolic techniques are used for the extraction process, such as the RST (Rhetorical Structure Theory) [72–74, 83].

Other researchers were interested in exploring the context of the linguistic indicators to determine the discursive nature of the utterance [4, 26]. For example, Minel [79] identified a set of discursive categories such as the thematic announcement, the underlining, the definition, the recapitulation and the conclusion. Note that the filtering could be an overlap between the discursive analysis and other factors such as the user profile.

#### 12.4.3 Hybrid Approach

Given that the produced summaries are generally not very coherent because of the disconnection of the extracted sentences from their context, the hybrid approach tries to fill this gap by combining numerical and symbolic methods to take into account the discourse features.

Most research works on extraction are based on the extraction of hybrid-based knowledge coming from various symbolic and numerical sources or even based on score attribution to information extracted in a symbolic way. In this context, Ono et al. [83] undertakes the extraction step by representing the source text in the form of an RST tree and assigning weights to its various nodes according to their position. Thus, it is the final score which judges the relevance of a tree node.

In the same context, Boudabbous et al. [17] proposed a hybrid approach based on the joint use of numerical and symbolic techniques to extract the relevant passages from the source text. The discourse symbolic analysis consists of a rhetorical analysis which aims to highlight, in a source text, the minimal units (*central* units) necessary for the extraction process. When no rhetoric relation is attributed to some units, the relevance of these units is determined by a learning mechanism based on the Support Vector Machine (SVM) algorithm.

### 12.5 Multiple Document Summarization Approaches

Research works on multiple document summarization should consider aspects of connectivity between the source documents, as well as aspects of redundancy, consistency, cohesion and organization of extract sentences. The simple approach

of concatenating the source documents into one document and then applying single document summarization methods is insufficient. Thus, several works on multiple document summarization propose specific methods that reflect document multiplicity.

We survey in this section the current state of the art for multiple document summarization. As done for single document summarization, we classify the proposed works in three main approaches: numerical, symbolic and hybrid.

### ***12.5.1 Numerical Approach***

The numerical approach for multiple document summarization regroups methods based on statistical and training techniques to produce extracts. These techniques are generally applied in order to select, in a first stage, the important sentences and avoid, in a second stage, the information redundancy. Most techniques, used for the selection stage, are inspired by research works proposed within the single document framework, whereas similarity techniques are applied for selected sentences to ensure the reduction of extract redundancy.

#### **Statistical Methods**

Most statistical methods use metrics which inform about the similarity between the source document textual units in order to create groups (clusters) of similar units or to avoid the redundancy if the clustering is not carried out. They also use statistical criteria to measure the importance of these units in order to classify them. We present below some statistical methods.

- Methods based on the application of an MMR (Maximum Marginal Relevance) metric proposed by Goldstein et al. [38]. Lin and Hovy [59] have combined MMR with five heuristics: sentence position, term frequency, concept signature, term clustering, and tense indicators. Mori et al. [80] have used MMR combined with an IGR (Informational Gain Rational) metric.
- Methods based on the use of the centroid concept introduced by Radev [88] to indicate the keywords of a collection. Sekine and Nobata [93], Erkan and Radev [33], Goldensohn et al. [15] have also used the centroid concept.
- Methods based on the Multiple Sequence Alignments (MSA) technique proposed by Lacatusu et al. [56] and started by Barzilay and Lapata [10], and on the algebraic and matrix techniques (e.g. the Latent Dirichlet Allocation method [8]).

## Training Based Methods

Many researchers have used training based methods for multiple document summarization: Conroy et al. [21] combine the MMR metric with a Hidden Markov Model, Vanderwende et al. [100] use training mechanisms for user query expansion. In the same way, Dunlavy et al. [28] proceed by question classification and apply the Hidden Markov Model which uses the word signature.

From the same point of view, training is used to infer similarity measurement between the sentences and the query words. Jagadeesh et al. [49] proposed to use a probabilistic model combined with an entropy measurement. Amini and Usunier [6] proposed a classification allowing determining the importance of a sentence class by comparing it to the query words. Within the framework of biographical summaries, Biadsy et al. [14] proposed the use of a classifier to determine the most important sentences of a text.

### 12.5.2 *Symbolic Approach*

The multi-document summarization methods adopting the symbolic approach use linguistic treatments and analyses which, according to the considered task and to the used resources, can be described as shallow or deep.

#### Shallow Analysis Based Methods

Shallow analysis based methods use linguistic indicators to detect the relations between the words or the sentences of the source documents. These relations make it possible to deduce the importance of the sentences, their similarity or dissimilitude and facilitate their classification or clustering.

In this framework, Filatova and Hatzivassiloglou [34] and Nobata and Sekine [82] proceeded by recognition of the named entities as well as the relations which they maintain. Zhou et al. [104] determined the category of the sentences resulting from articles describing human biographies. To summarize blogs, Ku et al. [54] carried out an analysis based on three word lists expressing the positive, negative and neutral opinions.

The linguistic indicators detected in document sentences can also be exploited to identify the textual units that can instantiate predefined summaries templates. This method was initially applied within the framework of simple document summarization (i.e. [31, 97]). It has also been used for multiple document summarization: Radev and McKeown [89] proposed the SUMMONS (SUMMarizing Online NewS) system for summarizing newspapers describing terrorist acts. White et al. [102] proposed instantiating the templates with information extracted from the source newspapers describing natural disasters. In the same way, Daumé et al. [24] proposed to use summary templates decomposed into two parts: the first part

describes the main event and the second one presents the event details. Harabagiu et al. [42] proposed to use the difference between the source articles to instantiate templates (implemented in the GISTEXTER system). Their method produces an ad hoc template when the system predefined template base is inadequate to the field of the document collection.

### Deep Analysis Based Methods

Deep analysis based methods use the discourse structure and the semantic and pragmatic structures of the source texts, in order to determine the rhetorical, semantic and pragmatic relations between the various units which compose them. These relations are then exploited on the clustering process level (similar units) as well as that of relevant unit selection (with strong connectivity). The main research works can be classified into three methods: the comprehension based method, the rhetorical structure based method and the lexical and syntactic analysis based method.

- Comprehension based methods: The comprehension based methods require the development of conceptual models and robust linguistic and data-processing tools. In this context, Mani and Bloedorn [69] based their work on Sowa conceptual graphs, Melli et al. [77] identified the semantic dependences (graphs) between the source texts on the one hand, and the user query on the other hand.
- Rhetorical structure based methods: The discursive analysis allows to explore the rhetorical relations between the sentences and the source documents. Thus, Radev [87] proposed an extension of the rhetorical structure theory called CST (cross structure theory). They added rules to detect the similarity/dissimilarity between the documents, the paragraphs and the sentences.
- Lexical and syntactic analysis based methods: The lexical and syntactic analyses allow identifying the key concepts, and their syntactic relations. Fuentes et al. [35] exploited the lexical chains, the co-reference chains and the named entity chains to extract the cohesion forms which connect the sentences of the source texts. The lexical chain technique was also used by Ercan [32]. Afantinos et al. [1] focused on the event evolution in the source documents and proposed the detection of the synchronic and diachronic relations between the described events. This detection is based on an ontology which exploits the syntactic and semantic characteristics of analyzed events.

#### 12.5.3 Hybrid Approach

The hybrid approach regroups methods that combine numerical and symbolic techniques. Note that this combination is often motivated either by the low quality of the generated summaries, or by limitation of the application fields of the initial

methods. Within the framework of the hybrid methods, we cite the one proposed by McKeown et al. [76]. This method is based on a linguistic and statistical analysis to determine the summary sentences which convey similar topics. These sentences are then ordered according to the publication date of their source articles and are combined according to grammatical rules. It should be noted that this method was implemented in the MultiGen system which handles documents treating the same event.

Hatzivassiloglou et al. [43] have proposed a method which organizes similar small textual units (sentences or paragraphs) in distinct groups. It allows, based on a selection and generation module, to reduce each of these groups to only one sentence.

Blair-Goldensohn et al. [15] used the rhetorical structures adopted by Radev to judge the belonging of a sentence to a summary. They used, within this framework, statistical criteria such as the centroid concept as well as lexical criteria to determine the important sentences.

In the same context, Ou et al. [84] proposed a hybrid method which combines syntactic, discursive and statistical techniques in order to automatically produce scientific summaries in the field of sociology. The final summary represents a framework describing four types of information namely: research concepts, interlink concepts, contextual relations and research methods.

In the context of query focused multi-document summarization, Yeh et al. [103] proposed to evaluate the relevance of a sentence to the query by combining similarities calculated from the vector space model and latent semantic analysis. They also proposed a modified Maximal Marginal Relevance metric in order to reduce redundancy by taking into account shallow feature salience.

Jaoua et al. [51] proposed a method which considers the extract as the minimal extraction unit. Extracts built from combination of the source document sentences are evaluated and then classified. The classification uses a set of criteria aiming to maximize the coverage of keywords and to minimize the information redundancy in the extract. The generation and the classification of extracts have been implemented by using a genetic algorithm. Jaoua et al. also highlighted the importance of using a multi-objective optimization strategy to classify the extracts. To improve the structural quality of the produced extracts, they proposed the integration of a revision step to handle the extract sentences reordering. They have also integrated compression mechanisms to further enhance the linguistic quality of the final extracts. The compression was based on a set of compression rules and on a parsing process.

Celikyilmaz and Hakkani-Tur [19] proposed a hybrid multi-document summarization method which considers the extraction as a two-step learning problem. The first step consists of building a generative model for pattern determination and the second uses a regression model for inference. They used a hierarchical topic model to determine the latent characteristics of sentences in document clusters and then to calculate their scores. Based on these scores, they trained a regression model. This model is based on the lexical and structural characteristics of the sentences. Finally, they used this model to determine new sentences which are candidates to form a summary.

## 12.6 Case of Semitic Languages

We survey in this section the current state of the art on automatic summarization of Semitic languages. While much effort has been put into the development of summarization systems for processing Modern Standard Arabic (MSA) and Hebrew, for other Semitic languages the development of such tools lags behind.

As introduced in Chap. 1, the processing of texts written in Semitic languages poses many challenges. Indeed, Semitic language scripts differ from Indo-European language scripts. Several linguistic theories, and, consequently, computational linguistic approaches, are often developed with a narrow set of (mostly European) languages in mind. The adequacy of such approaches to other families of languages is sometimes unrealistic. Moreover Semitic texts (except Maltese ones) are written from right to left whereas the usual treatments handle the texts from left to right. The non-adequacy of European language approaches to the case of Semitic languages also applies to automatic summarization when based on symbolic approaches (i.e., approaches that rely on grammars or where deep processing is needed, etc.). However when only shallow processing is used, we could assume that approaches proposed for European languages could work for any languages, among them Semitic languages.

In this context, a multilingual task was introduced in TAC'2011 to promote multi-document summarization approaches that are language independent. Each system which participated in the task was called to provide summaries for different languages. Arabic and Hebrew are among the languages that have been introduced in this task.

In the following, we first present the main automatic summarization systems that have participated in TAC'2011 and that are language independent. Then, we present automatic systems that are language dependent.

### 12.6.1 Language-Independent Systems

Livak et al., proposed Muse system which implements a supervised language-independent summarization approach based on optimization of multiple statistical sentence ranking methods [62]. This system extends the initial system proposed for Hebrew and English languages and uses the same method to process Arabic documents. The authors concluded that the same weighting model is applicable across multiple languages to rank important sentences.

Hmida et al. use MMR modified measure to rank sentences for different languages [44]. In order to reduce the dependence to language, the authors ignore the notion of word and use n-grams of characters as tokens to represent sentences. The language-independent MMR system (MMR-IND) was compared to the language-dependent MMR topline (MMR-DEP). Evaluations had been established for all languages provided by the TAC'11 multilingual corpus. For Semitic languages,

results showed that for Hebrew, MMR-DEP is worse than MMR-IND while for Arabic it is better.

J. Steinberger et al. proposed a language-independent summarization system that applies a singular value decomposition (SVD) on a term-by-sentence matrix [96]. In the initial used matrix, the only resource which is language dependent was a list of stop-words. The SVD determines the latent (orthogonal) dimensions, which in simple terms correspond to the different topics discussed in the source document. The author uses the resulting matrixes to identify and extract the most salient sentences. This system was implemented first for English. The TAC'2011 multilingual task allowed testing the system on other languages. For Semitic languages, the system was ranked at the top for Hebrew and it was lower than baseline for Arabic.

Conroy et al. proposed a system that ranks sentences by estimating the probability that a term will be included in a human-generated summary [22]. The probability is based on the term signature, and its co-occurrence on the corpus. The authors chose a non-redundant subset of high scoring using non-negative matrix factorization to minimize information redundancy in the summary. First, the system was used for English. This system was experimented with other languages (Arabic and Hebrew) in the multilingual summarization pilot for TAC'2011.

In the rest of this section, we present some automatic summarization systems that are language dependent. We focus only on Arabic, Hebrew, Maltese and Amharic because for the other Semitic languages we did not find any work that is directly or indirectly related to automatic summarization.

## 12.6.2 Arabic Systems

Most summarization systems developed for Semitic languages are based on shallow processing. The use of such techniques (i.e., statistical methods, machine learning, etc.) can be explained by the lack of NLP tools necessary for deep processing (parsers, morphological analyzers, etc.).

We present in this section the main Arabic summarization systems.

The Lakhas System [27] is an Arabic automatic summarization system of single documents. This system extracts the relevant sentences (i.e. sentences to be included in the summary) according to their weights. Lakhas is intended to generate very short summaries (headlines). To produce an English summary, Lakhas first summarizes the original Arabic document and then applies machine translation to produce the English summary. The advantage is to avoid problems coming from poor translations. Lakhas addresses multiple difficulties related to Arabic sentence splitting, tokenization, and lemmatization. The scoring function is based on the sentence position in the document, the number of subject terms (i.e. words that appear in the headlines) in the sentence, the number of *indicative words* in the document, and the tf.idf value of each word in the sentence. This system was

evaluated within the framework of the DUC'2004 conference which was about summarization of documents translated from Arabic to English.

Sobh et al. [95] have proposed an automatic summarization system for Arabic which integrates Bayesian, genetic programming (GP) and classification methods to extract the summary sentences. The proposed system is trainable and uses a manually annotated corpus. It is based on a statistical classification which classifies a sentence (i.e. as relevant or not) according to its feature vector. The importance of a sentence is determined by various heuristics such as position, cue phrases, word/phrase frequency, lexical cohesion, discourse structure and indicator phrases. GP is used for the optimization to obtain the best feature combination that classifies sentences. We should note that the system requires annotated training and testing corpora.

The AQBTSS system [30] is a query-based single document summarization system that accepts, as input, an Arabic document and a query (also in Arabic) and provides, as output, a summary. AQBTSS is based on learning techniques. Indeed, the authors adapted the traditional vector space model (VSM) and the cosine similarity to find the most relevant passages of the source text and then produce the text summary.

Schlesinger et al. [92] proposed CLASSY, an Arabic/English query-based multi-document summarizer system. They used an unsupervised modified k-means method to cluster multiple documents into different topics. They relied on the automatic translation of an Arabic corpus into English. The quality of the summaries was not high enough because of tokenization and sentence-splitting errors.

Alrahabi et al. [3] have proposed an automatic summarization system based on the contextual exploration method. This method uses linguistic knowledge and linguistic markers to determine relevant information such as thematic segments, definitional utterances and conclusive utterances.

Al-Sanie [5] has proposed a system that extends RST for Arabic language to build a discourse infrastructure for Arabic text summarization. The discourse level experimented in this work is based on rhetorical relations. The author identified a set of 23 rhetorical relations and when the relations between the text units are extracted, a full text representation is built in a binary tree structure called a schema or RS-tree. The summary consists of the top level nucleus of the rhetorical tree.

Maaloul et al. [64] and Bouadabbous et al. [16] proposed a method that combines numerical and symbolic techniques to produce extracts for single Arabic documents. The numerical features are determined by a learning technique based on the SVM algorithm. Symbolic features introduced in this work are based on the Rhetorical Structure Theory technique [71] and uses discourse knowledge. The method consists in determining the rhetorical relations (nucleus-satellite) between the minimal units of the text. After building a rhetorical relations tree, a filtering step is used to keep only important rhetorical relations. Note that this summarization method will be described in more detail in Sect. 12.7.

### ***12.6.3 Hebrew Systems***

For Hebrew, the first work in the field of automatic summarization is probably the one done by Hacohen et al. [39] for law articles. The summarization is done by extraction of the most relevant sentences. The developed system combines multiple statistical methods. To evaluate the weight to attribute to each method, the authors experimented with many machine learning algorithms such as: perceptron learning, Naive Bayesian learning, and genetic algorithm. The best results were achieved by the genetic algorithm.

Recent work has been done by Litvak et al. [61] who have developed a bilingual system (for Hebrew and English languages). The system uses a genetic algorithm to find the best linear combination of 31 sentence scoring metrics based on vector and graph representations of text documents (for both languages). Experimental results show that the system performances for both languages are quite similar.

### ***12.6.4 Maltese Systems***

For Maltese language there are few works on NLP in general and on automatic summarization in particular. Indeed, development of computational linguistic tools and resources for Maltese is as yet in its infancy. Some projects to develop such resources and tools for Maltese are in progress.

One of these projects could be found on the Maltese Language Resource Server (MLRS) <http://mlrs.research.um.edu.mt>. The Maltese Language Resource Server is a project coordinated by the Institute of Linguistics and the Department of Intelligent Computer Systems at the University of Malta whose primary aim is to create and make available basic language resources for the Maltese language, as well as develop natural language processing tools for Maltese. The project has benefited from funding from the Malta Council for Science and Technology (MCST) and the University of Malta.

Some information about current projects on a written corpus and a dictionary of Maltese are available at <http://um.edu.mt/linguistics/research/mlrs>. Other information about the SPAN (Speech Annotation for Maltese Corpora) project which aims at creating an annotated corpus of spoken Maltese could be found at <http://um.edu.mt/linguistics/research/span>.

To our knowledge, the only work that may have some connection to automatic summarization is [101]. The connection of this research work to Maltese is indirect: although the summarization system was developed for Maltese property sales contracts under Maltese law, it is based on a corpus of documents in the English language (i.e. Maltese is the national language of Malta, and a co-official language of the country alongside English). According to [101] a version of the system for Maltese documents could be straightforwardly developed but it has not yet been done.

### 12.6.5 Amharic Systems

Amharic is still an under-resourced language (as are most Semitic languages). Indeed, very few computational linguistic resources has been developed for Amharic and very little has been done in terms of NLP applications for this language. However, some research works have tried to develop computational linguistic resources for Amharic as rapidly as possible in order to create new ones as well as to extend and refine the resources themselves [36].

All available resources for Amharic could be downloaded from the resource portal for the Amharic NLP/CL community ([nlp.amharic.org](http://nlp.amharic.org)). Among them we can cite: the Amharic manually annotated corpus of 210,000 words and the AMMORPHO morphological stemmer. One should note that these resources are still fairly basic and further extensions are quite clearly needed.

To our knowledge there is no work that has been done on the automatic summarization field for Amharic. The only NLP research works that we found mainly address information retrieval, machine translation and speech recognition.

## 12.7 Case Study: Building an Arabic Summarization System (L.A.E)

We present in this section the *L.A.E* / *اللخاصل الآلي* system, an automatic summarizer for Arabic [53]. L.A.E is based on an original hybrid approach that combines numerical and symbolic techniques to extract the relevant text passages [66]. The symbolic technique relies on a rhetorical analysis which aims to identify the minimal units of the text and determine the rhetorical relations between them.

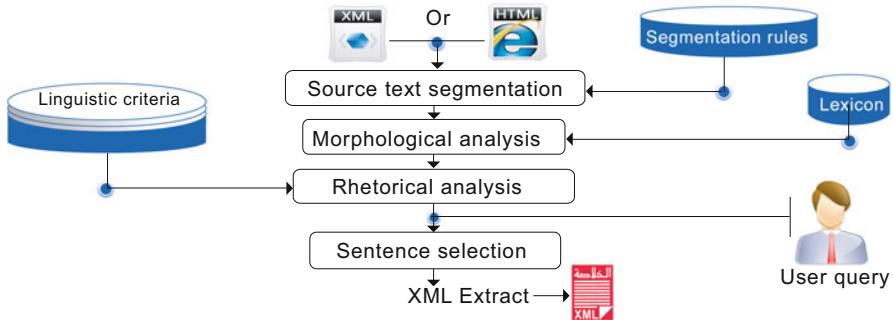
The numerical technique is based on supervised machine learning and uses the Support Vector Machine (SVM) algorithm. This technique allows to decide whether a sentence having certain rhetorical relations is relevant or not for the final extract (in particular, the rhetorical relation *Others* - آخر which is assigned when no other rhetorical relation is determined).

The learning mechanism relies on a training corpus of 185 news articles collected from the Web<sup>2</sup> without any restriction concerning their theme, content or volume. They have been manually annotated, by two human experts, to determine the rhetorical relations and their characteristics.

To our knowledge there is no other annotated corpus that could be used for Arabic automatic summarization. Hence, the authors constructed their own corpus. The training corpus size is relatively small. This is due to the fact that the construction of such corpora is a hard and expensive task.

---

<sup>2</sup><http://www.daralhayat.com>



**Fig. 12.3** L.A.E general architecture [17]

In the rest of this section, we present the L.A.E general architecture and its implementation details. Then, we describe some aspects of its use and present the evaluation results based on an original corpus, manually annotated in the context of this research work.

Note that each Arabic example given in this section will be followed by its transliteration using the Buckwalter system<sup>3</sup> and its English translation.

### 12.7.1 L.A.E System Architecture

The L.A.E system architecture is based on the proposed hybrid approach that combines symbolic and numerical techniques (rhetorical analysis and machine learning). It is composed of four main steps (see Fig. 12.3): source text segmentation, morphological analysis, rhetorical analysis and sentence selection.

One advantage of the hybrid approach is its ability to produce a dynamic summary according to the user query. Another advantage is the possibility of determining whether a sentence is relevant or not even though the rhetorical analysis fails in determining its rhetorical relation.

### 12.7.2 Source Text Segmentation

The objective of this step is to segment the source text (which could be a HTML or XML file) into smaller textual units: titles, sections, paragraphs and sentences. The segmentation relies on a rule-based approach proposed by Belguith et al. [12] to segment non-vowelled Arabic texts into paragraphs and sentences.

<sup>3</sup><http://www.qamus.org:80/transliteration.htm>

Let X, a splitting marker.	
IF	the left context of X is L
AND/OR	the right context of X is R
THEN	take the decision D (segment or not segment) with a success rate of n %.

**Fig. 12.4** Segmenting rule structure

Left context		Marker	Right context		
	Verb	و [w]	Space(s)	Noun	
IF The coordination conjunction و [w] (et) is preceded by space					
AND the Right context و of [w] (et) is a verb					
AND the Left context of و [w] (et) starts with a noun					
THEN و [w] (et) indicates the beginning of a new sentence with a success rate of 77%.					

**Fig. 12.5** A rule corresponding to the coordination conjunction و [w] (and)

The proposed approach consists in a contextual analysis of the punctuation marks, the coordination conjunctions and a list of particles that are considered as sentence boundaries. One hundred eighty-three rules have been proposed to segment Arabic texts into paragraphs and sentences [13]. Figure 12.4 shows the general structure of these segmenting rules.

The segmenting rules are classified into three main classes according to the considered splitting marker type (i.e. a punctuation mark, a coordination conjunction or a particle). Figure 12.5 shows an example of a rule corresponding to the coordination conjunction و [w] (and).

Example 12.1 shows that when applying the rule of Fig. 12.5 on text (T1), this latter will be segmented into two sentences: (S1) and (S2).

*Example 12.1.*

(T1)

ولد هذا العالم في مصر وحفظ القرآن الكريم في سن العاشرة من عمره.  
 [wld h\*A AIEAlm fy mSr wHfZ Alqr|n Alkrym fy sn AIEA\$rp mn Emrh]  
 This scientist was born in Egypt, and he learned the Holy Quran at the age of 10.

(S1)

ولد هذا العالم في مصر  
 [wld h\*A AIEAlm fy mSr]  
 This scientist was born in Egypt

**Table 12.1** HTML tag set used to segment *Dar al Hayet* corpus

HTML tags	Role
<P> and </P>	Mark the beginning/end of a paragraph
<DIV> and </DIV>	
<SPAN> and </SPAN>	
<font style = "font-family : Simplified Arabic, AB Geeza, Times New Roman, Times; color : Black; font-size:15pt;"> and </font>	Mark the beginning/end of a title
<fontstyle = "font-family : Simplified Arabic, AB Geeza, Times New Roman, Times; color : Black; font-size : 13pt;"> and </font>	Mark the beginning/end of a title

(S2)

وحفظ القرآن الكريم في سن العاشرة من عمره.

[w HfZ Alqrln Alkrym fy sn AIEA\$rp mn Emrh]

and he learned the Holy Quran at the age of 10.

In addition to the rule set described above, the segmentation is also based on a set of HTML tags that guides the segmentation process (i.e. in case of HTML documents). Note that the HTML tag set could relatively change from one corpus to another.

Table 12.1 shows the HTML tag set used for the *Dar al Hayet* corpus (i.e. news articles available on the *Dar al Hayet* magazine website).

The segmentation step provides, as output, an XML file containing the source text enriched with XML tags that indicate titles, sections, paragraphs and sentences boundaries (see Fig. 12.6).

## Morphological Analysis

The main goal of the morphological analysis is to determine, for each word, the list of its possible morpho-syntactic features (part of speech (POS), tense, gender, number, etc.).

The L.A.E system uses the morphological analyzer MORPH-2 [13]. MORPH-2 is based on a knowledge-based computational method to generate for each word its possible POS (noun, adjective, verb, possessive pronoun, demonstrative pronoun, preposition, etc.) and its morphological features namely, the gender, the number, the tense, the personal pronoun, the determination (determined/undetermined) and the semantic feature (human/non-human) [11]. MORPH-2 operates on five steps. It accepts an Arabic text (which can be not vocalized, partially vocalized or fully vocalized). A tokenization process is applied in the first step. Then, the system determines, in a second step, for each word, its possible clitics. An affixal analysis is then applied to determine all possible word affixes and roots. The fourth step consists of extracting the word morpho-syntactic features according to the valid affixes and root. Finally, a vocalization step is applied in the case of non-vocalized

```

<?xml version="1.0" encoding="utf-8" ?>
- <text>
<title>نماح السرطان بالإنكار</title>
- <section>
- <paragraph>
<sentence><؟ قلت أسائل السياسيين العراقيين عن مخاطر غرب العراق في الغوضى بفعل الفراغ الذي قد يخلفه إغلاق نظام صدام حسين</sentence>
<sentence><؟ سألت تحديداً عن احتمال اضطراب العلاقة بين المكونين السنوي والشعبي</sentence>
<sentence><؟ أستقبل السؤال بقدر غير قليل من الاستغراب والإنزعاج</sentence>
<sentence><؟ أستطيع إيجاز الأوجية على الشكل الآتي</sentence>
<sentence><؟ تخطي إذا فرأت المشهد العراقي بعيون لبنانية</sentence>
<sentence><؟ نحن لدينا مشكلة وحيدة اسمها صدام حسين</sentence>
<sentence><؟ لا جذور عندنا للمشاعر الطائفية والمذهبية</sentence>
<sentence><؟ تعترفي العائلات الكبرى على جناح شيعي وآخر سني</sentence>
<sentence><؟ التزاوج طبيعي والتداخل عميق في بغداد وخارها</sentence>
<sentence><؟ لا يخطر ببال العراقي أن يسأل إذا كان مجده شيعياً أم سنياً</sentence>
<sentence><؟ ليتك تبقينا بعيدين من الأمراض الوبائية</sentence>
<sentence><؟ المشاعر القومية عميقة لدى العراقيين وهي تقدم على المشاعر الأخرى</sentence>
<sentence><؟ العسكريون الشيعة قاتلوا ضد إيران على امتداد سنوات الحرب</sentence>
</paragraph>
+ <paragraph>
+ <paragraph>
+ <paragraph>
+ <paragraph>
+ <paragraph>
</section>
</text>

```

**Fig. 12.6** Example of a segmentation step output

or partially vocalized texts [20]. MORPH-2 uses a lexicon of 5,754 trilateral and quadrilateral roots to which all corresponding verbal and nominal schemas are associated. The combination of roots and verbal schemas provides 15,212 verbal stems. The combination of roots and nominal schemas provides 28,024 nominal stems. A set of computing rules is also stored in the lexicon. An interdigitation process between the verbal schemas and the appropriate rules provides 108,415 computed nouns [20].

The L.A.E morphological analysis step generates, as output, a text in XML format which consists of the segmentation step output enriched with tags indicating the words' morpho-syntactic features that are generated by the MORPH-2 system (see Fig. 12.7).

## Rhetorical Analysis

The rhetorical analysis has a double objective; first, binding two adjacent minimal units together, of which one has the status of nucleus (central segment) and the other has the status of nucleus or satellite (optional segment); and second, determining the rhetorical relations between the various juxtaposed minimal units of the paragraph.

**Fig. 12.7** Example of output of the morphological analysis step

```

<?xml version="1.0" encoding="utf-8" ?>
- <text>
<title>نماح السرطان بالإنكار</title>
- <section>
- <paragraph>
<sentence></sentence>
- <lexical_unit>
<unit>كانت رياح الغزو الأمريكي تذمر بالهيبوب.</unit>
<pos>اسم</pos>
<type>فاعل</type>
<tense>ماضي</tense>
</lexical_unit>
- <lexical_unit>
<unit>الغزو</unit>
<pos>اسم</pos>
<type>مصدر</type>
</lexical_unit>
- <lexical_unit>
<unit>الأمريكي</unit>
</lexical_unit>
- <lexical_unit>
<unit>تذمر</unit>
<pos> فعل</pos>
<type>غير ناسخ</type>
<tense>مضارع</tense>
</lexical_unit>
- <lexical_unit>
<unit>بالهيبوب</unit>
<pos>اسم</pos>
<type>مصدر</type>
</lexical_unit>
+ <sentence>
.....
.....
</section>
</text>
```

Based on a study corpus (i.e. 500 newspaper articles), 20 rhetorical relations have been manually determined by two human experts (see Fig. 12.8).

The determination of a relation is done by applying a set of constraints (rhetorical frames) on the nucleus, the satellite and the combination of the nucleus and the satellite. Thus, a rhetorical frame (see Fig. 12.9) consists of linguistic markers. These markers are classified into two types: linguistic indicators and complementary indexes [78]. Linguistic indicators announce important concepts that are relevant to the summarization task.

The complementary indices are looked for in the neighbouring indicator. Thus, according to the context, they can confirm or reject the rhetorical relation announced by the releasing indicator.

Example 12.2 illustrates a sentence extracted from one of the news articles of the study corpus. This sentence contains a *Specification - تفصیل* relation between the first minimal unit (1) and the second minimal unit (2).

"Others - آخر"	"Exception - إستثناء -"	"Concession - استدراك -"	"Explanation - تفسير -"
"Condition - شرط"	"Weighting - ترجيح -"	"Enumeration - تفصيل -"	"Classification - ترتيب -"
"Reduction - تقليل"	"Restriction - حصر -"	"Exemplification - عثيل -"	"Obviousness - قاعدة -"
"Joint - ربط"	"Definition - تعريف -"	"Confirmation - توكيده -"	"Negation - نفي -"
"Possibility - إمكان -"	"Justification - تعليل -"	"Specification - تخصيص -"	"Conclusion - استنتاج -"
"Assertion - جزم -"			

Fig. 12.8 Rhetorical relations list

Relation	Specification - تخصيص -
Constraint on (1):	Contains a complementary index ("but / بل", "not / لم", "no / لا")
Constraint on (2):	Contains the linguistic indicator ("in particular / لاسيما")
Position of the indicator:	In the middle
Minimal unit reserve:	(2)

Fig. 12.9 A rhetorical relation frame

*Example 12.2.*

(١) لكن أليبر قيسري لم يكن نزيل غرفته في ذلك الفندق فقط، بل كان أحد وجوه الشارع وبعض مقاهيها الشهيرة، (٢) لاسيما مقهى فلور الذي كان يقضي فيه ساعات وحيداً أو مع أشخاص عابرين.

[(1) lkn Olbyr qySry lm ykn nzyl grfth fy \*lk Alfdndq fqT, bl kAn OHd wjwh Al\$ArE wbED mqAhyhA Al\$hyrp,

(2) lAsymA mqhY “flwr” Al\*y kAn yqDy fyh sAEAt wHydA Ow mE O\$xAS EAbry.]

(1) Albert Kaisary was not only a resident in that hotel, **but** he was also one of the famous street and people cafes,

(2) **In particular** the cafe Floor in which he spends hours alone or with passengers.

A *Specification - تخصيص -* relation has generally the role of detailing what is indicated and confirming its meaning and clarifying it. The specification frame (see Fig. 12.9) is used to detect the *Specification - تخصيص -* rhetorical relation.

Empirical studies done on the corpus showed some ambiguities in determining the correct rhetorical relation between two adjacent segments.

*Example 12.3.*

(١) وفي هذا الحال الذي لا يسر، كانت ثمة تحذيرات من قبل جنرالات سابقين أن المصير السياسي العربي قد (٢) يذهب إلى مآلات خطيرة...

[(1) wfy h\*A AlHAI Al\*y lA yrs, kAnt vmp tH\*yrAt mn qbl jnrAlAt sAbqyn On AlmSyr AlsyAsy AlErby qd (2) y\*hb IIY mllAt xTyrp... ]

Relation	"Weighting - ترجيح"
Constraint on (1):	Contains the indicator ("فقط", "قد", "وقد")+ a verb in the future tense
Constraint on (2):	-
Position of the indicator:	In the middle
Minimal unit reserve:	(1)

**Fig. 12.10** Rhetorical frame corresponding to Example 12.3

Relation	"Assertion - جزم"
Constraint on (1):	Contains the indicator ("فقط", "قد", "وقد")+ a verb in the past tense
Constraint on (2):	-
Position of the indicator:	In the middle
Minimal unit reserve:	(2)

**Fig. 12.11** Rhetorical frame corresponding to Example 12.4

(1) In these unfavorable conditions, there are warnings from the general army that the Arabic politics in future **may** (2) lead to serious consequences...

*Example 12.4.*

(1) مسؤول رفيع من النادي الملكي قد (2) ذهب ألى فرنسا و راقب نجم ليون  
كريم بن زيماء

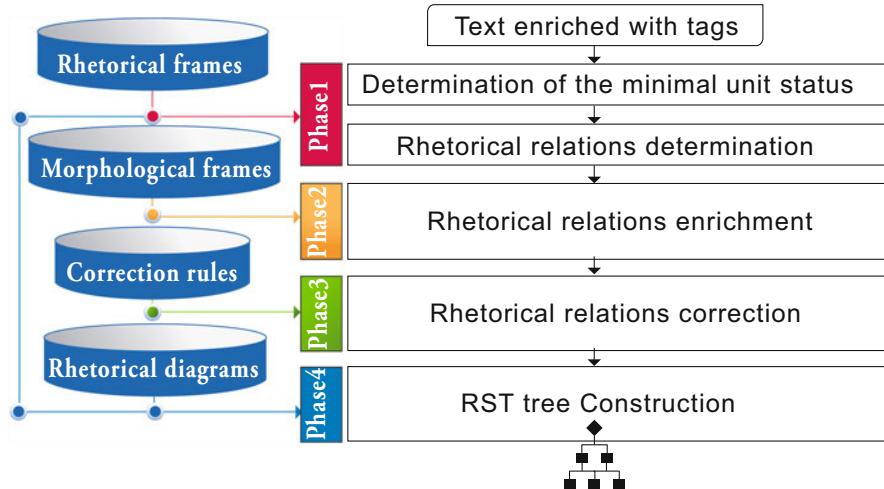
[(1) msWwl rfyE mn AlnAdy Almlky qd (2) \*hb AIY frnsA w rAqb njm lywn krym bn zymA...]

(1) A leader from the royal team (2) **may** have gone to France to observe' the Lyons' star Karim Benzema...

Examples 12.3 and 12.4 show that the releasing indicator **may**/قد provides two different rhetorical relations: *Weighting - ترجح* and *Assertion - جزم*. Due to the absence of complementary indexes in the indicator context, one could not confirm the concept announced by the indicator. Thus, to solve this ambiguity, Maaloul et al. [67] propose to use some morphological features (i.e., word POS, verb tense) of the words occurring immediately after the indicators.

The rhetorical frames corresponding to Examples 12.3 and 12.4 are presented in Figs. 12.10 and 12.11.

Moreover, to ensure the construction of a unique RST tree Keskes et al. [53] propose an enrichment phase of the rhetorical relations. This enrichment is based on morphological frames and correction frames. Figure 12.12 shows the different phases of the rhetorical analysis step. The first phase consists of determining the status of the minimal units (i.e. central segment or optional segment) and their rhetorical relations. This phase is based on a shallow analysis and more precisely



**Fig. 12.12** Rhetorical analysis phases

Let's consider S, a sentence.	
IF	S contains a rhetorical relation {Negation / نفي } followed by the index {because / لأن }
THEN	The rhetorical relation {Negation / نفي } is replaced by the {Explanation / تفسير } relation
AND	The minimal unit preceding the index {because / لأن } is considered as the nucleus (central segment)
AND	The minimal unit following the index {because / لأن } is considered as the satellite (optional unit).

**Fig. 12.13** Example of a correction rule (index-relation type)

on the contextual exploration technique [25]. It uses rhetorical frames based only on linguistic criteria (releasing indicators and complementary indexes).

The second phase consists of an enrichment of the detected rhetorical relations. It applies rhetorical frames which are based on morphological criteria in order to deduce other relations not identified during the first phase.

The third phase aims to correct the obtained rhetorical relations. It applies a set of correction rules on the set of rhetorical relations determined in the preceding phases. Note that the correction rules are of two types: index-relation or relation-relation. Figures 12.13 and 12.14 present two examples of these rules.

The fourth phase consists in producing the RST tree, judged to be the most representative of the text source structure. The RST tree construction is based on the five schema types proposed by Mann and Thompson [71]. It is also based on 63 rhetorical rules proposed by Keskes et al. [53].

Let's consider S, a sentence.	
IF	S contains a rhetorical relation {Confirmation / توكيد / نفي} followed by a rhetorical relation {Negation / نفي}
THEN	The two rhetorical relations are replaced by the {Affirmation / جزم} rhetorical relation
AND	The minimal unit preceding the releasing indicator of the {Negation / نفي} rhetorical relation is considered as the nucleus (central segment)
AND	The minimal unit following the releasing indicator of the {Negation / نفي} rhetorical relation is considered as the satellite (optional unit).

**Fig. 12.14** Example of correction rule (relation-relation type)

Extract type	List of retained rhetorical relations
Indicative extract	"Confirmation - توكيد", "Conclusion - استنتاج", "Assertion - جزم"
Informative extract	"Confirmation - توكيد", "Conclusion - استنتاج", "Assertion - جزم", "Justification - تعليل", "Reduction - تقليل", "Enumeration - تفصيل", "Weighting - ترجيح", "Possibility - إمكان"
Opinion extract	"Condition - شرط", "Concession - استدراك", "Exception - استثناء", "Confirmation - توكيد", "Obviousness - قاعدة", "Negation - نفي", "Conclusion - استنتاج", "Assertion - جزم", "Possibility - إمكان", "Restriction - حصر", "Justification - تعليل"

**Fig. 12.15** Retained rhetorical relations for the indicative, informative and opinion extracts

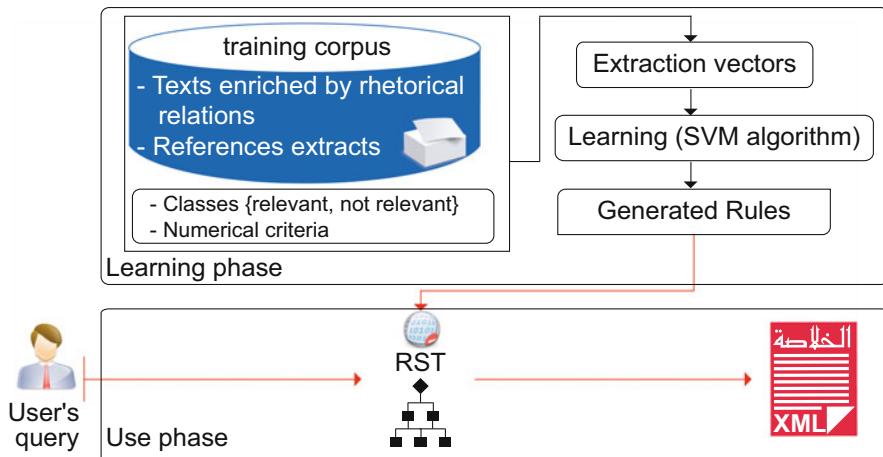
## Sentence Selection

The sentence selection step aims at identifying relevant sentences, candidates for the final extract. This extract is dynamic: it can be generated according to a particular type (indicative, informative, etc.) or according to a user query. Indeed, one user may prefer extracts focusing on definitional parts while another may be interested in extracts focusing on conclusive ones.

The sentence selection step is based on two main processes: the RST tree simplification process and the machine learning process.

**The RST tree simplification process:** it aims at pruning the RST tree by considering only interesting rhetorical relations according to the extract type. Indeed, for each extract type (i.e., indicative, informative, opinion, etc.), the human experts proposed a list of the most important rhetorical relations. Figure 12.15 shows the list of retained rhetorical relations for the indicative, informative and opinion extracts. Note that it is also possible for the user to choose the list of rhetorical relations.

**The machine learning process:** the rhetorical analysis step is not able to determine all rhetorical relations. Indeed, when no releasing indicator is detected in a sentence, it simply assigns the relation *Others* - آخر to it (i.e. the rhetorical relation *Others* - آخر is assigned when no other rhetorical relation is determined). In such a case,



**Fig. 12.16** Principle of supervised learning based systems

the machine learning process (i.e. more precisely supervised machine learning) is used in order to determine whether the sentence is relevant or not. Indeed, sentences having the relation *Others - آخر* could be pertinent. Ignoring those sentences when producing the final summary could negatively affect its quality. Thus, the machine learning process aims to determine among sentences connected by the rhetorical relations *Others - آخر* those which are relevant and consequently will be retained in the final summary.

The supervised machine learning process relies on two phases: the learning phase and the use phase (see Fig. 12.16).

**The learning phase:** it uses a training corpus composed of a set of source documents and their corresponding extracts. This phase is based on several numerical criteria useful for the generation of the SVM classification equation: sentence position in the text, sentence position in the paragraph, number of title words, number of bonus phrases, number of stigma phrases, anaphoric expressions, lexical co-occurrence, etc.

The training corpus is formed of 185 articles selected from the magazine *Dar al Hayat - دار الحياة*. The choice of the journalistic field is mainly justified by the important number of linguistic markers [65].

These articles cover several domains although they focus on some domains more than others (see Table 12.2). According to Boudabbous et al. [17], this is due to the fact that some domains present more diversity and richness in terms of events and temporal expressions (e.g. this is the case of the political event domain).

The training corpus was manually tagged by two human experts who selected the relevant sentences of each article. The reference summaries were generated manually after experts' agreement.

**Table 12.2** Training corpus details

Domain	Number of articles	Article size (sentence number)
Sport news	25	200
National news	45	1,800
Politics	50	2,800
Education	30	660
Science	35	490
Total	185 articles	5,950 sentences

**Fig. 12.17** A news article extracted from the training corpus

Figure 12.17 illustrates a news article extracted from the training corpus. The selected sentences are bounded by the symbol \$.

The **second phase** of the learning process, called the use phase, determines which sentences, having the rhetorical relations *Others - آخر*, are relevant to the final extract.

### 12.7.3 Interface

The L.A.E system is implemented using Delphi. Figure 12.18 shows the L.A.E system main interface. This system also integrates some tools, in particular the *STAR* tokenizer [12] and the *MORPH-2* morphological analyzer [11].

Figure 12.19 shows the user query interface. Many options are offered to the user and he can choose (i) the extract size in terms of a number of sentences or a reduction rate, (ii) the extract type (e.g. informative, indicative, opinion) or a personalized extract (i.e. in this case he can specify the rhetorical relation types to be considered when generating the extract).



Fig. 12.18 The L.A.E system interface



Fig. 12.19 User query interface

#### 12.7.4 Evaluation and Discussion

The test corpus consists of 100 newspaper articles extracted from *Dar al Hayet* magazine. This corpus is different from the training corpus and the study corpus presented above. For each test corpus article, there is a summary reference manually extracted by two human experts. A summary reference is a set of relevant sentences extracted, by the experts, from the source news articles.

A sentence is considered to be relevant in case both experts agree on it. In the other case, it is considered to be irrelevant unless it gets a kappa index greater than 0.68.

In order to show the efficiency of the proposed hybrid method, the L.A.E system was evaluated in two ways. The first evaluation was done by the L.A.E (V0) system

	Recall	Precision	F-measure
L.A.E / (v0) الألخاصل الآلي	0.21	0.32	0.21
L.A.E / (v1) الألخاصل الآلي	0.47	0.61	0.53

**Fig. 12.20** L.A.E. system evaluation results

which is based only on a purely symbolic approach (i.e. the sentence selection is based only on the RST analysis) and the second evaluation was done by the L.A.E (V1) system which is based on a hybrid approach (i.e., the sentence selection is based on the RST analysis and on the machine learning process).

Figure 12.20 shows that the L.A.E (V1) evaluation results are largely better than the L.A.E (V0) ones. Indeed, L.A.E (V0) obtained 0.21 for F-measure while the L.A.E (V1) F-measure reached 0.53. These results confirm that applying a hybrid approach could enhance the summarization system performance.

## 12.8 Summary

In this chapter we first described some key concepts related to the summarization task: task definition, summary types, general architecture of a summarizing system, extraction vs. abstraction, evaluation measures for summarization, the major known challenges, automatic summarization and insights from the evaluation campaigns.

Then, we have discussed the main approaches for both single and multiple document summarization. Advances in automatic summarization for Semitic languages and main related research works have been discussed. As a conclusion, we can say that the main challenging obstacle of these research works is due to (i) the characteristics of the Semitic languages such as the lack of short vowels, the absence of punctuation marks, relatively long sentences and rich morphology and (ii) the sparseness of resources (i.e., NLP tools and corpora) that might stymie building robust system for automatic summarization and encourage system evaluation on large corpora.

Finally, we have described a research work where a system for automatic summarization of Arabic texts has been developed. The main aim of this research work was to investigate an original hybrid approach to summarize Arabic texts. In order to achieve this endeavor numerical techniques were combined with symbolic ones. This approach has the advantage of being able to decide according to the source text and the symbolic technique results whether a learning approach is needed or not, in order to enhance the extract quality.

Our main goal was to offer a document that could help researchers, engineers and practitioners better understand the summarization task and the intricacies involved in building automatic summarization systems especially when it comes to morphologically rich/complex languages such as Semitic languages.

## References

1. Afantinos SD (2008) Summarizing reports on evolving events – part ii: non-linear evolution. In: Bernadette SE, Zock M (eds) Proceedings of the 5th international workshop on natural language processing and cognitive science (NLP-CS 2008), Barcelona, pp 3–12
2. Alemany LA, Castellón I, Climent S, Fort MF, Padró L, Rodríguez H (2004) Approaches to text summarization: questions and answers. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial* 8(22):79–102
3. Alrahabi M, Mourad G, Djouia B (2004) Filtrage sémantique de textes en arabe en vue d'un prototype de résumé automatique. In: dans les actes de la conference JEP/TALN'04, Fès
4. Alrahabi M, Djouia B, Desclés JP (2006) Annotation sémantique des énonciations en arabe. In: INFORSID'2006, Hammamet
5. Al-Sanie W (2005) Towards an infrastructure for Arabic text summarization using rhetorical structure theory. Master thesis in computer science, King Saud University, Riyadh
6. Amini MR, Usunier N (2009) Incorporating prior knowledge into a transductive ranking algorithm for multi-document summarization. In: Proceedings of the 32nd international ACM SIGIR conference on research and development in information retrieval, SIGIR'09, Boston. ACM, pp 704–705
7. Amini M, Tombros A, Usunier N, Lalmas M (2007) Learning-based summarisation of XML documents. *Inf Retr* 10(3):233–255
8. Arora R, Ravindran B (2008) Latent dirichlet allocation based multi-document summarization. In: AND, Singapore, pp 91–97
9. Barzilay R, Elhadad M (1997) Using lexical chains for text summarization. In: Proceedings of the ACL/EACL 1997 workshop on intelligent scalable text summarization, Madrid, pp 10–17
10. Barzilay R, Lapata M (2005) Collective content selection for concept-to-text generation. In: Proceedings of the conference on human language technology and empirical methods in natural language processing, HLT'05, Vancouver. Association for Computational Linguistics, pp 331–338. <http://dx.doi.org/10.3115/1220575.1220617>
11. Belguith LH, Chaaben N (2004) Implémentation du système morph2 d'analyse morphologique pour l'arabe non voyellé. In: Quatrièmes journées scientifiques des jeunes chercheurs en Génie Electrique et Informatique (GEI'2004), Monastir
12. Belguith LH, Baccour L, Ghassan M (2005) Segmentation de textes arabes basée sur l'analyse contextuelle des signes de ponctuations et de certaines particules. Actes de la 12ème conférence sur le Traitement Automatique des Langues Naturelles TALN'2005, Dourdan, vol 1, pp 451–456
13. Belguith LH, Aloulou C, Ben Hamadou A (2007) Maspar : De la segmentation à l'analyse syntaxique de textes arabes. In: CEPADUES-Editions (ed) Revue information interaction intelligence I3, vol 2, pp 9–36. ISSN:1630-649x, <http://www.revue-i3.org/>
14. Biadsy F, Hirschberg J, Filatova E (2008) An unsupervised approach to biography production using wikipedia. In: Association for Computational Linguistics, Columbus, pp 807–815
15. Blair-Goldensohn S, Evans D, Hatzivassiloglou V, McKeown K, Nenkova A, Passonneau R, Schiffman B, Schlaikjer A, Siddharthan A, Siegelman S (2004) Columbia University at DUC 2004. In: Proceedings of the document understanding conference, Boston, pp 23–30
16. Boudabbous MM, Maaloul MH, Belguith LH (2010) Digital learning for summarizing Arabic documents. In: Proceeding of the 7 th international conference on natural language processing, IceTAL'10, Reykjavik
17. Boudabbous MM, Keskes I, Maaloul MH, Belguith LH (2011) Automatic summarization of Arabic texts. In: 7th international computing conference in Arabic 2011 (ICCA2011), Riadh
18. Carbonell J, Goldstein J (1998) The use of MMR, diversity-based reranking for reordering documents and producing summaries. In: Research and development in information retrieval, Melbourne. Association for Computing Machinery, New York, pp 335–336
19. Celikyilmaz A, Hakkani-Tur D (2010) A hybrid hierarchical model for multi-document summarization. In: ACL, Uppsala, pp 815–824

20. Chaaben N, Belguith LH, Ben Hamadou A (2010) The morph2 new version: a robust morphological analyzer for Arabic texts. In: Actes des 10emes journées internationales d'analyse statistique des données JADT'2010, Rome. <http://jadt2010.uniroma1.it/>
21. Conroy JM, Goldstein J, Schlesinger JD, O'leary DP (2004) Left-brain/right-brain multi-document summarization. In: Proceedings of the document understanding conference DUC'04, Boston
22. Conroy JM, Schlesinger JD, Kubina J (2011) CLASSY 2011 at TAC: guided and multi-lingual summaries and evaluation metrics. In: Proceedings of TAC'11, Gaithersburg
23. Dang HT, Owczarzak K (2009) Overview of TAC 2009 summarization track. In: Proceedings of the second text analysis conference, Gaithersburg
24. Daumé H III, Echihabi A, Marcu D, Munteanu DS, Soricut R (2002) GLEANS: a generator of logical extracts and abstracts for nice summaries. In: Proceedings of the second document understanding conference (DUC), Philadelphia, pp 9–14
25. Desclés JP (1997) Systèmes d'exploration contextuelle. In: Co-texte et calcul du sens – (Claude Guimier). Presses universitaires de Caen, pp 215–232
26. Desclés J-P, Minel J-L (2005) Interpréter par exploration contextuelle. In: Corblin F, Gardent C (eds) Interpréter en contexte. Hermès, Paris, pp 305–328
27. Douzidia F, Lapalme G (2004) Lakhlas, an Arabic summarization system. In: Proceedings of DUC'04, NIST, Boston, pp 128–135
28. Dunlavy DM, O'Leary DP, Conroy JM, Schlesinger JD (2007) QCS: a system for querying, clustering and summarizing documents. Inf Process Manag 43(6):1588–1605. Text summarization
29. Edmundson HP (1969) New methods in automatic extracting. J Assoc Comput Mach 16(2):264–285
30. El-Haj M, Hammo B (2008) Evaluation of query-based Arabic text summarization system. In: Proceeding of the IEEE international conference on natural language processing and knowledge engineering, Beijing. IEEE Computer Society, pp 1–7
31. Ellouze M (2004) Des schémas rhétoriques pour le contrôle de la cohérence et génération de résumés automatiques d'articles scientifiques. Thèse de doctorat, Ecole Nationale des sciences de l'Informatique, Université de Manouba, Tunis
32. Ercan G (2006) Automated text summarization and keyphrase extraction. Phd thesis, Bilkent University
33. Erkan G, Radev DR (2004) Lexpagerank: prestige in multi-document text summarization. In: EMNLP, Barcelona
34. Filatova E, Hatzivassiloglou V (2003) Domain-independent detection, extraction, and labeling of atomic events. In: Proceedings of the RANLP'03 conference, Borovetz
35. Fuentes M, Massot M, Rodríguez H, Alonso L (2003) Headline extraction combining statistic and symbolic techniques. In: DUC03, Edmonton. Association for Computational Linguistics
36. Gambäck B, Asker L (2010) Experiences with developing language processing tools and corpora for Amharic. In: Cunningham P, Cunningham M (eds) Proceedings of IST-Africa 2010, the 5th conference on regional: impact of information society technologies in Africa, Durban. <http://www.sics.se/~gamback/publications/istafrika10.pdf>
37. Giannakopoulos G, Karkaletsis V, Vouros G, Stamatopoulos P (2008) Summarization system evaluation revisited: N-gram graphs. ACM Trans Speech Lang Process 5(3):1–5
38. Goldstein J, Mittal V, Carbonell J, Callan J (2000) Creating and evaluating multi-document sentence extract summaries. In: Proceedings of the ninth international conference on informationand knowledge management, McLean. ACM, New York, pp 165–172
39. HaCohen-Kerner Y, Malin E, Chasson I (2003) Summarization of Jewish law articles in Hebrew. In: Nygard KE (ed) Proceedings of the 16th international conference on computer applications in industry and engineering, ISCA, Imperial Palace Hotel, Las Vegas, pp 172–177
40. Hahn U (1998) Automatic extracting – a poor man's approach to automatic abstracting. In: International workshop on extraction, filtering and automatic summarization (RIFRA'98), Sfax

41. Hahn U, Mani I (2000) The challenges of automatic summarization. Computer 33(11)29–36. <http://dx.doi.org/10.1109/2.881692>
42. Harabagiu SM, Lacatusu VF, Maiorano SJ (2003) Multi-document summaries based on semantic redundancy. In: FLAIRS conference, St. Augustine, pp 387–391
43. Hatzivassiloglou V, Klavans J, Holcombe M, Barzilay R, Kan M, McKeown K (2001) SIMFINDER: a flexible clustering tool for summarization. In: Proceedings of the NAACL workshop on automatic summarization, Pittsburgh, pp 41–49
44. Hmida F, Favre B (2011) LIF at TAC multiling: towards a truly language independent summarizer. In: Proceedings of TAC'11, Gaithersburg
45. Hovy E (1999) Cross-lingual information extraction and automated text summarization. In: Multilingual information management: current levels and future abilities, chap 3. Istituti editoriali e poligrafici internazionali, Pisa
46. Hovy E, Lin CY (1999) Automated text summarization in summarist. In: Mani I, Maybury MT (eds) Advances in automatic text summarization. MIT, Cambridge
47. Hovy E, Marcu D (1998) Automated text summarization tutorial. In: COLING/ACL'98, Montreal
48. Hovy E, Yew Lin C, Zhou L, Fukumoto J (2006) Automated summarization evaluation with basic elements. In: Proceedings of the fifth conference on language resources and evaluation (LREC'06), Genoa
49. Jagarlamudi J, Pingali P, Varma V (2007) Capturing sentence prior for query-based multi-document summarization. In: RIAO, Pittsburgh
50. Jaoua M, Hamadou AB (2003) Automatic text summarization of scientific articles based on classification of extract's population. In: Proceedings of the 4th international conference on computational linguistics and intelligent text processing, CICLING'03, Mexico City. Springer, Berlin/Heidelberg, pp 623–634
51. Jaoua FK, Belguithand LH, Jaoua M, BenHamadou A (2009) An automatic multi-documents summarization method based on extracts classification. Int J Comput Sci Eng Syst (IJCSES) 3:221–231
52. Jones KS (1999) Automatic summarising: factors and directions. In: Advances in automatic text summarization. MIT, Cambridge, pp 1–12
53. Keskes I, Boudabous MM, Maaloul MH, Belguith LH (2012) Etude comparative entre trois approches de résumé automatique de documents arabes. In: Actes de la conférence conjointe JEP-TALN-RECITAL'2012: TALN, Grenoble
54. Ku LW, Liang YT, Chen HH (2006) Opinion extraction, summarization and tracking in news and blog corpora. In: AAAI spring symposium: computational approaches to analyzing weblogs, Stanford, pp 100–107
55. Kupiec J, Pedersen J, Chen F (1995) A trainable document summarizer. In: Proceedings of the 18th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR'95, Seattle. ACM, New York, pp 68–73. <http://doi.acm.org/10.1145/215206.215333>
56. Lacatusu VF, Maiorano SJ, Harabagiu SM (2004) Multi-document summarization using multiple-sequence alignment. In: LREC, Lisbon
57. Lin CY (2004) ROUGE: a package for automatic evaluation of summaries. In: Proceedings of ACL workshop on text summarization branches out, Barcelona, p 10
58. Lin CY, Hovy E (1997) Identifying topics by position. In: Proceedings of the fifth conference on applied natural language processing, Washington, DC. Morgan Kaufmann, San Francisco, pp 283–290. <http://dx.doi.org/10.3115/974557.974599>
59. Lin CY, Hovy E (2002) Automated multi-document summarization in neats. In: Proceedings of the second international conference on human language technology research, HLT'02, San Diego. Morgan Kaufmann, San Francisco, pp 59–62
60. Lin CY, Hovy E (2003) The potential and limitations of automatic sentence extraction for summarization. In: Proceedings of the HLT-NAACL 03 on text summarization workshop, HLT-NAACL-DUC'03, Edmonton, vol 5. Association for Computational Linguistics, Stroudsburg, pp 73–80

61. Litvak M, Lipman H, Ben-Gur A, Kisilevich S, Keim DA, Last M (2010) Towards multilingual summarization: a comparative analysis of sentence extraction methods on English and Hebrew corpora. In: Proceedings of the 4th international workshop on cross lingual information access, Beijing, pp 61–69. <http://bib.dbvis.de/uploadedFiles/219.pdf>
62. Litvak M, Last M, Friedman M, Kisilevich S (2011) MUSE – a multilingual sentence extractor. In: Computational linguistics & applications (CLA 11), Jachranka. <http://bib.dbvis.de/uploadedFiles/362.pdf>
63. Luhn H (1958) The automatic creation of literature abstracts. IBM J 2:159–165
64. Maaloul MH, Keskes I, Belguith LH (2010) Résumé automatique de documents arabes basé sur la technique RST. In: Actes de TALN 2010, Montréal
65. Maaloul MH, Khemakhem ME, Belguith LH (2008) Al lakas el'eli: un système de résumé automatique de documents arabes. In: International Business Information Management Association (IBIMA'2008), Marrakesh
66. Maaloul MH, Keskes I, Belguith LH, Blache P (2010) Automatic summarization of Arabic texts based on rst technique. In: International conference on enterprise information systems (ICEIS) 2, Funchal
67. Maaloul MH, Ajjal W, Belguith LH (2012) Role of linguistic analysis in detecting rhetorical relations. In: International conference on Arabic language processing, CITALA'2012, Rabat
68. Mani I (2001) Automatic summarization. John Benjamins, Amsterdam/Philadelphia
69. Mani I, Bloedorn E (1999) Summarizing similarities and differences among related documents. Inf Retr 1(1–2):35–67
70. Mani I, Maybury MT (2001) Automatic summarization. In: Association for Computational Linguistics, Toulouse
71. Mann WC, Thompson SA (1988) Rhetorical structure theory: toward a functional theory of text organization. Text 8(3):243–281
72. Marcu D (2000) The theory and practice of discourse parsing and summarization. MIT, Cambridge
73. Marcu D, Carlson L, Watanabe M (2000) The automatic translation of discourse structures. In: ANLP, Seattle, pp 9–17
74. Mathkour HI, Touir AA, Al-Sanea WA (2008) Parsing Arabic texts using rhetorical structure theory. J Comput Sci 4(9):713–720
75. Maybury MT (ed) (1999) Advances in automatic text summarization. MIT, Cambridge
76. McKeown KR, Klavans JL, Hatzivassiloglou V, Barzilay R, Eskin E (1999) Towards multidocument summarization by reformulation: progress and prospects. In: Proceedings of the sixteenth national conference on artificial intelligence and the eleventh innovative applications of artificial intelligence conference innovative applications of artificial intelligence, AAAI'99/IAAI'99, Orlando. American Association for Artificial Intelligence, pp 453–460
77. Melli G, Shi Z, Wang Y, Liu Y, Sarkar A, Popowich F (2006) Description of SQUASH, the SFU question answering summary handler for the DUC-2006 summarization task. In: Proceedings of the document understanding conference 2006 (DUC'2006), New York City
78. Minel JL (2002) Filtrage sémantique: du résumé automatique à la fouille de textes. Hermès Science, Paris
79. Minel JL, Descles JP, Cartier E, Crispino G, Ben Hazez S, Jackiewicz A (2009) Résumé automatique par filtrage sémantique d'informations dans des textes. Revue Techniques et Sciences Informatiques
80. Mori T, Nozawa M, Asada Y (2004) Multi-answer-focused multi-document summarization using a question-answering engine. In: Proceedings of the 20th international conference on computational linguistics, COLING'04, Geneva. Association for Computational Linguistics
81. Nenkova A, Passonneau R (2004) Evaluating content selection in summarization: the pyramid method. In: Human language technologies: conference of the North American chapter of the Association of Computational Linguistics HLT/NAACL, Boston, pp 145–152
82. Nobata C, Sekine S (2004) CRL/NYU summarization system at DUC-2004. In: DUC'2004, Boston

83. Ono K, Sumita K, Miike S (1994) Abstract generation based on rhetorical structure extraction. In: Proceedings of COLING, Kyoto, pp 344–348
84. Ou S, Khoo CSG, Goh DH (2008) Design and development of a concept-based multi-document summarization system for research abstracts. *J Inf Sci* 34(3):308–326
85. Paice CD (1990) Constructing literature abstracts by computer: techniques and prospects. *Inf Process Manag* 26(1):171–186. Special issue: Natural Language Processing and Information Retrieval
86. Paice CD, Jones PA (1993) A ‘select and generate’ approach in automatic abstracting. In: Mcnery T, Paice CD (eds) 14th information retrieval colloquium, Lancaster. Springer
87. Radev DR (2000) A common theory of information fusion from multiple text sources step one: cross-document structure. In: Proceedings of the 1st SIGdial workshop on discourse and dialogue, SIGDIAL’00, Hong Kong, vol 10. Association for Computational Linguistics, pp 74–83
88. Radev DR (2001) Experiments in single and multidocument summarization using mead. In: First document understanding conference, New Orleans
89. Radev DR, McKeown KR (1998) Generating natural language summaries from multiple on-line sources. *Comput Linguist* 24(3):470–500. <http://dl.acm.org/citation.cfm?id=972749.972755>
90. Roussarie L, Amsili P (2002) Discours et compositionnalite. In: Actes de la 9eme Conference sur le Traitement Automatique des Langues Naturelles (TALN 2002), Nancy, vol 1, pp 383–388. <http://talana.linguist.jussieu.fr/~laurent/Papiers/Taln2002.ps.gz>
91. Salton G, Singhal A, Mitra M, Buckley C (1997) Automatic text structuring and summarization. *Inf Process Manag* 33(3):193–207. [http://dx.doi.org/10.1016/S0306-4573\(96\)00062-3](http://dx.doi.org/10.1016/S0306-4573(96)00062-3)
92. Schlesinger JD, O’Leary DP, Conroy JM (2008) Arabic/English multi-document summarization with classy – the past and the future. In: Gelbukh AF (ed) CICLing, Haifa. Lecture notes in computer science, vol 4919. Springer, pp 568–581
93. Sekine S, Nobata C (2003) A survey for multi-document summarization. In: Proceedings of the HLT-NAACL 03 on text summarization workshop, HLT-NAACL-DUC’03, Edmonton, vol 5. Association for Computational Linguistics, Stroudsburg, pp 65–72. <http://dx.doi.org/10.3115/1119467.1119476>
94. Sitbon L (2007) Robustesse en recherche d’information: application a l’accessibilite aux personnes handicapees. PhD thesis, Universite d’Avignon
95. Sobh I, Darwish N, Fayek M (2007) An optimized dual classification system for Arabic extractive generic text summarization. In: Proceedings of the 7th conference on language engineering, ESLEC’07, Cairo
96. Steinberger J, Kabadjov M, Steinberger R, Tanev H, Turchi M, Vanni Z (2011) JRC’s participation at TAC 2011: guided and multilingual summarization tasks. In: Proceedings of TAC’11, Gaithersburg
97. Teufel S, Moens M (1997) Sentence extraction as a classification task. In: Proceedings of the workshop on intelligent scalable text summarization at the ACL/EACL conference, Madrid, pp 58–65
98. Tratz S, Hovy E (2008) Summarisation evaluation using transformed basic elements. In: Proceedings TAC 2008, Gaithersburg, NIST, p 10p
99. Turney PD (2000) Learning algorithms for keyphrase extraction. *Inf Retr* 2(4):303–336
100. Vanderwende L, Suzuki H, Brockett C, Nenkova A (2007) Beyond sumbasic: task-focused summarization with sentence simplification and lexical expansion. *Inf Process Manag* 43(6):1606–1618
101. Vella G (2010) Automatic summarization of legal documents. Technical report, Master’s thesis. Department CSAI, University of Malta
102. White M, Korelsky T, Cardie C, Ng V, Pierce D, Wagstaff K (2001) Multi-document Summarization via information extraction. In: Proceedings first international conference on human language technology research, San Diego, pp 263–269. <http://acl.ldc.upenn.edu/H/H01/H01-1054.pdf>

103. Yeh JY, Ke HR, Yang WP (2006) Query-focused multidocument summarization based on hybrid relevance analysis and surface feature salience. In: Proceedings of the 6th WSEAS international conference on simulation, modelling and optimization, SMO'06, Lisbon. World Scientific and Engineering Academy and Society (WSEAS), pp 464–469
104. Zhou L, Ticrea M, Hovy EH (2004) Multi-document biography summarization. In: EMNLP, Barcelona, pp 434–441

# Chapter 13

## Automatic Speech Recognition

Hagen Soltau, George Saon, Lidia Mangu, Hong-Kwang Kuo,  
Brian Kingsbury, Stephen Chu, and Fadi Biadsy

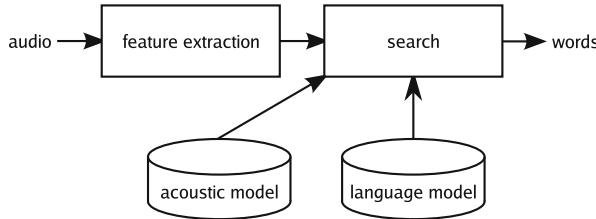
### 13.1 Introduction

In this chapter we describe techniques to build a high performance speech recognizer for Arabic and related languages. The key insights are derived from our experience in the DARPA GALE program, a 5-year program devoted to enhancing the state-of-the-art in Arabic speech recognition and translation. The most important lesson is that general speech recognition techniques work very well also on Arabic. An example is the issue of vowelization: short vowels are often not transcribed in Arabic, Hebrew, and other Semitic languages. Semi-automatic vowelization procedures, specifically designed for the language, can improve the pronunciation lexicon. However, we also can simply choose to ignore the problem at the lexicon level, and compensate for the resulting pronunciation mismatch with the use of discriminative training of the acoustic models. While we focus on Arabic, in this chapter, we speculate that the vast majority of the issues we address here will completely carry over to other Semitic languages. We have tested the approaches discussed in this chapter only on Arabic, as that is the Semitic language with the most resources. Our experimental results demonstrate that such language-independent techniques can solve language-specific issues at least to a large extent. Another example is morphology, where we show that a combination of language-independent techniques (an efficient decoder to deal with large vocabulary and exponential language models) and language-specific techniques (a neural network language model that uses morphological and syntactic features) lead to good results.

---

H. Soltau (✉) • G. Saon • L. Mangu • H.-K. Kuo • B. Kingsbury • S. Chu  
IBM T.J. Watson Research Center, Yorktown Heights, NY, USA  
e-mail: [hsoltau@us.ibm.com](mailto:hsoltau@us.ibm.com); [aon@us.ibm.com](mailto:aon@us.ibm.com); [mangu@us.ibm.com](mailto:mangu@us.ibm.com); [hkuo@us.ibm.com](mailto:hkuo@us.ibm.com);  
[bedk@us.ibm.com](mailto:bedk@us.ibm.com); [schu@us.ibm.com](mailto:schu@us.ibm.com)

F. Biadsy  
Google, New York, NY, USA  
e-mail: [fadi.biadsy@gmail.com](mailto:fadi.biadsy@gmail.com)



**Fig. 13.1** Block diagram of an automatic speech recognition system

For these reasons we describe in the text a list of both language-independent and language-specific techniques. We describe also a full-fledged LVCSR system for Arabic that makes best use of all the techniques. We also demonstrate how this system can be used to bootstrap systems for related Arabic dialects and Semitic languages.

### 13.1.1 Automatic Speech Recognition

Modern speech recognition systems use a statistical pattern recognition approach to the problem of transforming speech signals to text. This approach is data-driven: to build a speech recognition system, practitioners collect speech and text data that are representative of a desired domain (e.g., news broadcasts or telephone conversations), use the collected data to build statistical models of speech signals and text strings in the target domain, and then employ a search procedure to find the best word string corresponding to a given speech signal, where the statistical models provide an objective function that is optimized by the search process. A high-level block diagram of such a speech recognition system is given in Fig. 13.1.

More precisely, in the statistical framework, the problem of speech recognition is cast as

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmax}} P(\mathbf{W}|\mathbf{X}; \Theta) \quad (13.1)$$

where  $\mathbf{W}$  is a word sequence,  $\hat{\mathbf{W}}$  is the optimal word sequence,  $\mathbf{X}$  is a sequence of acoustic feature vectors, and  $\Theta$  denotes model parameters.

Solving this problem directly is challenging, because it requires the integration of knowledge from multiple sources and at different time scales. Instead, the problem is broken down by applying Bayes' rule and ignoring terms that do not affect the optimization, as follows:

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmax}} P(\mathbf{W}|\mathbf{X}; \Theta) \quad (13.2a)$$

$$= \underset{\mathbf{W}}{\operatorname{argmax}} \frac{P(\mathbf{X}|\mathbf{W}; \Theta) P(\mathbf{W}; \Theta)}{P(\mathbf{X}; \Theta)} \quad (13.2b)$$

$$= \underset{\mathbf{W}}{\operatorname{argmax}} P(\mathbf{X}|\mathbf{W}; \Theta) P(\mathbf{W}; \Theta) \quad (13.2c)$$

Referring back to Fig. 13.1, we can identify different components of a speech recognition system with different elements of Eq.(13.2). The *feature extraction* module (Sect. 13.2.1) computes sequences of acoustic feature vectors,  $\mathbf{X}$ , from audio input. The *acoustic model* (Sect. 13.2.1) computes  $P(\mathbf{X}|\mathbf{W}; \Theta)$ : the probability of the observed sequence of acoustic feature vectors,  $\mathbf{X}$ , given a hypothesized sequence of words,  $\mathbf{W}$ . The *language model* (Sect. 13.3.1) computes  $P(\mathbf{W}; \Theta)$ , the prior probability of a hypothesized sequence of words. The *search* process (Sect. 13.3.2) corresponds to the argmax operator.

### 13.1.2 Introduction to Arabic: A Speech Recognition Perspective

An excellent introduction to the Arabic language in the context of ASR can be found in Kirchhoff et al. [27]. Here we describe only a couple of special characteristics of Arabic that may not be familiar to non-Arabic speakers: vowelization and morphology.

#### 1. Vowelization

Short vowels and other diacritics are typically not present in modern written Arabic text. Thus a written word can be ambiguous in its meaning and pronunciation. An Arabic speaker can resolve the ambiguity based on human knowledge and various contextual cues, such as syntactic and semantic information. Although Arabic automatic diacritization has received considerable attention by NLP researchers, the proposed approaches are still error-prone, especially on non-formal texts. When designing an ASR system, we therefore need to consider whether to represent words in the vowelized or un-vowelized form for the language model. Another consideration is whether the pronunciation model or dictionary should contain information derived from the diacritics, such as short vowels.

#### 2. Morphology

Arabic is a morphologically rich language. In Arabic morphology, most morphemes are comprised of a basic word form (the root or stem), to which affixes can be attached to form whole words. Arabic white-space delimited words may be then composed of zero or more prefixes, followed by a stem and zero or more

suffixes. Because of the rich morphology, the Arabic vocabulary for an ASR system can become very large, on the order of one million words, compared with English which typically has a vocabulary on the order of one hundred thousand words. The large vocabulary exacerbates the problem of data sparsity for language model estimation.

Arabic is the only Semitic language for which we have a sufficient amount of data, through the DARPA GALE program; therefore we have decided to focus on developing and testing our ASR techniques only for Arabic. We will describe how our design decisions helped us to successfully overcome some of the Arabic-dependent issues using new sophisticated language-dependent and independent modeling methods and good engineering.

It is important to note that the majority of the modeling techniques and algorithms outlined in this chapter have already been tested on a set of languages such as English, Mandarin, Turkish and Arabic. While we only discuss these ASR approaches for Arabic in this chapter, there is no reason to believe that these approaches would not work for other Semitic languages. The fundamental reason for this is that we were able to tackle language-specific problems with language-independent techniques. As an illustration, we also discuss in this chapter the similarities between the two Semitic languages Arabic and Hebrew. We speculate that the same language-dependent techniques used to address the challenges of Arabic ASR, such as the morphological richness of the language and diacritization would also work for Hebrew.

### ***13.1.3 Overview***

This chapter is organized as follows. In the first two sections, we describe the two major components for state-of-the-art LVCSR: the acoustic model and the language model. For each model, we distinguish between language-independent and language-specific techniques. The language-specific techniques for the acoustic models include vowelization and modeling of dialects in decision trees. The language-specific parts of the language model include a neural network model that incorporates morphological and syntactic features. In Sect. 13.4, we describe how all these techniques are used to build a full-fledged LVCSR system that achieves error rates below 10 % on an Arabic broadcast news task. In Sect. 13.5, we describe techniques that allow us to port Modern Standard Arabic (MSA) models to other dialects, in our case to Levantine. We describe a dialect recognizer and how this can be used to identify relevant training subsets for both the acoustic and language model. We describe a decoding technique that allows us to use a set of dialect-specific models simultaneously during run-time for improved recognition performance. Section 13.6 describes the various data sets we used for system training, development, and evaluation.

## 13.2 Acoustic Modeling

### 13.2.1 Language-Independent Techniques

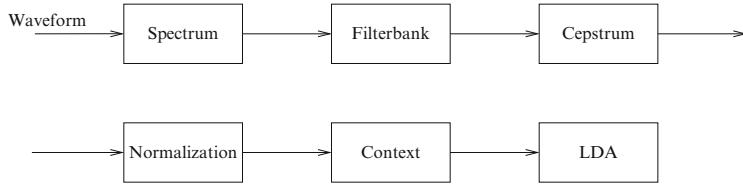
#### Feature Extraction

The goal of the feature extraction module is to compute a representation of the audio input that preserves as much information about its linguistic content as possible, while suppressing variability due to other phenomena such as speaker characteristics or the acoustic environment. Moreover, this representation should be compact (typically generating about 40 parameters for every 10 ms of audio) and should have statistical properties that are compatible with the Gaussian mixture models most often used for acoustic modeling. A very common form of feature extraction is *Mel frequency cepstral coefficients* (MFCCs) [16], and most other approaches to feature extraction, such as perceptual linear prediction (PLP) coefficients, employ similar steps to MFCCs, so we describe their computation in detail below.

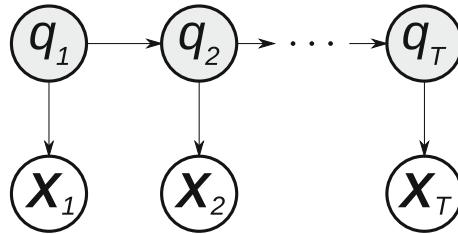
The steps for computing MFCCs are as follows.

1. The short-time fast Fourier transform (FFT) is used to compute an initial time-frequency representation of the signal. The signal is segmented into overlapping frames that are usually 20–25 ms in duration, with one frame produced every 10 ms, each frame is windowed, and a power spectrum is computed for the windowed signal.
2. The power spectral coefficients are binned together using a bank of triangular filters that have constant bandwidth and spacing on the Mel frequency scale, a perceptual frequency scale with higher resolution at low frequencies and lower resolution at high frequencies. This reduces the variability of the speech features without severely impacting the representation of phonetic information. The filter bank usually contains 18–64 filters, depending on the task, while the original power spectrum has 128–512 points, so significant data reduction takes place here.
3. The dynamic range of the features is reduced by taking the logarithm. This operation also means that the features can be made less dependent on the frequency response of the channel and on some speaker characteristics by removing the mean of the features over a sliding window, on an utterance-by-utterance basis, or for all utterances attributed to a given speaker.
4. The features are then decorrelated and smoothed by taking a low-order discrete cosine transform (DCT). Depending on the task, 13–24 DCT coefficients are retained.

In order to remove the effect of channel distortions, the cepstral coefficients are normalized so that they have zero mean and unit variance on a per utterance or a per speaker basis. The final feature stream includes the local temporal characteristics of the speech signal because these convey important phonetic information. Temporal context across frames can be incorporated by computing speed and acceleration



**Fig. 13.2** Frontend pipeline



**Fig. 13.3** Graphical model representation of a hidden Markov model

coefficients (or delta and delta-delta coefficients) from the neighboring frames within a window of typically  $\pm 4$  frames. These dynamic coefficients are appended to the static cepstra to form the final 39-dimensional feature vector. A more modern approach is to replace this ad-hoc heuristic with a linear projection matrix that maps a vector of consecutive cepstral frames to a lower-dimensional space. The projection is estimated to maximize the phonetic separability in the resulting subspace. The feature vectors thus obtained are typically modelled with diagonal covariance Gaussians. In order to make the diagonal covariance assumption more valid, the feature space is rotated by means of a global semi-tied covariance transform. This sequence of processing steps is illustrated in Fig. 13.2.

## Acoustic Modeling

Speech recognition systems model speech acoustics using hidden Markov models (HMMs), which are generative models of the speech production process. A graphical model representation of an HMM is given in Fig. 13.3. An HMM assumes that a sequence of  $T$  acoustic observations  $\mathbf{X} = \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T$  is generated by an underlying discrete-time stochastic process that is characterized by a single, discrete state  $q_t$  taking on values from an alphabet of  $N$  possible states. The state of the generative process is *hidden*, indicated by the shading in Fig. 13.3. The HMM also makes two important conditional independence assumptions. The first is that the state of the generating process at time  $t$ ,  $q_t$ , is conditionally independent of all states and observations, given the state at the previous time step,  $q_{t-1}$ . The second assumption is that the acoustic observation at time  $t$ ,  $\mathbf{x}_t$ , is conditionally independent

of all states and observations, given  $q_t$ . These conditional independence assumptions are denoted by the directed edges in Fig. 13.3.

An HMM is specified by four elements: an alphabet of  $N$  discrete states for the hidden, generative process; a prior distribution over  $q_0$ , the initial state of the generative process;  $P(q_t|q_{t-1})$ , an  $N \times N$  matrix of state transition probabilities; and  $\{P(\mathbf{x}_t|q_t)\}$ , a family of state-conditional probability distributions over the acoustic features. In most large-vocabulary speech recognition systems, the distribution of initial states is uniform over legal initial states. Similarly, the state transition matrix only allows a limited number of possible state transitions, but the distribution over allowable transitions is taken as uniform. Remaining are the methods used to define the state alphabet, which will be described in detail later in this section, and  $\{P(\mathbf{x}_t|q_t)\}$ , the observation probability distributions.

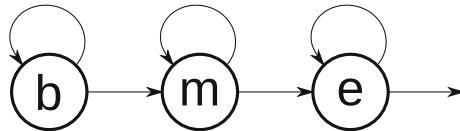
The standard approach to modeling the acoustic observations is to use Gaussian mixture models (GMMs)

$$P(\mathbf{x}_t|q_j) = \sum_{m=1}^M w_{mj} (2\pi)^{-\frac{k}{2}} |\boldsymbol{\Sigma}_{mj}|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}_t - \boldsymbol{\mu}_{mj})^T \boldsymbol{\Sigma}_{mj}^{-1} (\mathbf{x}_t - \boldsymbol{\mu}_{mj})} \quad (13.3)$$

where state  $q_j$  has  $M$   $k$ -dimensional Gaussian mixture components with means  $\boldsymbol{\mu}_{mj}$  and covariance matrices  $\boldsymbol{\Sigma}_{mj}$ , as well as mixture weights  $w_{mj}$  such that  $\sum_m w_{mj} = 1$ . Note that in most cases the covariance matrices are constrained to be diagonal. GMMs are a useful model for state-conditional observation distributions because they can model in a generic manner variability in the speech signal due to various factors, because their parameters can be estimated efficiently using the EM algorithm, and because their mathematical structure allows for forms of speaker and environmental adaptation based on linear regression.

To understand how the HMM state alphabet is defined, it is necessary to understand how words are modeled in large-vocabulary speech recognition systems. Speech recognition systems work with a finite, but large-vocabulary (tens to hundreds of thousands) of words that may be recognized. Words are modeled as sequences of more basic units: either phonemes or graphemes. Phonemes are the basic sound units of a language, and are thus a very natural compositional unit for word modeling. However, using phonemes entails a significant amount of human effort in the design of the speech recognition dictionary: somebody has to produce one or more phonetic pronunciations for every word in the dictionary. This cost has driven the use of graphemic dictionaries, in which words are modeled directly as sequences of letters. While this approach works well for some languages, such as Finnish, that have essentially phonetic spellings of words, it works less well for other languages. For example, consider the pair of letters “GH” in English, which can sound like “f” as in the word “enough,” like a hard “g” as in the word “ghost,” or can be silent as in the word “right.”

To characterize the temporal structure of the basic speech units (phonemes or graphemes), each unit is modeled with multiple states. A popular choice for modeling is illustrated in Fig. 13.4, where the HMM topology is strictly left-to-right



**Fig. 13.4** A typical 3-state, *left-to-right* HMM, drawn as a finite-state machine. The beginning (*b*), middle (*m*), and end (*e*) of a basic speech unit are modeled separately to characterize the temporal structure of the unit

with self loops on the states. The 3-state model shown can have separate observation distributions for the beginning, middle, and end of the unit. If different units have different durations, these may be represented by allocating a greater or fewer number of states to different units.

A final, key ingredient in modeling of speech acoustics is the concept of context dependence. While phonemes are considered to be the basic units of speech, corresponding to specific articulatory gestures, the nature of the human speech apparatus is such that the acoustics of a phoneme are strongly and systematically influenced by the context in which they appear. Thus, the “AE” sound in the word “man” will often be somewhat nasalized because it is produced between two nasal consonants, while the “AE” in “rap” will not be nasalized. Context-dependence can also help with the ambiguity in going from spelling to sound in graphemic systems. Returning to the “GH” example from above, we know for English that a “GH” that occurs at the end of a word and is preceded by the letters “OU” is likely to be pronounced “f”.

Although it produces more detailed acoustic models, context-dependence requires some form of parameter sharing to ensure that there is sufficient data to train all the models. Consider, for example, *triphone* models that represent each phone in the context of the preceding and following phone. In a naive implementation that represents each triphone individually, a phone alphabet of 40 phones would induce a triphone alphabet of  $40^3 = 64,000$  triphones. If phones are modeled with 3-state, left-to-right HMMs as shown above, this would lead to  $3 \times 64,000 = 192,000$  different models. Due to phonotactic constraints, some of these models would not occur, but even ignoring those, the model set would be too large to be practical.

The standard solution to this explosion in the number of models is to cluster the model set using decision trees. Given an alignment of some training data, defined as a labeling of each frame with an HMM state (e.g., the middle state of “AE”), all samples sharing the same label can be collected together and a decision tree can be grown that attempts to split the samples into clusters of similar samples at the leaves of the tree. The questions that are asked to perform the splits are questions about the context of the samples: the identities of the phonetic or graphemic units to the left and right; the membership of the neighboring units in classes such as “vowels,” “nasals,” or “stops;” and whether or not a word boundary occurs at some position to the left or right. A popular splitting criterion is data likelihood under a single, diagonal-covariance Gaussian distribution. In this case, the decision trees

can be trained efficiently by accumulating single-Gaussian sufficient statistics for each context-dependent state, and then growing the decision trees. Once a forest of decision trees has been grown for all units, they are pruned to the desired size. Typically, a few thousand to ten thousand context-dependent states will be defined for a large-vocabulary speech recognition system.

The training process for speech recognition systems is usually iterative, beginning with simple models having few parameters, and moving to more complex models having a larger number of parameters. In the case of a new task, where there are no existing models that are adequately matched to it, speech recognition training begins with a *flat start* procedure. In a flat start, very simple models with no context-dependence and only a single Gaussian per state are initialized directly from the reference transcripts as follows. First, each transcript is converted from a string of words to a string of phones by looking up the words in the dictionary. If a word has multiple pronunciations, a pronunciation is selected at random. This produces a sequence of  $N$  phones. Next, the corresponding sequence of acoustic features is divided into  $N$  equal-length segments, and sufficient statistics for each model are accumulated from its segments. Once the models are initialized, they are refined by running the EM algorithm, and the number of Gaussian mixture components per state is gradually increased.

The models that are produced by a flat start are coarse, and usually have poor transcription accuracy; however, they are sufficient to perform *forced alignment* of the training data. In the forced alignment procedure, the reference word transcripts are expanded into a graph that allows for the insertion of silence between words and the use of the different pronunciation variants in the dictionary, and then the best path through the graph given an existing set of models and the acoustic features for the utterance is found using dynamic programming. The result of this procedure is an alignment of the training data in which every frame is labeled with an HMM state. Given an alignment, it is possible to train context-dependent models, as described above. Typically, for a new task, several context-dependent models of increasing size (in terms of the number of context-dependent HMM states and the total number of Gaussians) will be trained in succession, each relying on a forced alignment from the previous model.

For ASR systems we are interested in the optimality of the recognition accuracy however, and we aim to train the acoustic model discriminatively so as to achieve the lowest word error rate on unseen test data. Directly optimizing the word error rate is hard because it is not differentiable. Alternative approaches look at optimizing smooth objective functions related to word error rate (WER) such as minimum classification error (MCE), maximum mutual information (MMI) and minimum phone error (MPE) criteria. Discriminative training can be applied either to the model parameters (Gaussian means and variances) or to the feature vectors. The latter is done by computing a transformation called feature-space MPE (fMPE) that provides time-dependent offsets to the regular feature vectors. The offsets are obtained by a linear projection from a high-dimensional space of Gaussian posteriors which is trained such as to enhance the discrimination between correct

and incorrect word sequences. Currently, the most effective objective function for model and feature-space discriminative training is called boosted MMI and is inspired by large-margin classification techniques.

## **Speaker Adaptation**

Speaker adaptation aims to compensate for the acoustic mismatch between training and testing environments and plays an important role in modern ASR systems. System performance is improved by conducting speaker adaptation during training as well as at test time by using speaker-specific data. Speaker normalization techniques operating in the feature domain aim at producing a canonical feature space by eliminating as much of the inter-speaker variability as possible. Examples of such techniques are: vocal tract length normalization (VTLN), where the goal is to warp the frequency axis to match the vocal tract length of a reference speaker, and feature-space maximum likelihood linear regression (fMLLR), which consists in affinely transforming the features to maximize the likelihood under the current model. The model-based counterpart of fMLLR, called MLLR, computes a linear transform of the Gaussian means such as to maximize the likelihood of the adaptation data under the transformed model.

### **13.2.2 *Vowelization***

One challenge in Arabic speech recognition is that there is a systematic mismatch between written and spoken Arabic. With the exception of texts for beginning readers and important religious texts such as the Qur'an, written Arabic omits eight diacritics that denote short vowels and consonant length:

1. fatha /a/,
2. kasra /i/,
3. damma /u/,
4. fathatayn (word-ending /an/),
5. karsratayn (word-ending /in/),
6. dammatayn (word-ending /un/),
7. shadda (consonant doubling), and
8. sukuṇ (no vowel).

There are two approaches to handling this mismatch between the acoustics and transcripts. In the “unvowelized” approach, words are modeled graphemically, in terms of their letter sequences, and the acoustics corresponding to the unwritten diacritics are implicitly modeled by the Gaussian mixtures in the acoustic model. In the “vowelized” approach, words are modeled phonemically, in terms of their sound sequences, and the correct vowelization of transcribed words is inferred during training. Note that even when vowelized models are used the word error rate

calculation is based on unvowelized references. Diacritics are typically not orthographically represented in Arabic texts. Diacritization is generally not necessary to make the transcript readable by Arabic literate readers. Thus, Arabic ASR systems typically do not output fully diacritized transcripts. Therefore, the vowelized forms are mapped back to unvowelized forms in scoring – it is also the NIST scoring scheme. In addition, the machine translation systems we use currently require unvowelized input. An excellent discussion of the Arabic language and automatic speech recognition can be found in Kirchhoff et al. [27].

One of the biggest challenges in building vowelized models is initialization: how to obtain a first set of vowelized models when only unvowelized transcripts are available. One approach is to have experts in Arabic manually vowelize a small training set [33]. The obvious disadvantage is that this process is quite labor intensive, which motivates researchers to explore automated methods [2]. Following the recipe in [2], we discuss our bootstrap procedure and some issues related to scaling up to large vocabularies.

## Pronunciation Dictionaries

The words in the vocabulary of both the vowelized and unvowelized systems are assumed to be the same, and they do not contain any diacritics, just as they appear in most written text. In the unvowelized system, the pronunciation of a word is modeled by the sequence of letters in the word. For example, there is a single unvowelized pronunciation of the word *Abwh*.

Abwh(01) A b w h

The short vowels are not explicitly modeled, and it is assumed that speech associated with the short vowels will be implicitly modeled by the adjacent phones. In other words, short vowels are not presented in our phoneme set; acoustically, they will be modeled as part of the surrounding consonant acoustic models.

In the vowelized system, however, short vowels are explicitly modeled in both training and decoding. We use the Buckwalter Morphological Analyzer (Version 2.0) [7], and the Arabic Treebank to generate vowelized variants of each word. The pronunciation of each variant is modeled as the sequence of letters in the diacriticized word, including the short vowels. For *shadda* (consonant doubling), an additional consonant is added, and for *sukun* (no vowel), nothing is added. For example, there are four vowelized pronunciations of the written word *Abwh*.

Abwh(deny/refuse/+they+it/him)	A a b a w o h u
Abwh(desire/aspire/+they+it/him)	A a b b u w h u
Abwh(father+its/it)	A a b u w h u
Abwh(reluctant/unwilling+his/its)	A b u w h u

**Table 13.1** Comparison of different initialization methods for vowelized models

Training method	WER (RT-04)
Flat-start	23.0 %
Bootstrap	22.8 %

The vowelized training dictionary has 243,368 vowelized pronunciations, covering a word list of 64,496 words. The vowelization rate is about 95 %. For the remaining 5 % of words that are not covered, we back off to unvowelized forms.

In the following subsections, we focus on the vowelized system. Since written transcripts of audio data do not usually contain short vowel information, how does one train the initial acoustic model? One could use a small amount of data with manually vowelized transcripts to bootstrap the acoustic model. Alternatively, one could perform flat-start training.

### Flat-Start Training vs. Manual Transcripts

Our flat-start training procedure initializes context-independent HMMs by distributing the data equally across the HMM state sequence. We start with one Gaussian per state, and increase the number of parameters using mixture splitting interleaved within 30 forward/backward iterations. Now, the problem is that we have 3.8 vowelized pronunciations per word on average, but distributing the data requires a linear state graph for the initialization step. To overcome this problem, in the first iteration of training we randomly select pronunciation variants. All subsequent training iterations operate on the full state graph representing all possible pronunciations.

We compare this approach to manually vowelized transcripts where the correct pronunciation variant is given. BBN distributed 10 h of manually vowelized development data (BNAD-05, BNAT-05) that we used to bootstrap vowelized models. These models are then used to compute alignments for the standard training set (FBIS + TDT4). A new system is then built using fixed alignment training, followed by a few forward/backward iterations to refine the models. The error rates in Table 13.1 suggest that manually vowelized transcripts are not necessary. The fully automatic procedure is only 0.2 % worse. We opted for the fully automatic procedure in all our experiments, including the evaluation system.

### Short Models for Short Vowels

We noticed that the vowelized system performed poorly on broadcast conversational speech. It appeared that the speaking rate is much faster, and that the vowelized state graph is too large to be traversed with the available speech frames. The acoustic models do not permit state skipping. One solution is to model the three short vowels with a shorter, 2-state HMM topology. The results are shown in Table 13.2.

**Table 13.2** Comparison of different HMM topologies for short vowels

Topology	RT-04	BCAD-05
3-state	19.0 %	28.9 %
2-state	18.5 %	27.4 %

**Table 13.3** OOV/WER ratio for an unvowelized system on RT-04

Vocabulary	OOV rate	WER
129k	2.9 %	20.3 %
589k	0.8 %	19.0 %

**Table 13.4** OOV/WER ratio for a vowelized system on RT-04

Vocab.	Variants	Vowel. rate	OOV rate	WER
129k	538k	90.4 %	2.9 %	19.8 %
589k	1967k	72.6 %	0.8 %	18.3 %

The improvements on RT-04 (broadcast news) are relatively small; however, there is a 1.5 % absolute improvement on BCAD-05 (broadcast conversations).

### Vowelization Coverage of the Test Vocabulary

As mentioned before, we back off to unvowelized forms for those words not covered by Buckwalter and Arabic Treebank. The coverage for the training dictionary is pretty high at 95 %. On the other hand, for a test vocabulary of 589k words, the vowelization rate is only 72.6 %. The question is whether it is necessary to manually vowelize the missing words, or whether we can get around that by backing off to the unvowelized pronunciations. One way to test this – without actually providing vowelized forms for the missing words – is to look at the OOV/WER ratio. The assumption is that the ratio is the same for a vowelized and an unvowelized system if the dictionary of the vowelized system does not pose any problems. More precisely, if we increase the vocabulary and we get the same error reduction for the vowelized system, then, most likely, there is no fundamental problem with the vowelized pronunciation dictionary.

For the unvowelized system, when increasing the vocabulary from 129k to 589k, we reduce the OOV rate from 2.9 % to 0.8 %, and we reduce the error rate by 1.3 % (Table 13.3). For the vowelized system, we see a similar error reduction of 1.5 % for the same vocabulary increase (Table 13.4). The system has almost 2 million vowelized pronunciations for a vocabulary of 589k words. The vowelization rate is about 72.6 %. In other words, 17.4 % of our list of 589k words are unvowelized in our dictionary. Under the assumption that we can expect the same OOV/WER ratio for both the vowelized and unvowelized system, the results in Table 13.3 and Table 13.4 suggest that the back-off strategy to the unvowelized forms is valid for our vowelized system.

**Table 13.5** Effect of pronunciation probabilities on WER

System	RT-04	BCAD-05
Unvowelized	17.0 %	25.4 %
Vowelized	16.0 %	26.0 %
+ Pron. prob	14.9 %	25.1 %

**Table 13.6** Comparison of vowelized and unvowelized models at various adaptation passes, no pronunciation probabilities. RT-04 test set

Decoding pass	Unvowelized	Vowelized
SI	29.5 %	25.0 %
VTLN	26.2 %	23.1 %
FMLLR	23.0 %	21.1 %
MLLR	22.1 %	20.4 %

## Pronunciation Probabilities

Our decoding dictionary has about 3.3 pronunciations per word on average. Therefore, estimating pronunciation probabilities is essential to improve discrimination between the vowelized forms. We estimated the pronunciation probabilities by counting the variants in the 2,330-h training set.

The setup consists of ML models, and includes all the adaptation steps (VTLN, FMLLR, MLLR). The test sets are RT-04 (Broadcast News) and BCAD-05 (Broadcast Conversations). Adding pronunciation probabilities gives consistent improvements between 0.9 % and 1.1 % on all test sets (Table 13.5). Also, pronunciation probabilities are crucial for vowelized models; they almost double the error reduction from vowelization. We investigated several smoothing techniques and longer word contexts, but did not see any further improvements over simple unigram pronunciation probabilities.

## Vowelization, Adaptation, and Discriminative Training

In this section, we summarize additional experiments with vowelized models. One interesting observation is that the gain from vowelization decreases significantly when more refined adaptation and training techniques are used. Table 13.6 shows the error rate of unvowelized and vowelized models with and without adaptation. The relative improvement from vowelization is more than 15 % at the speaker-independent level. However, after applying several normalization and adaptation techniques, the gain drops to 7.7 % at the MLLR level.

An even more drastic reduction of the vowelization gain is observed after discriminative training (Table 13.7). The vowelized setup includes the 2-state HMMs for short vowels and pronunciation probabilities. While discriminative training reduces the error rate by 4.9 % for the unvowelized setup, we observed an error reduction of only 3.4 % for the vowelized models.

It seems that standard adaptation and discriminative techniques can at least partially compensate for the invalid model assumption of ignoring short vowels, and the improvements from vowelization are subsequently reduced when using

**Table 13.7** Comparison of vowelized and unvowelized models before and after discriminative training. Vowelized system uses pronunciation probabilities and 2-state HMMs for short vowels. Both systems use speaker adaptation: VTLN, FMLLR, and MLLR. DEV-07 test set

Models	Unvowelized	Vowelized
ML	17.1 %	14.8 %
fMPE + MPE	12.2 %	11.4 %

**Table 13.8** Acoustic models trained without and with additional TRANSTAC data

Acoustic model	WER (DEV-07)
2330h GALE	19.2 %
+ 500h TRANSTAC	19.6 %

better adaptation and training techniques. Thus, well-engineered speech recognition systems using only language-independent techniques can perform almost as well as systems using language-specific methods.

### 13.2.3 Modeling of Arabic Dialects in Decision Trees

As shown in Table 13.28, the training data comes from a variety of sources, and there are systematic variations in the data, including dialects (Modern Standard Arabic, Egyptian Arabic, Gulf Arabic, etc.), broadcasters (Al Jazeera, Al Arabiya, LBC, etc.), and programs (Al Jazeera morning news, Al Jazeera news bulletin, etc.).

The problem we face is how to build acoustic models on diverse training data. Simply adding more training data from a variety of sources does not always improve system performance. To demonstrate this, we built two acoustic models with identical configurations. In one case, the model was trained on GALE data (2,330 h, including unsupervised BN-03), while in the second case we added 500 h of TRANSTAC data to the training set. TRANSTAC data contains Iraqi Arabic (Nadji spoken Arabic and Mesopotamian Arabic). Both GALE and TRANSTAC data are Arabic data, but they represent very different dialects and styles. The model trained on additional TRANSTAC data is 0.4 % worse than our GALE model (see Table 13.8).

Both acoustic models used 400k Gaussians, a comparatively large number that should be able to capture the variability of different data sources. Furthermore, we did not find that increasing the number of Gaussians could offset the reduced performance caused by the addition of unmatched training data.

Because adding more training data will not always improve performance, we need to find which part of the training data is relevant. Training separate models for each category or dialect requires sufficient training data for each category, significantly more human and computational effort, and algorithms that reliably cluster training and test data into meaningful categories. We propose instead to model dialects directly in the decision trees: a data-driven method for building dialect-specific models without splitting the data into different categories.

Similar approaches using additional information in decision trees can be found in the literature. Reichl and Chou [37] used gender information to specialize decision trees for acoustic modeling. Speaking rate, SNR, and dialects were used in [19], and questions about the speaking mode were used in [45] to build models for hyperarticulated speech.

## Decision Trees with Dialect Questions

We extend our regular decision tree algorithm by including non-phonetic questions about dialects. The question set contains our normal phonetic context questions, as well as dynamic questions regarding dialect. Dialect questions compete with phonetic context questions to split nodes in the tree. If dialect information is irrelevant for some phones, it will simply be pruned away during tree training.

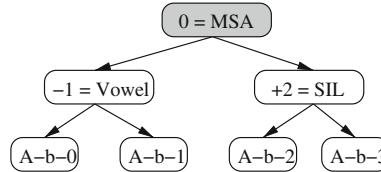
The training of dialect-specific trees and Gaussian mixture models is straightforward. The decision tree is grown in a top-down clustering procedure. At each node, all valid questions are evaluated, and the question with the best increase in likelihood is selected. We use single Gaussians with diagonal covariances as node models. Statistics for each unclustered context-dependent HMM state are generated in one pass over the training data prior to the tree growing. Dialect information is added during HMM construction by tagging phones with additional information. The additional storage cost this entails is quite significant: the number of unique, unclustered context-dependent HMM states increases roughly in proportion to the number of different tags used.

The questions used for tree clustering are written as conjugate normal forms. Literals are basic questions about the phone class or tag for a given position. This allows for more complex questions such as *Is the left context a vowel and is the channel Al Jazeera* ( $-1 = \text{Vowel} \&\& 0 = \text{AlJazeera}$ ). Similarly, more complex questions on the source may be composed. For example, to ask for Al Jazeera, one would write ( $0 = \text{AlJazeeraMorning} \text{ or } 0 = \text{AlJazeeraAfternoon}$ ). The idea is to allow a broad range of possible questions, and to let the clustering procedure select the relevant questions based on the training data. The questions cover all the channel and dialect information available from the audio filenames.

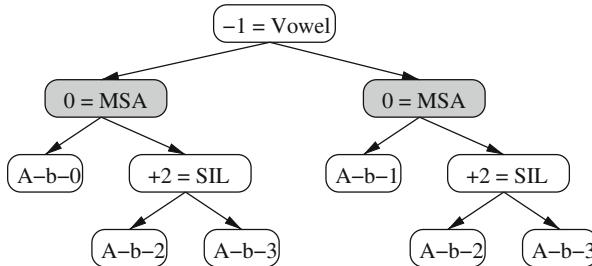
We used this technique to build a dialect-dependent decision tree for the acoustic models trained on the combined GALE and TRANSTAC data. We generated a tree with 15,865 nodes, and 8,000 HMM states. Approximately 44 % of the states depend on the dialect tag, so the remaining 56 % of the models are shared between two very different data sources: GALE and TRANSTAC.

## Building Static Decoding Graphs for Dynamic Trees

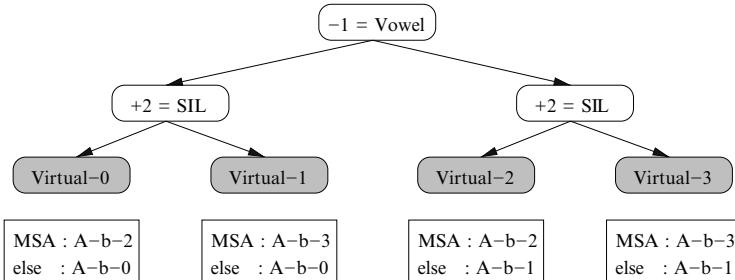
Training dialect-specific models is relatively easy; however, decoding with such models is more complicated if a statically compiled decoding graph is used. The problem is that the decision tree contains dynamic questions that can be answered



**Fig. 13.5** Decision tree with dialect questions. Each question is of the format  $\text{ContextPosition} = \text{Class}$ . Leaves are marked with HMM states  $A-b-0, \dots$  (phone /A/, begin state)



**Fig. 13.6** Decision tree after one transformation step. The original root  $0 = \text{MSA}$  was replaced by the left child  $-1 = \text{Vowel}$ , and a new copy of  $0 = \text{MSA}$  was created



**Fig. 13.7** Reordered decision tree. Dynamic questions are replaced by virtual leaves

only at run-time, and not when the graph is compiled. The solution for this problem is to separate the decision tree into two parts: a static part containing only phonetic questions, and a dynamic part for the dialect questions. The decision tree is reordered such that no dynamic question occurs above a static question. The static part of the decision tree can be compiled into a decoding graph as usual, while the dynamic part of the tree is replaced by a set of virtual leaves. The decoder maintains a lookup table that transforms each virtual leaf to a corresponding dialect-specific leaf at run-time.

In the following we explain how to reorder the decision tree such that dynamic questions do not occur above static questions. Figures 13.5–13.7 illustrate the process. In this example, the root of the tree is marked with the question  $0 = \text{MSA}$ .

**Table 13.9** Reordered tree statistics for GALE + TRANSTAC setup

Number of nodes	15,865
Number of leaves	8,000
Dialect-dependent leaves	44.3 %
Number of reordered nodes	29,137
Number of virtual leaves	7,085

If the center phone is marked as Modern Standard Arabic (MSA), the right branch will be chosen, otherwise the left branch.

The reordering algorithm consists of a sequence of transformation steps. In each step, a dynamic question node is pushed down one level by replacing it with one of its children nodes. Figure 13.6 shows the tree after applying one transformation step. In each transformation step, the selected dynamic question node is duplicated together with one of its branches. In this example, the right branch starting with  $+2 = SIL$  is duplicated. The node to be moved up (promoted) ( $-1 = Vowel$  in Fig. 13.6) is separated from its subtrees and moved up one level, with the duplicated subtrees becoming its children. The subtrees that were originally children of the promoted node become its grandchildren. The resulting tree is equivalent to the original tree, but the dynamic question is now one level deeper.

The reordering procedure terminates when no transformation step can be applied. In the worst case, this procedure causes the decision tree's size to grow exponentially; however, in practice we observe only moderate growth. Table 13.9 summarizes the tree statistics after reordering. The number of internal nodes grew by only a factor of 2, which is easily managed for decoding graph construction.

The last step is to separate the static and dynamic tree parts. The dynamic question nodes are replaced by virtual leaves for the graph construction. The virtual leaves correspond to lookup tables that map virtual leaves to physical HMM states at run-time. The static decoding graph can now be constructed using the tree with virtual leaves. At run-time, dialect information is available,<sup>1</sup> and virtual leaves can be mapped to the corresponding physical HMM states for acoustic score computation.

## Experiments

We use our vowelized Arabic model as a baseline in our experiments. The vocabulary has about 737,000 words, and 2.5 million pronunciations. The language model is a 4-gram LM with 55M n-grams. Speaker adaptation includes VTLN and FMLLR. All models are ML trained. In addition to the GALE EVAL-06 and DEV-07 test sets, we also used a TRANSTAC test set comprising 2 h of audio.

---

<sup>1</sup>This can be done via a separate dialect ID tool, as described in Sect. 13.5.1, selecting the dialect with the best likelihood, or other sources of information.

**Table 13.10** Comparison of regular tree and tree with dialect questions. GALE DEV-07 test set

Acoustic model	Regular tree	Dialect tree
2330h GALE	19.2 %	18.6 %
+ 500h TRANSTAC	19.6 %	18.7 %

**Table 13.11** Comparison of regular tree and tree with dialect questions. TRANSTAC test set

Acoustic model	Regular tree	Dialect tree
2330h GALE	35.9 %	—
+ 500h TRANSTAC	25.9 %	24.7 %

**Table 13.12** Comparison of regular tree and tree with dialect questions with unsupervised training data. GALE EVAL-06 and DEV-07 test sets

Test set	Regular tree	Dialect tree
EVAL-06	30.2 %	29.6 %
DEV-07	20.3 %	19.5 %

The dialect labels are derived from the audio file names. The file names encode TV channel and program information.

In the first experiment we train four acoustic models. Each model has 8,000 states and 400,000 Gaussians. The models are trained on either 2,330 h of GALE data or on the GALE data plus 500 h of TRANSTAC data. For each training set, we train one model using the regular (phonetic context only) question set and one using phonetic and dialect questions. The test set is DEV-07. The results are summarized in Table 13.10. For the GALE model, we see an improvement of 0.6 % WER. The improvement for the GALE + TRANSTAC training set is slightly higher, 0.9 %. The results suggest that the decision tree with dialect questions can better cope with diverse, and potentially mismatched, training data.

In the second experiment (Table 13.11), we use the same set of acoustic models as before, but the vocabulary and language model are now TRANSTAC-specific and the test set is drawn from TRANSTAC data. Adding TRANSTAC data improves the error rate from 35.9 to 25.9 %. Adding the dialect-specific questions to the tree-building process improves the error rate by an additional 1.2 %. We did not test the dialect tree trained on GALE data only. The tree does not contain any TRANSTAC-related questions, since the models were trained on GALE data only.

In the final experiment, we use a large amount of unsupervised training data from our internal TALES data collection. The acoustic model was trained on 2,330 h of GALE data plus 5,600 h of unsupervised training data. The results are shown in Table 13.12. The dialect-dependent decision tree reduces the error rate by 0.6 % on EVAL-06 and 0.8 % on DEV-07. While adding more unsupervised training data does not help if large amounts of supervised training data are available, we observe that the dialect tree is able to compensate for adding “unuseful” data (Table 13.12).

### 13.3 Language Modeling

#### 13.3.1 Language-Independent Techniques for Language Modeling

##### Base $N$ -Grams

The standard speech recognition model for word sequences is the n-gram language model. To see how an n-gram model is derived, consider expanding the joint probability of a sequence of  $M$  words in terms of word probabilities conditioned on word histories:

$$\begin{aligned} P(w_M, w_{M-1}, \dots, w_3, w_2, w_1) &= P(w_1) \times P(w_2|w_1) \times P(w_3|w_2, w_1) \times \dots \times \\ &\quad P(w_M|w_{M-1}, \dots, w_3, w_2, w_1) \end{aligned} \tag{13.4}$$

Given that the words are drawn from a dictionary containing tens to hundreds of thousands of words, it is clear that the probability of observing a given sequence of words becomes vanishingly small as the length of the sequence increases. The solution is to make a Markov assumption that the probability of a word is conditionally independent of previous words, given a history of fixed length  $h$ . That is,

$$\begin{aligned} P(w_M, w_{M-1}, \dots, w_3, w_2, w_1) &\approx P(w_1) \times P(w_2|w_1) \times P(w_3|w_2, w_1) \times \dots \times \\ &\quad P(w_M|w_{M-1}, w_{M-2}) \end{aligned} \tag{13.5}$$

A model that makes a first-order Markov assumption, conditioning words only on their immediate predecessors, is called a *bigram* model, because it deals with pairs of words. Likewise, a model that makes a second-order Markov assumption is called a *trigram* model because it deals with word triplets: two-word histories and the predicted word. Equation (13.5) illustrates a trigram model.

N-gram language models are trained by collecting a large amount of text and simply counting the number of times each word occurs with each history. However, even with the n-gram assumption, there is still a problem with data sparsity: a model based only on counting the occurrences of words and histories in some training corpus will assign zero probability to legal word sequences that the speech recognition system should be able to produce. To cope with this, various forms of smoothing are used on language models that reassign some probability mass from observed events to unobserved events. The models described below generally use modified Kneser–Ney smoothing [1, 13].

## Model M

Model M [11] is a class-based  $n$ -gram model; its basic form is as follows:

$$p(w_1 \cdots w_l) = \prod_{j=1}^{l+1} p(c_j | c_{j-2}c_{j-1}, w_{j-2}w_{j-1}) p(w_j | w_{j-2}w_{j-1}c_j) \quad (13.6)$$

It is composed of two submodels, a model predicting classes and a model predicting words, both of which are exponential models. An exponential model  $p_\Lambda(y|x)$  is a model with a set of *features*  $\{f_i(x, y)\}$  and equal number of parameters  $\Lambda = \{\lambda_i\}$  where

$$p_\Lambda(y|x) = \frac{\exp(\sum_i \lambda_i f_i(x, y))}{Z_\Lambda(x)} \quad (13.7)$$

and where  $Z_\Lambda(x)$  is a normalization factor defined as

$$Z_\Lambda(x) = \sum_{y'} \exp\left(\sum_{i=1}^F \lambda_i f_i(x, y')\right) \quad (13.8)$$

Let  $p_{ng}(y|\omega)$  denote an exponential  $n$ -gram model, where we have a feature  $f_{\omega'}$  for each suffix  $\omega'$  of each  $\omega y$  occurring in the training set; this feature has the value 1 if  $\omega'$  occurs in the current event and 0 otherwise. For example, the model  $p_{ng}(w_j | w_{j-1}c_j)$  has a feature  $f_\omega$  for each  $n$ -gram  $\omega$  in the training set of the form  $w_j, c_j w_j$ , or  $w_{j-1}c_j w_j$ . Let  $p_{ng}(y|\omega_1, \omega_2)$  denote a model containing all features in  $p_{ng}(y|\omega_1)$  and  $p_{ng}(y|\omega_2)$ . Then, the distributions in Eq. (13.6) are defined as follows for the trigram version of Model M:

$$p(c_j | c_{j-2}c_{j-1}, w_{j-2}w_{j-1}) \equiv p_{ng}(c_j | c_{j-2}c_{j-1}, w_{j-2}w_{j-1}) \quad (13.9)$$

$$p(w_j | w_{j-2}w_{j-1}c_j) \equiv p_{ng}(w_j | w_{j-2}w_{j-1}c_j) \quad (13.10)$$

To smooth or regularize Model M, it has been found that  $\ell_1 + \ell_2^2$  regularization works well; i.e., the parameters  $\Lambda = \{\lambda_i\}$  of the model are chosen to minimize

$$\mathcal{O}_{\ell_1 + \ell_2^2}(\Lambda) = \log \text{PP}_{\text{train}} + \frac{\alpha}{C_{\text{tot}}} \sum_i |\lambda_i| + \frac{1}{2\sigma^2 C_{\text{tot}}} \sum_i \lambda_i^2 \quad (13.11)$$

where  $\text{PP}_{\text{train}}$  denotes training set perplexity and where  $C_{\text{tot}}$  is the number of words in the training data. The values  $\alpha$  and  $\sigma$  are regularization hyperparameters, and the values ( $\alpha = 0.5, \sigma^2 = 6$ ) have been found to give good performance for a wide range of operating points. A variant of iterative scaling can be used to find the parameter values that optimize Eq. (13.11).

## Neural Network Language Model

The neural network language model (NNLM) [3, 17, 44] uses a continuous representation of words, combined with a neural network for probability estimation. The model size increases linearly with the number of context features, in contrast to exponential growth for regular  $n$ -gram models. Details of our implementation, speed-up techniques, as well as the probability normalization and optimal NN configuration, are described in [18].

The basic idea behind neural network language modeling is to project words into a continuous space and let a neural network learn the prediction in that continuous space, where the model estimation task is presumably easier than the original discrete space [3, 17, 44]. The continuous space projections, or *feature vectors*, of the preceding words (or context features) make up the input to the neural network, which then will produce a probability distribution over a given vocabulary. The feature vectors are randomly initialized and are subsequently learned, along with the parameters of the neural network, so as to maximize the likelihood of the training data. The model achieves generalization by assigning to an unseen word sequence a probability close to that of a “similar” word string seen in the training data. The similarity is defined as being close in the multi-dimensional feature space. Since the probability function is a smooth function of the feature vectors, a small change in the features leads to only a small change in the probability.

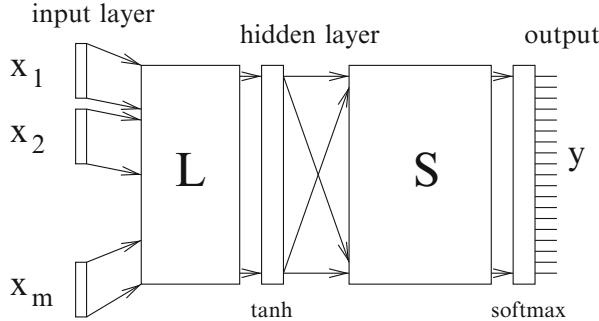
To compute the conditional probability  $P(y|x_1, x_2, \dots, x_m)$ , where  $x_i \in V_i$  (*input vocabulary*) and  $y \in V_o$  (*output vocabulary*), the model operates as follows: First for every  $x_i, i = 1, \dots, m$ , the corresponding feature vector (continuous space projection) is found. This is simply a table lookup operation that associates a real vector of fixed dimension  $d$  with each  $x_i$ . Secondly, these  $m$  vectors are concatenated to form a vector of size  $m \cdot d$ . Finally this vector is processed by the neural network which produces a probability distribution  $P(\cdot|x_1, x_2, \dots, x_m)$  over vocabulary  $V_o$  at its output.

Note that the input and output vocabularies  $V_i$  and  $V_o$  are independent of each other and can be completely different. Training is achieved by searching for parameters  $\Phi$  of the neural network and the values of feature vectors that maximize the penalized log-likelihood of the training corpus:

$$L = \frac{1}{T} \sum_t \log P(y^t|x_1^t, \dots, x_m^t; \Phi) - R(\Phi) \quad (13.12)$$

where superscript  $t$  denotes the  $t$ th event in the training data,  $T$  is the training data size and  $R(\Phi)$  is a regularization term, which in our case is a factor of the L2 norm squared of the hidden and output layer weights.

The model architecture is given in Fig. 13.8 [3, 17, 44]. The neural network is fully connected and contains one hidden layer. The operations of the input and hidden layers are given by:



**Fig. 13.8** The neural network architecture

$$\mathbf{f} = (f_1, \dots, f_{d \cdot m}) = (\mathbf{f}(x_1), \mathbf{f}(x_2), \dots, \mathbf{f}(x_m))$$

$$g_k = \tanh(\sum_j f_j L_{kj} + B_k^1) \quad k = 1, 2, \dots, h$$

where  $\mathbf{f}(x)$  is the  $d$ -dimensional feature vector for token  $x$ . The weights and biases of the hidden layer are denoted by  $L_{kj}$  and  $B_k^1$  respectively, and  $h$  is the number of hidden units.

At the output layer of the network we have:

$$z_k = \sum_j g_j S_{kj} + B_k^2 \quad k = 1, 2, \dots, |V_o|$$

$$p_k = \frac{e^{z_k}}{\sum_j e^{z_j}} \quad k = 1, 2, \dots, |V_o| \quad (13.13)$$

with the weights and biases of the output layer denoted by  $S_{kj}$  and  $B_k^2$  respectively. The softmax layer (Eq. (13.13)) ensures that the outputs are valid probabilities and provides a suitable framework for learning a probability distribution.

The  $k$ th output of the neural network, corresponding to the  $k$ th item  $y_k$  of the output vocabulary, is the desired conditional probability:  $p_k = P(y^t = y_k | x_1^t, \dots, x_m^t)$ .

The neural network weights and biases, as well as the input feature vectors, are learned simultaneously using stochastic gradient descent training via back-propagation algorithm, with the objective function being the one given in Eq. (13.12). Details of the implementation, speed-up techniques, as well as the probability normalization and optimal NN configuration, are described in [18].

Given the large vocabulary of the Arabic speech recognition system, data sparsity is an important problem for conventional  $n$ -gram LMs. Our experience is that NNLM significantly reduces perplexity as well as word error rate in speech recognition. Results will be presented later together with those of an NNLM that incorporates syntactic features.

### 13.3.2 Language-Specific Techniques for Language Modeling

In this section, we describe a language model that incorporates morphological and syntactic features for Arabic speech recognition [29, 30]. This method is language specific in the sense that certain language-specific resources are used, for example, an Arabic parser.

With a conventional  $n$ -gram language model, the number of parameters can potentially grow exponentially with the context length. Given a fixed training set, as  $n$  is increased, the number of unique  $n$ -grams that can be reliably estimated is reduced. Hence in practice, a context no longer than three words (corresponding to a 4-gram LM) is used.

This problem is exacerbated by large vocabularies for morphologically rich languages like Arabic. Whereas the vocabulary of an English speech recognition system typically has under 100k words, our Arabic system has about 800k words. The idea of using rich morphology information in Arabic language modeling has been explored by several researchers. The most common idea has been to use segments, which are the result of breaking an inflected word into parts, for better generalization when estimating the probabilities of  $n$ -gram events [28]. As an example, the white-space delimited word *tqAbhlm* (she met them) is segmented into three morphs: prefix *t* (she), followed by stem *qAb* (met) and suffix *hm* (them).

Compared to a regular word  $n$ -gram model, a segmented word  $n$ -gram model has a reduced context. To model longer-span dependencies, one may consider context features extracted from a syntactic parse tree such as head word information used in the Structured Language Model (SLM) [10]. Syntactic features can be useful in any language. Here is an example in English:

The *girl* who lives down the street *searched* the bushes in our neighbor's backyard *for* her lost kitten.

Through a parse tree, one can relate the words *girl*, *searched*, and *for*. Such long-span relationships cannot be captured with a 4-gram language model. Various types of syntactic features such as head words, non-terminal labels, and part-of-speech tags, have been used in a discriminative LM framework as well as in other types of models [15].

Using many types of context features (morphological and syntactic) makes it difficult to model with traditional back-off methods. Learning the dependencies in such a long context is difficult even with models such as factored language models [28] due to the large number of links that need to be explored. On the other hand, the neural network language model (NNLM) [3, 17, 44] is very suitable for modeling such context features. The NNLM uses a continuous representation of words, combined with a neural network for probability estimation. The model size increases linearly with the number of context features, in contrast to exponential growth for regular  $n$ -gram models. Another advantage is that it is not required to define a back-off order of the context features. The model converges to the same solution no matter how the context features are ordered, as long as the ordering is consistent.

**Table 13.13** WER results of NNLM using word and syntactic features

LM	EVAL08U	BN	BC
4-gram	9.4 %	6.9 %	12.5 %
4-gram + word NNLM	9.1 %	6.5 %	12.1 %
4-gram + word NNLM + syntax NNLM	8.6 %	6.2 %	11.5 %

The following text processing steps are used to extract morphological and syntactic features for context modeling with an NNLM. Arabic sentences are processed to segment words into (hypothesized) prefixes, stems, and suffixes, which become the tokens for further processing. In particular, we use Arabic Treebank (ATB) segmentation, which is a *light* segmentation adopted to the task of manually writing parse trees in the ATB corpus [32]. After segmentation, each sentence is parsed, and syntactic features are extracted. The context features used by the NNLM include segmented words (morphs) as well as syntactic features such as exposed head words and their non-terminal labels.

Table 13.13 shows the WER results of using NNLMs, with word features only or with morphological and syntactic features. The results are given for an evaluation set EVAL08U, with a breakdown for the broadcast news (BN) and broadcast conversations (BC) portions. An NNLM with word features reduced the WER by about 3 % relative, from 9.4 to 9.1 %. Using morphological and syntactic features further reduced the WER by 5 % relative, from 9.1 to 8.6 %. It is seen that syntactic features are helping both BN and BC. Specifically, through the use of syntactic features, for BN, the WER improved by 4.6 % (6.5–6.2 %) and for BC, the WER improved by 5.0 % (12.1–11.5 %).

Although the modeling methodology behind the syntax NNLM is language independent, when a new language or dialect is encountered, certain resources such as a segmenter and parser may have to be developed or adapted. In our experiments, even though the syntax NNLM was trained on only 12 million words of data, two orders of magnitude less than the text corpora of over 1 billion words used to train the *n*-gram LM, it was still able to provide significant improvements to the WER. For a new language with little data available to train the *n*-gram LM, the syntax LM is likely to help even more.

## Search

The search space for large-vocabulary speech recognition is both enormous and complex. It is enormous because the number of hypotheses to consider grows exponentially with the length of the hypothesized word strings and because the number of different words that can be recognized is typically in the tens to hundreds of thousands. The search space is complex because there is significant structure induced by the language model, the dictionary, and the decision trees used for HMM state clustering. One approach to search, the *static* approach, represents the

components of the search space (the language model, dictionary, and decision trees) as weighted finite-state transducers (WFSTs), and then uses standard algorithms such as WFST composition, determinization, and minimization to pre-compile a *decoding graph* that represents the search space. The search process then simply reads in this graph and performs dynamic programming search on it, given a sequence of acoustic feature vectors, to perform speech recognition. Such static decoders can be very efficient and can be implemented with very little code because all of the complexity is pushed into the precompilation process. However, the size of the language models that can be used with static decoders is limited by what models can successfully be precompiled into decoding graphs. An alternative approach, *dynamic search*, constructs the relevant portion of the search space on the fly. Such decoders can use much larger language models, but are also significantly more complicated to write.

## 13.4 IBM GALE 2011 System Description

In this section we describe IBM’s 2011 transcription system for Arabic broadcasts, which was fielded in the GALE Phase 5 machine translation evaluation. Like most systems fielded in competitive evaluations, our system relies upon multiple passes of decoding, acoustic model adaptation, language model rescoring, and system combination to achieve the lowest possible word error rate.

### 13.4.1 Acoustic Models

We use an acoustic training set composed of approximately 1,800 h of transcribed Arabic broadcasts provided by the Linguistic Data Consortium (LDC) for the GALE evaluations.

Unless otherwise specified, all our acoustic models use 40-dimensional features that are computed by an LDA projection of a supervector composed from 9 successive frames of 13-dimensional mean- and variance-normalized PLP features followed by diagonalization using a global semi-tied covariance transform [20], and use pentaphone cross-word context with a “virtual” word-boundary phone symbol that occupies a position in the context description, but does not generate an acoustic observation. Speaker-adapted systems are trained using VTLN and fMLLR. All the models use variable frame rate processing [14].

Given that the short vowels and other diacritic markers are typically not orthographically represented in Arabic texts, we have a number of choices for building pronunciation dictionaries: (1) unvowelized (graphemic) dictionaries in which the short vowels and diacritics are ignored, (2) vowelized dictionaries which use the Buckwalter morphological analyzer [7] for generating possible vowelized pronunciations and (3) vowelized dictionary which uses the output of

a morphological analysis and disambiguation tool (MADA) [23]; the assignment of such diacritic markers is based on the textual context of each word (to distinguish word senses and grammatical functions).<sup>2</sup> Our 2011 transcription system uses the acoustic models described below.

- **SI** A speaker-independent, unvowelized acoustic model trained using model-space boosted maximum mutual information [35]. The PLP features for this system are only mean-normalized. The **SI** model comprises 3k states and 151k Gaussians.
- **U** A speaker-adapted, unvowelized acoustic model trained using both feature- and model-space BMMI. The **U** model comprises 5k states and 803k Gaussians.
- **SGMM** A speaker-adapted, Buckwalter vowelized subspace Gaussian mixture model [36,43] trained with feature- and model-space versions of a discriminative criterion based on both the minimum phone error (MPE) [34] and BMMI criteria. The **SGMM** model comprises 6k states and 150M Gaussians that are represented using an efficient subspace tying scheme.
- **V** A speaker-adapted, Buckwalter vowelized acoustic model trained using the feature-space BMMI and model-space MPE criteria. The changes in this model compared to all the other models are: (1) the “virtual” word boundary phones are replaced with word-begin and word-end tags, (2) it uses a dual decision tree that specifies 10k different Gaussian mixture models, but 50k context-dependent states, (3) it uses a single, global decision tree and (4) expands the number of phones on which a state can be conditioned to  $\pm 3$  within words. This model has 801k Gaussians.
- **BS** A speaker-adapted, unvowelized acoustic model using Bayesian sensing HMMs where the acoustic feature vectors are modeled by a set of state-dependent basis vectors and by time-dependent sensing weights [40]. The Bayesian formulation comes from assuming state-dependent Gaussian priors for the weights and from using marginal likelihood functions obtained by integrating out the weights. The marginal likelihood is Gaussian with a factor analyzed covariance matrix with the basis providing a low-rank correction to the diagonal covariance of the reconstruction error [42]. The details of this model are given in Sect. 13.4.1.
- **M** A speaker-adapted system, MADA vowelized system, with an architecture similar to **V**. The details of this model are given in Sect. 13.4.1.
- **NNU, NNM** Speaker-adapted acoustic models which use neural network features. They were built using either the unvowelized lexicon (**NNU**) or the MADA one (**NNM**). Section 13.4.1 describes these models in more detail.

---

<sup>2</sup>MADA operates by examining a list of all possible Buckwalter morphological analyses for each word, and then selecting the analysis that matches the current context best using SVM classifiers. MADA uses 19 distinct weighted morphological features. The selected analyses carry complete diacritic, lexemic, glossary and morphological information; thus all disambiguation decisions are made in one step. See [23] for more details.

## Bayesian Sensing HMMs (BS)

### Model Description

Here, we briefly describe the main concepts behind Bayesian sensing hidden Markov models [40]. The state-dependent generative model for the  $D$ -dimensional acoustic feature vectors  $\mathbf{x}_t$  is assumed to be

$$\mathbf{x}_t = \Phi_i \mathbf{w}_t + \boldsymbol{\epsilon}_t \quad (13.14)$$

where  $\Phi_i = [\phi_{i1}, \dots, \phi_{iN}]$  is the basis (or dictionary) for state  $i$  and  $\mathbf{w}_t = [w_{t1}, \dots, w_{tN}]^T$  is a time-dependent weight vector. The following additional assumptions are made: (1) when conditioned on state  $i$ , the reconstruction error is zero-mean Gaussian distributed with precision matrix  $R_i$ , i.e.  $\boldsymbol{\epsilon}_t|s_t = i \sim \mathcal{N}(\mathbf{0}, R_i^{-1})$  and (2) the state-conditional prior for  $\mathbf{w}_t$  is also zero-mean Gaussian with precision matrix  $A_i$ , that is  $\mathbf{w}_t|s_t = i \sim \mathcal{N}(\mathbf{0}, A_i^{-1})$ . It can be shown that, under these assumptions, the marginal state likelihood  $p(\mathbf{x}_t|s_t = i)$  is also zero-mean Gaussian with the factor analyzed covariance matrix [42]

$$S_i \triangleq R_i^{-1} + \Phi_i A_i^{-1} \Phi_i^T \quad (13.15)$$

In summary, the state-dependent distributions are fully characterized by the parameters  $\{\Phi_i, R_i, A_i\}$ . In [40], we discuss the estimation of these parameters according to a maximum likelihood type II criterion, whereas in [41] we derive parameter updates under a maximum mutual information objective function.

### Automatic Relevance Determination

For diagonal  $A_i = \text{diag}(\alpha_{i1}, \dots, \alpha_{iN})$ , the estimated precision matrix values  $\alpha_{ij}$  encode the relevance of the basis vectors  $\phi_{ij}$  for the dictionary representation of  $\mathbf{x}_t$ . This means that one can use the trained  $\alpha_{ij}$  for controlling model complexity. One can first train a large model and then prune it to a smaller size by discarding the basis vectors which correspond to the largest precision values of the sensing weights.

### Initialization and Training

We first train a large acoustic model with 5,000 context-dependent HMM states and 2.8 million diagonal covariance Gaussians using maximum likelihood in a discriminative FMMI feature space. The means of the GMM for each state are then clustered using k-means. The initial bases are formed by the clustered means. The resulting number of mixture components for the Bayesian sensing models after the clustering step was 417k. The precision matrices for the sensing weights and the

reconstruction errors are assumed to be diagonal and are initialized to the identity matrix.

The models are trained with six iterations of maximum likelihood type II estimation. Next, we discard 50 % of the basis vectors corresponding to the largest precision values of the sensing weights and retrain the pruned model for two additional ML type II iterations. We then generate numerator and denominator lattices with the pruned models and perform four iterations of boosted MMI training of the model parameters as described in [41]. The effect of pruning and discriminative training is discussed in more details in [42].

### MADA-Based Acoustic Model (M)

This acoustic model is similar to the **V** model. It uses a global tree, word position tags, and a large phonetic context of  $\pm 3$ . While the MADA-based model uses approximately the same number of Gaussians, the decision tree uses only one level, keeping the number of HMM states to 10,000. Since the MADA-based model uses a smaller phone set than the Buckwalter vowelized models, we were able to reuse the vowelized alignments and avoid the flat-start procedure. In this section we describe the strategy used for constructing training and decoding pronunciation dictionaries, the main difference between this system and the **V** system. Both pronunciation dictionaries are generated following [5] with some slight modification.

#### Training Pronunciation Dictionary

Here we describe an automatic approach to building a pronunciation dictionary that covers all words in the orthographic transcripts of the training data. First, for each utterance transcript, we run MADA to disambiguate each word based on its context in the transcript. MADA outputs all possible fully-diacritized morphological analyses for each word, ranked by their confidence, the MADA confidence score. We thus obtain a fully-diacritized orthographic transcription for training. Second, we map the highest-ranked diacritization of each word to a set of pronunciations, which we obtain from the 15 pronunciation rules described in [5]. Since MADA may not always rank the best analysis as its top choice, we also run the pronunciation rules on the **second** best choice returned by MADA, when the difference between the top two choices is less than a threshold determined empirically (in our implementation we chose 0.2). The IBM system is flexible enough to allow specifying multiple diacritized word options at the (training) transcript level. A sentence can be a sequence of fully diacritized word pairs as opposed to a sequence of single words. This whole process gives us fully disambiguated and diacritized training transcripts with more than one or two options per word.

## Decoding Pronunciation Dictionary

For building the decoding dictionary we run MADA on the transcripts of the speech training data as well as on the Arabic Gigaword corpus. In this dictionary, all pronunciations produced (by the pronunciation rules) for all diacritized word instances (from MADA first and second choices) of the same undiacritized form are mapped to the undiacritized and *normalized* word form. Word normalization here refers to removing diacritic markers and replace Buckwalter normalized Hamzat-Wasl ({}), <, and > by the letter ‘A’. Note that it is standard to produce undiacritized transcripts when recognizing MSA. Diacritization is generally not necessary to make the transcript readable by Arabic-literate readers. Therefore, entries in the decoding pronunciation dictionary need only to consist of undiacritized words mapped to a set of phonetically-represented diacritizations.

A pronunciation confidence score is calculated for each pronunciation. We compute a pronunciation score  $s$  for a pronunciation  $p$  as the average of the MADA confidence scores of the MADA analyses of the word instances that this pronunciation was generated from. We compute this score for each pronunciation of a normalized undiacritized word. Let  $m$  be the maximum of these scores. Now, the final pronunciation confidence score for  $p$  is  $-\log_{10}(c/m)$ . This basically means that the best pronunciation receives a penalty of 0 when chosen by the ASR decoder. This dictionary has about 3.6 pronunciations per word when using the first and second MADA choices.

## Neural Network Acoustic Models (NNU and NNM)

The neural network feature extraction module uses two feature streams computed from mean and variance normalized, VTLN log Mel spectrograms, and is trained in a piecewise manner, in which (1) a state posterior estimator is trained for each stream, (2) the unnormalized log-posteriors from all streams are summed together to combine the streams, and (3) features for recognition are computed from the bottleneck layer of an autoencoder network. One stream, the lowpass stream, is computed by filtering the spectrograms with a temporal lowpass filter, while the other stream, the bandpass stream, is computed by filtering the spectrograms with a temporal bandpass filter. Both filters are 19-point FIR filters. The lowpass filter has a cutoff frequency of 24 Hz. The bandpass filter has a differentiator-like (gain proportional to frequency) response from 0 to 16 Hz and a high-pass cutoff frequency of 27 Hz. The posterior estimators for each stream compute the probabilities of 141 context-independent HMM states given an acoustic input composed from 19 frames of 40-dimensional, filtered spectrograms. They have two 2048-unit hidden layers, use softsign nonlinearities [21] between layers, and use a softmax nonlinearity at the output. The softsign nonlinearity is  $y = x/(1 + |x|)$ . Initial training optimizes the frame-level cross-entropy criterion. After convergence, the estimators are further refined to discriminate between state sequences using the minimum phone error

criterion [24, 34]. Stream combination is performed by discarding the softmax output layer for each stream posterior estimator, and summing the resulting outputs, which may be interpreted as unnormalized log-posterior probabilities. We then train another neural network, containing a 40-dimensional bottleneck layer, as an autoencoder, and use the trained network to reduce the dimensionality of the neural network features. The original autoencoder network has a first hidden layer of 76 units, a second hidden layer of 40 units, a linear output layer, and uses softsign nonlinearities. The training criterion for the autoencoder is the cross-entropy between the *normalized* posteriors generated by processing the autoencoder input and output vectors through a softmax nonlinearity. Once the autoencoder is trained, the second layer of softsign nonlinearities and the weights that expand from the 40-dimensional bottleneck layer back to the 141-dimensional output are removed. Details of this NN architecture are described in [39].

Once the features are computed, the remaining acoustic modeling steps are conventional, using 600k 40-dimensional Gaussians modeling 10k quinphone context-dependent states, where we do both feature- and model-space discriminative training using the BMMI criterion. Two acoustic models were trained using the neural-net features: one (**NNM**) used a MADA-vowelized lexicon, while the other (**NNU**) used an unvowelized lexicon. Note that the posterior estimators used in feature extraction were trained with MADA-vowelized alignments.

### 13.4.2 Language Models

For training language models we use a collection of 1.6 billion words, which we divide into 20 different sources. The two most important components are the broadcast news (**BN**) and broadcast conversation (**BC**) acoustic transcripts (7.5 million words each) corresponding to 1,800 h of speech transcribed by LDC for the GALE program. We use a vocabulary of 795,000 words, which is based on all available corpora, and is designed to completely cover the acoustic transcripts. To build the baseline language model, we train a 4-gram model with modified Kneser–Ney smoothing [13] for each source, and then linearly interpolate the 20 component models with the interpolation weights chosen to optimize perplexity on a held-out set. We combine all the 20 components into one language model using entropy pruning [48]. By varying the pruning thresholds we create (1) a 913 million n-gram LM (no pruning) to be used for lattice rescoring (**Base**) and (2) a 7 million n-gram LM to be used for the construction of static, finite-state decoding graphs.

In addition to the baseline language models described above, we investigated various other techniques which differ in either the features they employ or the modeling strategy they use. These are described below.

- **ModelM** A class-based exponential model [11]. Compared to the models used in the previous evaluation, we use a new enhanced word classing [12]. The bigram mutual information clustering method used to derive word classes in the original

Model M framework is less than optimal due to mismatches between the classing objective function and the actual LM, so the new method attempts to address this discrepancy. Key features of the new method include: (a) a class-based model that includes word n-gram features to better mimic the nature of the actual language modeling, (b) a novel technique for estimating the likelihood of unseen data for the clustering model, and (c) n-gram clustering compared to bigram clustering in the original method. We build Model M models with improved classing on 7 of the corpora with the highest interpolation weights in the baseline model.

- **WordNN** A 6-gram neural network language model using word features. Compared to the model used in the P4 evaluation [25], we train on more data (44 million words of data from **BN**, **BC** and **Archive**). We also enlarge the neural network architecture (increased the feature vector dimension from 30 to 120 and the number of hidden units from 100 to 800) and normalize the models. We create a new LM for lattice rescoring by interpolating this model with the 7 **ModelM** models and **Base**, with the interpolation weights optimized on the held-out set. In the previous evaluation we did not get an improvement by interpolating the WordNN model with model M models, but the changes made this year result in significant improvements.
- **SyntaxNN** A neural network language model using syntactic and morphological features [29]. The syntactic features include exposed head words and their non-terminal labels, both before and after the predicted word. For this neural network model we used the same training data and the same neural network architecture as the one described for **WordNN**. This language model is used for n-best rescoring.
- **DLM** A discriminative language model trained using the minimum Bayes risk (MBR) criterion [31]. Unlike the other LMs, a DLM is trained on patterns of confusion or errors made by the speech recognizer. Our potential features consist of unigram, bigram, and trigram morphs, and we used the perceptron algorithm to select a small set of useful features. With the selected features, we trained the DLM using an MBR-based algorithm, which minimizes the expected loss, calculated using the word error information and posterior probabilities of the N-best hypotheses of all the training sentences. To prepare data for DLM training, we used a Phase 3 unvowelized recognizer trained on 1,500 h of acoustic data to decode an unseen 300 h set. This unseen training set was provided in Phase 4, but adding this data to acoustic or language model training did not improve the system, so it is an ideal set for DLM training. During the evaluation, a single MBR DLM thus trained was used to rescore the N-best hypotheses from all the systems. Although there is a mismatch between training and test conditions, improvements were still observed. Details of these experiments as well as post-eval experiments are presented in [31].

Having many diverse language models, the challenge is to be able to combine them while achieving additive gains, and Sect. 13.4.4 describes our strategy.

### 13.4.3 System Combination

We employ three different techniques for system combination. The first technique is cross-adaptation ( $\times$ ), where the fMLLR and MLLR transforms required by a speaker-adapted acoustic model are computed using transcripts from some other, different speaker-adapted acoustic model. The second technique is tree-array combination (+), a form of multi-stream acoustic modeling in which the acoustic scores are computed as a weighted sum of scores from two or more models that can have different decision trees [47]. The only requirement for the tree-array combination is that the individual models are built using the same pronunciation dictionary. The third technique is hypothesis combination using the `nbest -rover` [50] tool from the SRILM toolkit [49]. In all these combination strategies, the choice of systems to combine was based on performance on a variety of development sets.

### 13.4.4 System Architecture

IBM's 2011 GALE Arabic transcription system is a sequence of multiple passes of decoding, acoustic model adaptation, language model rescoring, and system combination steps. In this section, we show how all the models described in the previous sections are combined to generate the final transcripts. We report results on several data sets: DEV'07 (2.5 h); DEV'09 (2.8 h); EVAL'09, the unsequestered portion of the GALE Phase 4 evaluation set (4.2 h); and EVAL'11, the GALE Phase 5 evaluation set (3 h). EVAL'11 is unseen data on which no tuning was done. In our 2011 evaluation system we have the following steps.

1. Cluster the audio segments into hypothesized speakers.
2. Decode with the **SI** model.
3. Compute VTLN warp factors per speaker using transcripts from (2).
4. Decode using the **U** model cross-adapted on **SI**.
5. Decode using the **SGMM** model cross-adapted on (4).
6. Compute best frame rates per utterance using transcripts from (5).
7. Decode using the **SGMM** model cross-adapted on **U** and frame rates from (6).
8.
  - a. Using the **U** model and transcripts from (5), compute fMLLR and MLLR transforms.
  - b. Using the **BS** model and transcripts from (5), compute fMLLR and MLLR transforms.
  - c. Using the **NNU** model and transcripts from (5), compute fMLLR and MLLR transforms.
  - d. Decode and produce lattices using a tree-array combination of the **U** model with transforms from (8a), the **BS** model with transforms from (8b) and **NNU** with transforms from (8c).
9.
  - a. Using the **M** model and transcripts from (5), compute fMLLR and MLLR transforms.

**Table 13.14** Word error rates for the final three combined models before and after adding LM rescoring passes

Step	Decoding pass	DEV'09	EVAL'09	EVAL'11
(8d)	$(U + BS + NNU) \times SGMM \times U$	12.6 %	9.5 %	8.9 %
(9c)	$(M + NNM) \times SGMM \times U$	13.3 %	9.8 %	9.2 %
(10)	$V \times SGMM \times U$	13.5 %	9.8 %	9.5 %
(14a)	(8d) + simplex	11.4 %	8.4 %	7.8 %
(14b)	(9c) + simplex	11.9 %	8.6 %	8.1 %
(14c)	(10) + simplex	12.5 %	9.2 %	8.4 %
(15)	(14a) + (14b) + (14c)	11.1 %	8.1 %	7.4 %

- b. Using the **NNM** model and transcripts from (5), compute fMLLR and MLLR transforms.
- c. Decode and produce lattices using a tree-array combination of the **M** model with transforms from (9a) and the **NNM** model with transforms from (9b).
- 10. Decode and produce lattices using the **V** model, frame rates from (6) and fMLLR and MLLR transforms computed using transcripts from (5).
- 11. Using an interpolation of **Base**, 7 **ModelM** and one **WordNN** language models
  - a. Rescore lattices from (8d), extract 50-best hypotheses
  - b. Rescore lattices from (9c), extract 50-best hypotheses
  - c. Rescore lattices from (10), extract 50-best hypotheses.
- 12. Parse the 50-best lists from (11) and score them with a **SyntaxNN** language model; produce new language model scores for each hypothesis.
- 13. Score the 50-best lists from (11) with a discriminative language model and produce new language model scores for each hypothesis.
- 14. Combine acoustic scores, language model scores from (11), syntax LM scores from (12) and discriminative LM scores from (13) using simplex
  - a. Add the new scores to the hypotheses from (11a)
  - b. Add the new scores to the hypotheses from (11b)
  - c. Add the new scores to the hypotheses from (11c).
- 15. Combine the hypotheses from (14a), (14b), and (14c) using the `nbest-rover` tool from the SRILM toolkit [49]. This constitutes the final output (Table 13.14).

Table 13.15 shows the word error rates obtained after adding new language models either for lattice or n-best rescoring for the (8d) system. It can be seen that each additional rescoring pass improves the performance, and that the total improvement from language modeling rescoring is 1.1–1.2 % absolute on all the sets. Similar improvements have been obtained on the other two systems that are part of the final system combination ((9c) and (10)).

**Table 13.15** Word error rates for different LM rescoring steps on the  $(U + BS + NNU) \times SGMM \times U.vfr$  (8d) set of lattices

Step	Language model	DEV'09	EVAL'09	EVAL'11
(8d)	Base	12.6 %	9.5 %	8.9 %
(11)	+ ModelM and WordNN	11.7 %	8.8 %	8.2 %
(12)	+ SyntaxNN	11.6 %	8.6 %	7.9 %
(13)	+ DLM	11.5 %	8.6 %	8.0 %
(14)	+ SyntaxNN and DLM	11.4 %	8.4 %	7.8 %

## 13.5 From MSA to Dialects

One of the key challenges in Arabic speech recognition research is how to handle the differences between Arabic dialects. Most recent work on Arabic ASR has addressed the problem of recognizing Modern Standard Arabic (MSA). Little work has focused on dialectal Arabic [26, 51]. Arabic dialects differ from MSA and each other in many dimensions of the linguistic spectrum, morphologically, lexically, syntactically, and phonologically. What makes Arabic dialect challenging in particular is the lack of a well-defined spelling system, resources (i.e., acoustic and LM training data) as well as tools (such as morphological analyzers and disambiguation tools).

In this section, we report a series of experiments about how we can progress from Modern Standard Arabic (MSA) to Levantine ASR, in the context of the GALE DARPA program. While our GALE models achieved very low error rates, we still see error rates twice as high when decoding dialectal data. We make use of a state-of-the-art Arabic dialect recognition system to automatically identify Levantine and MSA subsets in mixed speech of a variety of dialects including MSA. Training separate models on these subsets, we show a significant reduction in word error rate over using the entire data set to train one system for both dialects. During decoding, we use a tree array structure to mix Levantine and MSA models automatically using the posterior probabilities of the dialect classifier as soft weights. This technique allows us to mix these models without sacrificing performance for either variety. Furthermore, using the initial acoustic-based dialect recognition system's output, we show that we can bootstrap a text-based dialect classifier and use it to identify relevant text data for building Levantine language models.

### 13.5.1 Dialect Identification

As mentioned above, we are interested in building Levantine-specific models using the available GALE data. Recall that this data contains a mix of dialects in addition to MSA and that this data has no specific dialect annotations. To build a Levantine-specific ASR system, we need dialect annotations for each utterance since Arabic

speakers, in broadcast conversations (BC), tend to code mix/switch between MSA and their native dialects across utterances and even within the same utterance.<sup>3</sup> In this work, we build a dialect recognition system to identify dialects at the utterance level.

Biadsy et al. [4, 6] have previously shown that a dialect recognition approach that relies on the hypothesis that certain phones are realized differently across dialects achieves state-of-the-art performance for multiple dialect and accent tasks (including Arabic). We make use of this system (described next) to annotate some of our Arabic GALE data.

## Phone Recognizer and Front-End

The dialect recognition approach makes use of phone hypotheses. Therefore, we first build a triphone context-dependent phone recognizer. The phone recognizer is trained on MSA using 50 h of GALE speech data of broadcast news and conversations with a total of 20,000 Gaussians. We use one acoustic model for silence, one for non-vocal noise and another to model vocal noise. We utilize a unigram phone model trained on MSA to avoid bias for any particular dialect.<sup>4</sup> We also use FMLLR adaptation using the top CD-phone sequence hypothesis. Our phone inventory includes 34 phones, 6 vowels and 28 consonants.

## Phone GMM-UBM and Phonetic Representation

The first step in the dialect recognition approach is to build a ‘universal’ acoustic model for each context-independent phone type. In particular, we first extract acoustic features (40d feature vectors after CMVN and FMLLR) aligned to each phone instance in the training data (a mix of dialects). Afterwards, using the frames aligned to the same phone type (in all training utterances), we train a Gaussian Mixture Model (GMM), with 100 Gaussian components with diagonal covariance matrices, for this phone type, employing the EM algorithm. Therefore, we build 34 GMMs. Each phone GMM can be viewed as a GMM-Universal Background Model (GMM-UBM) for that phone type, since it models the general realization of that phone across dialect classes [38]. We call these GMMs *phone GMM-UBMs*.

Each phone type in a given utterance ( $U$ ) is represented with a single MAP (Maximum A-Posteriori) adapted GMM. Specifically, we first obtain the acoustic frames aligned to every phone instance of the same phone type in  $U$ . Then these frames are used to MAP adapt the means of the corresponding phone GMM-UBM

---

<sup>3</sup>In this work, we do not attempt to identify code switching points; we simply assume that an utterance is spoken either in MSA or in purely a regional dialect.

<sup>4</sup>We use true phonetic labels here by generating pronunciation dictionaries using MADA, following [4].

using a relevance factor of  $r = 0.1$ . The resulting GMM of phone type  $\phi$  is called the *adapted phone-GMM* ( $f_\phi$ ). The intuition here is that  $f_\phi$  ‘summarizes’ the variable number of acoustic frames of all the phone instances of a phone-type  $\phi$  in a new distribution specific to  $\phi$  in  $U$  [6].

### A Phone-Type-Based SVM Kernel

Now, each utterance  $U$  can be represented as a set  $S_U$  of adapted phone-GMMs, each of which corresponds to one phone type. Therefore, the size of  $S_U$  is at most the size of the phone inventory ( $|\Phi|$ ). Let  $S_{U_a} = \{f_\phi\}_{\phi \in \Phi}$  and  $S_{U_b} = \{g_\phi\}_{\phi \in \Phi}$  be the adapted phone-GMM sets of utterances  $U_a$  and  $U_b$ , respectively. Using the kernel function in Eq. (13.16), designed by Biadsy et al. [6] which employed the upper bound of KL-divergence-based kernel (13.17), proposed by Campbell et al. [9], we train a binary SVM classifier for each pair of dialects. This kernel function compares the ‘general’ realization of the same phone types across a pair of utterances.

$$K(S_{U_a}, S_{U_b}) = \sum_{\phi \in \Phi} K_\phi(f'_\phi, g'_\phi) \quad (13.16)$$

where  $f'_\phi$  is the same as  $f_\phi$  but we subtract from its Gaussian mean vectors the corresponding Gaussian mean vectors of the phone GMM-UBM (of phone type  $\phi$ ).  $g'_\phi$  is obtained similarly from  $g_\phi$ . The subtraction forces zero contributions from Gaussians that are not affected by the MAP adaptation. And,

$$K_\phi(f_\phi, g_\phi) = \sum_i (\sqrt{\omega_{\phi,i}} \Sigma_{\phi,i}^{-\frac{1}{2}} \mu_i^f)^T (\sqrt{\omega_{\phi,i}} \Sigma_{\phi,i}^{-\frac{1}{2}} \mu_i^g) \quad (13.17)$$

where,  $\omega_{\phi,i}$  and  $\Sigma_{\phi,i}$  respectively are the weight and diagonal covariance matrix of Gaussian  $i$  of the phone GMM-UBM of phone-type  $\phi$ ;  $\mu_i^f$  and  $\mu_i^g$  are the mean vectors of Gaussian  $i$  of the *adapted* phone-GMMs  $f_\phi$  and  $g_\phi$ , respectively.

It is interesting to note that, for (13.16), when  $K_\phi$  is a linear kernel, such as the one in (13.17), each utterance  $S_{U_x}$  can be represented as a single vector. This vector, say  $W_x$ , is formed by stacking the mean vectors of the adapted phone-GMM (after scaling by  $\sqrt{\omega_\phi} \Sigma_\phi^{-\frac{1}{2}}$  and subtracting the corresponding  $\mu_\phi$ ) in some (arbitrary) fixed order, and zero mean vectors for phone types not in  $U_x$ . This representation allows the kernel in (13.16) to be written as in (13.18). This vector representation can be viewed as the ‘phonetic fingerprint’ of the speaker. It should be noted that, in this vector, the phones constrain which Gaussians can be affected by the MAP adaptation (allowing comparison under linguistic constraints realized by the phone recognizer), whereas in the GMM-supervector approach [8], in theory, any Gaussian can be affected by any frame of any phone.

$$K(S_{U_a}, S_{U_b}) = W_a^T W_b \quad (13.18)$$

**Table 13.16** F-Measure for each dialect class using the four-way classifier with 30 s cuts

Dialect	F-measure
Levantine	90.7 %
Iraqi	86.7 %
Gulf	87.1 %
Egyptian	98.6 %

### 13.5.2 ASR and Dialect ID Data Selection

As noted above, the GALE data is not annotated based on dialects. Moreover, to the best of our knowledge, there is no Arabic dialect corpus of similar domain and/or acoustic condition as broadcast conversations. Fortunately, there are telephone conversation corpora available from the LDC for four Arabic dialects (Egyptian, Levantine, Gulf, and Iraqi). To address the acoustic recording and domain issues we build two systems.

In our first system, we train our dialect recognition on dialect data taken from spontaneous telephone conversations from the following Appen corpora: Iraqi Arabic (478 speakers), Gulf (976), and Levantine (985). For Egyptian, we use the 280 speakers in CallHome Egyptian and its supplement. Using the kernel-based approach described above, we train a binary SVM classifier for each pair of dialects on 30 s cuts of 80 % of the speakers (of each corpus). Each cut consists of consecutive speech segments totaling 30 s in length (after removing silence). Multiple cuts are extracted from each speaker.<sup>5</sup> As a result, we obtain six binary classifiers for the four broad Arabic dialects.

To label utterances with dialect ID tags, we need a single four-way classifier to classify the dialect of the speaker to one of the four dialects. To build such a classifier, we first run the six SVM binary classifiers on the remaining 20 % held-out speakers. Every SVM binary classifier provides a posterior probability  $P(C_1|x)$  for each test sample  $x$  of belonging to class  $C_1$ . We use these posteriors as features to train a four-way logistic regression on the 20 % set. The 10-fold cross validation of this classifier is 93.3 %; the F-measure of each dialect class is shown in Table 13.16.

We run this system to annotate a portion of our GALE BC data (after down-sampling to 8 KHz). The dialect recognition system classified 54 h of Levantine speech with a relatively high confidence. Since the dialect ID system is trained on telephone conversations as opposed to broadcast conversations, we asked the LDC to validate/filter the output of the system. We find that about 36 h out of 54 h are tagged as “mostly Levantine”, a 10 h set contains code switching between MSA and Levantine at the utterance level, and an 8 h set contains either other dialects or MSA. Recall that our system is not trained to identify MSA.

We extract a 4 h test set (**LEV\_4h**) to be used for reporting results in all the Levantine ASR experiments. From the remaining 32 h we extract all the utterances

---

<sup>5</sup>The equal error rate reported by Biadsy et al. [6] of this dialect recognition system on the remaining 20 % held-out speakers is 4 %.

**Table 13.17** MADA AM used for dialect ID, WER test

System	WER on DEV-07
50k Gaussians, 1k states, ML	16.8 %
200k Gaussians, 5k states, ML	15.4 %
200k Gaussians, 5k states, fBMMI + BMMI	12.5 %

longer than 20 s, this yields approximately 10 h of data (**LEV\_10**). Part of the transcripts released by LDC for the GALE program have “non-MSA” annotations. This allows us to select a 40 h MSA corpus by choosing speakers whose utterances have no such markings. From this set we select 4 h for our MSA ASR experiments (**MSA\_4h**). From the remaining data, we further select a 10 h set with utterances longer than 20 s (**MSA\_10**).

### 13.5.3 Dialect Identification on GALE Data

Given that now we have gold standard BC MSA and Levantine data (**MSA\_10** and **LEV\_10**), we can train another dialect recognition system to distinguish MSA vs. Levantine for BC acoustic conditions. We divide **LEV\_10** into 9 h for training and 1 h for testing our dialect recognition system. Similarly **MSA\_10** is divided into 9 h for training and 1 h for testing. Note that this amount of acoustic data is typically not sufficient to train dialect identification systems; however, we are interested in making use of the rest of the data for other experiments.

As described in Sect. 13.5.1, for the dialect identification system we need a phone decoder; therefore we carry out a number of experiments to find the best strategy for building it. We train three MADA Vowelized (i.e., a true phonetic-based system) triphone acoustic models in which we vary the number of Gaussians and the number of states, using either ML or discriminative training. First, we test these models for word recognition with our unpruned 4-gram LM. Table 13.17 shows the word error rates on the DEV-07 set.

In the next test, we use the triphone models to decode phone sequences with different phone language models. For each phone decoder we build a dialect classification system using the SVM-Kernel approach described in Sect. 13.5.1. We train the models on 9 h of Levantine data and 9 h of MSA data, and evaluate the results on a test set which contains 1 h of Levantine and 1 h of MSA data. Table 13.18 shows the dialect classification rates for the different acoustic model and phone language model combinations. Based on these results we decided to use the smallest, simplest model (50k Gaussians ML model with unigram phone language model) for the subsequent experiments.

**Table 13.18** Dialect classification performance

System/features	Classification accuracy
50k ML 1-gram phone LM	85.1 %
50k ML 3-gram phone LM	84.5 %
200k ML, 3-gram phone LM	84.9 %
200k fBMMI + BMMI, 3-gram	83.0 %

**Table 13.19** 300 h AM tested on DEV-07

System	Unvowelized	BW vowelized	MADA vowelized
ML	16.6 %	14.2 %	13.9 %
fBMMI + BMMI	12.7 %	11.8 %	11.7 %

### 13.5.4 Acoustic Modeling Experiments

#### Comparing Vowelizations

We select a 300-h subset from our entire GALE training set and train speaker adaptive acoustic models for all three lexical setups. The decoding setup includes VTLN, FMLLR, and MLLR and we use an unpruned 4-gram LM with a 795k vocabulary. First, we test the models on one of our standard GALE development sets, DEV-07, shown in Table 13.19. Pronunciation probabilities are used for both, Buckwalter and MADA systems. Buckwalter and MADA vowelizations perform similarly, while the unvowelized models are 2.7 % worse at the ML level. However, we want to note that the difference is only 1 % after discriminative training. This indicates that discriminative training of context-dependent (CD) GMM models is able to compensate for the lack of (knowledge based) pronunciation modeling to a large degree.

In the next comparison, we test the models on a newly defined MSA test set. The reason behind this set is that we want to use the same methodology for defining/selecting a test set for both Levantine and MSA. We would like to analyze the difficulty of Levantine when compared to MSA under exactly same conditions. We are basically reducing effects related to how and from where the test sets are chosen. DEV-07, for example, is a test set defined by LDC which consists of mostly very clean broadcast news data. This is very likely the reason behind our very low error rates. The MSA\_4h test set is selected randomly from broadcast conversations of our training set and labeled as MSA by our dialect classifier. The reason to select the data from broadcast conversations is to match the conditions of the Levantine test set. All of the Levantine data comes from BC as well. The error rates on this MSA test set (Table 13.20) is almost twice as high as the error rates on DEV-07 (Table 13.19), although both are non-dialectal (MSA) test data. We also see that all three models perform at a similar level (21.2–21.8 %) after discriminative training.

We now compare the models on Levantine data (LEV\_4). Recall that this Levantine test set is part of the GALE corpus identified automatically by our dialect classifier and manually verified by LDC (see Section above). The same

**Table 13.20** 300 h AM tested on MSA\_4h

System	Unvowelized	BW vowelized	MADA vowelized
ML	28.6 %	27.0 %	25.7 %
fBMMI + BMMI	21.8 %	21.7 %	21.2 %

**Table 13.21** 300 h AM tested on LEV\_4h

System	Unvowelized	BW vowelized	MADA vowelized
ML	48.2 %	50.3 %	48.1 %
fBMMI + BMMI	39.7 %	42.1 %	40.8 %

methodology for selecting the test data is used for MSA\_4h and LEV\_4h. Both MSA\_4h and LEV\_4h test sets are excluded from the training of the acoustic and language models. Looking at Tables 13.20 and 13.21, we observe two main points:

1. The error rate for Levantine is almost twice as high as for MSA (39.7 vs. 21.8 %). We compare here the Levantine error rate to MSA\_4h and not to DEV-07. This allows us to attribute the increase in error rate to dialect and not to other effects (how the test set was chosen and how carefully the transcripts were done).
2. Another interesting observation is that the unvowelized models perform best on Levantine (39.4 vs. 40.8 and 42.1 %). We speculate that this is due to the fact that the Buckwalter analyzer, MADA, and the pronunciation rules are designed for MSA – which do not work properly for Levantine words. A dialect-specific morphological analyzer would very likely improve results, but it is unclear that it would significantly reduce the error rate on Levantine given that the unvowelized perform comparably well on MSA data (Table 13.20).

### Selecting Dialect Data from the 300-h Training Subset

We now run the dialect recognition system on our 300-h subset of the GALE training corpus. Out of this training set, we obtain about 37 h labeled as Levantine. This is not sufficient to train a set of acoustic models. One option is to use a deep MLLR regression tree or MAP training. In our experience MLLR works well for limited domain adaptation data, but will not be able to fully utilize a large amount of domain adaptation data. While MAP works better with more adaptation data, it is difficult to use it in combination with feature space discriminative training.

Instead, we use a form of training with weighted statistics. The advantage is that all components of the model (including decision trees) are trained at all training stages (ML, DT) with the new domain data. In our case we have additional information in the form of a dialect posterior probability for each utterance from the dialect classifier. We use this posterior to weight the statistics of each utterance during ML and discriminative training. The modified statistics are computed as shown in formula (13.19).

**Table 13.22** Comparing weighting schemes of training statistics on LEV\_4h, 300 h setup, unvowelized ML models

Training data	WER
Unweighted (300 h)	48.2 %
Hard-weighted (37 h)	48.3 %
Soft-weighted (300 h)	45.3 %

**Table 13.23** 300 h AM tested on LEV\_4h

System	Unvowelized	BW vowelized	MADA vowelized
ML	45.3 %	47.3 %	45.5 %
fBMMI + BMMI	38.4 %	41.4 %	39.2 %

$$E(x) = \sum_i P(\text{dialect}|x_i) * x_i$$

$$E(x^2) = \sum_i P(\text{dialect}|x_i) * x_i^2 \quad (13.19)$$

Table 13.22 shows a comparison of different weighting schemes. In the first row, we simply train on all 300 h regardless whether they are Levantine or MSA. This model gives us an error rate of 48.2 %. In the second row, we train only on the selected Levantine subset of 37 h. The error rate is slightly higher, 48.3 %, due to the lack of training data. In the third row, we train on the same 300 h, but weight the statistics of each utterance individually by the posterior score of the dialect classifier. This provides us with a smoothing of the models, avoids overtraining and we get a 2.9 % error reduction.

We apply now the soft-weighting scheme to all vowelization setups and compare the models both after ML and fBMMI + BMMI training in Table 13.23. The improvement from focusing on Levantine training data can be seen by comparing Table 13.21 with Table 13.23. For example, for the unvowelized models, we obtain 2.9 % absolute error reduction at the ML level, and 1.3 % after discriminative training. Note that we do not add training data, rather we find relevant subsets that match our target dialect.

### Tree Array Combination

When we focus the training on Levantine, we can expect the model to perform worse on MSA data. In fact, the error rate increases from 12.7 to 15.1 % on DEV-07 when we use the Levantine models (Table 13.24). Our toolkit allows us to combine models with different decision trees into one single decoding graph [46]. This enables us to combine different acoustic models in one decoding pass on the fly, without making a hard model selection. The combined acoustic score is the weighted sum of the log likelihoods of the combined models. In our case, we combine the MSA and LEV unvowelized models. The results are in Table 13.24. The first two rows represent

**Table 13.24** Tree array combination of general models with Levantine models in one decoding pass, 300-h unvowelized fBMMI + BMMI setup

Weight for MSA model	Weight for LEV models	DEV-07	LEV_4h
1.0	0.0	12.7 %	39.7 %
0.0	1.0	15.1 %	38.4 %
0.5	0.5	13.3 %	38.2 %
Dialect classifier soft weight		12.9 %	38.4 %

**Table 13.25** Comparing weighting schemes of training statistics on LEV\_4h, 1,800-h setup, unvowelized ML models

Training data	WER
Unweighted (1,800 h)	47.0 %
Hard-weighted (237 h)	42.3 %
Soft-weighted (1,800 h)	43.5 %

the extreme cases where either the MSA or LEV model is used exclusively. In the third row, we weight both models equally and constant for all utterances. The error rate on DEV-07 is 13.3 %, 0.6 % higher than when just using the MSA model, but much better than when using the LEV models only (15.1 %). On the other hand, we get a small improvement on the Levantine test set (38.4 % goes to 38.2 %). This is a system combination effect. We used tree arrays in the past as an alternative to ROVER or cross-adaptation, for example in our latest GALE evaluation. In the last row in Table 13.24 we use the posterior of the dialect classifier as a soft weight for model combination on a per utterance basis. This automatic strategy gives us an error rate that is close to the optimal performance of a model selected manually.

### Selecting Dialect Data from the 1,800-h Training Set

The full GALE training corpus consists of about 18,00 h. Similar to the previous experiments, but now focusing exclusively on the unvowelized models, we generate dialect labels for the entire training corpus. The dialect recognition system identified about 237 h as Levantine in the GALE corpus (or 13 %). In Table 13.25, we compare different weighting schemes for the Levantine data. In contrast to the 300 h setup (Table 13.22), the best error rate is achieved now by training exclusively on the 237 h of Levantine data and not by using the dialect scores to weight the statistics. The reason is simply that the amount of Levantine training data is now large enough to train acoustic models and we do not need to add data as was the case for the previous experiments when we had only 37 h of Levantine data.

After discriminative training (fBMMI + bMMI) of the 237 h unvowelized Levantine models, the error rate goes down to 36.3 %. In other words, we can lower the error rate by almost 10 % relative by focusing on relevant subsets of the training data and the dialect classifier together with the tree array decoding technique which allows us to use both Levantine and MSA models in one decoding pass, so the engine can handle both dialectal and non-dialectal utterances at the same time.

**Table 13.26** Text only dialect classification using Levantine and MSA language models

Test data	Dialect classification
MSA_4h	86.0 %
Lev_4h	87.2 %

**Table 13.27** LM rescoring with Levantine LM

Training data	WER
913m 4-gram baseline LM	36.3 %
+ 3-gram Levantine LM from 238 h set	35.4 %
+ 4-gram Levantine weighted LM (all text sources)	35.1 %

### 13.5.5 *Dialect ID Based on Text Only*

The experiments described in Sect. 13.5.4 demonstrate that the acoustic training data contains relevant dialect subsets which when detected can improve the acoustic models. In this section, we report on a similar strategy for language modeling, but now we built a dialect classifier based on text only – no audio data is used. First, we build a Kneser–Ney smoothed 3-gram Levantine LM on the 2 million words corresponding to the transcripts of the 237 h of Levantine acoustic training data (identified automatically). Similarly, we build an MSA language model from all the utterances which are classified as MSA with more than 95 % probability by the dialect annotator. We build a text dialect classifier which simply checks the log-likelihood ratio of the two LMs on a given utterance. Table 13.26 shows that we can predict the dialect reliably even when only text data is available.

#### Levantine LM

Our GALE language models are trained on a collection of 1.6 billion words, which we divide into 20 parts based on the source. We train a 4-gram model with modified Kneser–Ney smoothing [13] for each source, and then linearly interpolate the 20 component models with the interpolation weights chosen to optimize perplexity on a held-out set. In order to build a Levantine language model, we run the text dialect annotator described above on each of the 20 text sources and build 4-gram language models on the 20 dialectal subparts. The new 20 dialect language models are interpolated with the 20 original ones. We optimize the interpolation weights of the 40 language models on a Levantine held-out set. Table 13.27 shows the improvements obtained by adding dialect data to the original language model. Note that the improvement from adding dialect language models is less than the one obtained from dialect acoustic models. One reason for this is the fact that the initial dialect data is selected from the BC part of the training data, and the BC language model has a high weight in the baseline interpolated LM.

## Finding Levantine Words

We can identify dialectal words if we compute how many times the word occurs in the Levantine corpus vs. the MSA one. After sorting the count ratios, we find the following words ranked at the top of the list: Em, hyk, bdw, bdk, ylly, blbnAn, which are in fact Levantine words. Note that identifying dialectal words can be useful for building better pronunciation dictionaries for dialects as well as for machine translation.

## 13.6 Resources

As explained in Sect. 13.1.1 a speech recognizer consists of models representing the underlying acoustic and language characteristics. In order to build a speech recognizer, language-specific data for training these models is required. The Linguistic Data Consortium ([www.ldc.upenn.edu](http://www.ldc.upenn.edu)) is a good starting point to look for publically available data resources. In this section we describe the available data for Arabic that was mostly collected as part of DARPA programs.

### 13.6.1 Acoustic Training Data

We used the following corpora for acoustic model training in various experiments presented here:

- 85 h of FBIS and TDT-4 audio with transcripts provided by BBN,
- 51 h of transcribed GALE data provided by the LDC for the GALE Phase 1 (P1) evaluation,
- 700 h of transcribed GALE data provided by LDC for the Phase 2 (P2) and Phase 2.5 (P2.5) evaluations,
- 5.6 h of BN data with manually vowelized transcripts provided by BBN (BNAT-05),
- 5.1 h of BN data with manually vowelized transcripts provided by BBN (BNAD-05),
- 500 h of transcribed Iraqi data (TRANSTAC),
- 1,800 h of untranscribed audio from the EARS BN-03 corpus, and
- 10,000 h of untranscribed audio collected at IBM Research (TALES).

BNAT-05 and BNAD-05 were used only for a comparison of flat-start training to manual data for the initialization of vowelized models. The TRANSTAC data was used only for experiments on dialect modeling. The TALES data was used only for experiments on large-scale unsupervised training, including tests of dialect models. The evaluation system described in Sect. 13.4 was trained only on data available to all GALE program participants: FBIS, TDT-4, GALE P1 and P2, and EARS BN-03.

**Table 13.28** Source variability in Arabic broadcast training data

Source	Hours
Al Arabiya	2,455
Al Jazeera Morning News	306
Al Jazeera Midday News	326
Al Jazeera News Bulletin	975
Dubai News	27
ESC News	81
LBC (Lebanese) Flash News	88
LBC International News	161
Voice of America	184
Al Alam	47
NBN News	131

Arabic broadcast news and conversations are highly variable, coming from many different broadcasters and containing a mixture of Modern Standard Arabic (MSA) and dialectical Arabic. To illustrate this variability, Table 13.28 provides a breakdown of a sample comprising 130 h of GALE transcribed data, 750 h of untranscribed EARS BN-03 data, and 5,600 h of untranscribed TALES data by source. Note that only the predominant sources in the sample are listed; there is also a long tail containing many additional sources which are represented by smaller amounts of audio.

### 13.6.2 *Training Data for Language Modeling*

We used the following resources for language modeling:

- Transcripts of the audio data released by LDC (7M words),
- The Arabic Gigaword corpus (500M words),
- News group and web log data collected by LDC (22M words),
- Web transcripts for broadcast conversations collected by CMU/ISL (100M words), and
- web text data collected by Cambridge University and CMU (200M words).

Automatic transcripts from the unsupervised training corpus were not used for language modeling.

### 13.6.3 *Vowelization Resources*

Besides the training data, there are other resources that are helpful to build systems for Arabic, but not necessarily essential. Many research groups use these tools for their Arabic LVCSR systems to generate vowelized pronunciation lexica.

In Sect. 13.2.2 we demonstrate how these tools can be used to generate vowelized forms for unvowelized corpora and improve the speech recognition performance.

### 1. Buckwalter’s morphological analyzer

The morphological analyzer [7] can be used to decompose (unvowelized) words into morphemes and provide a list of vowelizations. It is also available from LDC.

### 2. MADA (Morphological Analysis and Disambiguation for Arabic)

This tool [22] can be used to rank and refine the vowelized variants by performing a sentence-wide analysis. It uses the results from Buckwalter’s morphological analyzer to rerank them using an SVM classifier that uses morphological and n-gram features.

### 3. Arabic Treebank

The Arabic Treebank ([www.ircs.upenn.edu/arabic](http://www.ircs.upenn.edu/arabic)) consists of part-of-speech, morphological and syntax annotations for a corpus collected as part of the DARPA TIDES program. The corpus can be used to find vowelized forms for pronunciation lexica.

## 13.7 Comparing Arabic and Hebrew ASR

The goal of this section is to describe the challenges involved in building an ASR system for modern Hebrew. We hypothesize that most of the techniques used for building an Arabic ASR will carry over to Hebrew as well.

Hebrew is a morphologically rich language, therefore, as in Arabic, the vocabulary size can be very large, if explicit morphological modeling is not employed, affecting the lexicon size as well as the language model. All of this may highly likely lead to higher language model perplexity, larger decoder search space, and large memory requirements. We have seen in this chapter how to handle these issues for Arabic, hence the same techniques are likely to carry over. The diacritization phenomenon, or so-called Niqqud in Hebrew, is almost identical to that of Arabic. Short vowels and consonant alternation markers are written as diacritic markers above, below or inside letters. Similar to Arabic, most modern Hebrew texts are written with almost no Niqqud whatsoever. The same word with different Niqqud typically changes meaning, POS, and/or pronunciations, as in Arabic.

As in Arabic, there is almost a one-to-one mapping between fully diacritized Hebrew words to pronunciations using simple pronunciation rules. Therefore, automatic morphology analysis and disambiguation techniques may be useful for Hebrew to build a “vowelized” system as we have seen for Arabic. We also hypothesize that a completely “unvowelized” system, as in Arabic, will also perform relatively well for Hebrew. In other words, ignoring the Niqqud completely may provide us with a relatively accurate system, particularly when discriminative training is used. Unlike Arabic, in modern Hebrew, there is no notion of regional dialects. This suggests that building a Hebrew ASR system is far easier than building

a unified Arabic ASR system. This is due to the facts that Hebrew has a single well-defined spelling system, a single vocabulary set, a single morphological and syntactic system and a single phonetic inventory.

## 13.8 Summary

We described in this chapter LVCSR techniques, both language-independent and language-specific, to obtain high performance LVCSR systems. Our main findings are based on our experience from the 5-year DARPA GALE program, aimed at improving Arabic speech recognition and translations. Our main findings are:

1. Current LVCSR technology works fairly robustly on a set of different languages. Ninety percent of the improvements in our system come from basic technology that works across languages. It is important to get the basic techniques for acoustic and language modeling right before addressing language-specific issues.
2. The lack of diacritics in written texts is one of the more important language-specific issues. Short vowels are not written but spoken, creating a mismatch between transcripts and audio files. The first solution is to use semi-automatic vowelization procedures such as Buckwalter's morphological analyzer. These tools can be used to add vowelized pronunciation variants to the lexicon. Flat-start training over pronunciation graphs can then initialize acoustic models. Repeated system building will lead to very good vowelized acoustic models.
3. Discriminative training is able to compensate for modeling errors. Instead of trying to model short vowels, we explore the idea of letting our acoustic models learn to deal with modeling errors in the pronunciation lexicon. Long-span phonetic context allows decision trees to create states corresponding to effects related to missing short vowels. Discriminative training (feature and model space boosted MMI or MPE) trains acoustic models in a way that corrects modeling errors. Experimental results show that unvowelized models are very competitive once speaker adaptation and discriminative training is added.
4. The rich morphology of the Arabic language is another issue that is important for LVCSR. The rich morphology creates two problems. The first problem is that a lot of words are not covered by regular sized vocabularies. The second problem is data sparsity for language modeling. One solution for these problems is to build a morpheme-based vocabulary and language model. This approach yields very low OOV rates while maintaining regular sized vocabularies. The downside is a postprocessing step that converts the recognition output back to words. This postprocessing step will create additional errors since the mapping is usually ambiguous.

A more elegant approach, in our opinion, is to rely on careful engineering. A well-written LVCSR decoder can easily work with very large vocabularies. Then, we can use word-based vocabularies and simply increase the size of the vocabulary to millions of words. No postprocessing of the recognition output

- is required and the OOV rates on standard tests are below 0.5 %. Class-based exponential language models such as Model-M have proved to be very capable.
5. Increased data sets make language-specific techniques less important. During the course of the GALE program the amount of training data increased from 100 h to more than 1,300 h. We observed that improvements from vowelization were reduced significantly when more training data became available.
  6. Languages such as Arabic cover a wide variety of dialects. We presented several methods for helping LVCSR systems cope with dialects. One approach is to make acoustic models dialect specific. Adding dialect questions to the decision tree is a data-driven way to generate HMM states specific to certain dialects. Another approach is to bootstrap a dialect-specific system by leveraging MSA models. The dialect recognition system allows us to focus on relevant training subsets. While specialized models for Levantine perform poorly on MSA, the tree array decoding procedure allows us to mix both models without sacrificing performance. Also, we showed that we can build a text-only dialect classifier that performs as well as a dialect classifier requiring audio data. The text-only dialect classifier enables us to find relevant LM text data. Another application is pronunciation modeling where the text-based dialect classifier can provide us with candidate words that occur only in Levantine.
  7. Diversity of acoustic models is good for system combination. Instead of relying on either vowelized or unvowelized models, we use both models. The combination can be done as cross-adaption, confusion network combination or other methods. System combination makes LVCSR technology more robust on a variety of test sets and improves the error rate significantly.

**Acknowledgements** We would like to acknowledge the support of DARPA under Grant HR0011-06-2-0001 for funding part of this work. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

## References

1. Kneser R, Ney H (1995) Improved backing-off for m-gram language modeling. In: International conference on acoustics, speech, and signal processing. ICASSP-95, Detroit, vol I, pp 181–184
2. Afify M, Nguyen L, Xiang B, Abdou S, Makhoul J (2005) Recent progress in Arabic broadcast news transcription at BBN. In: Proceedings of the Interspeech, Lisbon, pp 1637–1640
3. Bengio Y, Ducharme R, Vincent P, Jauvin C (2003) A neural probabilistic language model. *J Mach Learn Res* 3:1137–1155
4. Biadsy F (2011) Automatic dialect and accent recognition and its application to speech recognition. PhD. thesis, Columbia University
5. Biadsy F, Habash N, Hirschberg J (2009) Improving the Arabic pronunciation dictionary for phone and word recognition with linguistically-based pronunciation rules. In: Proceedings of NAACL/HLT 2009, Colorado

6. Biadsy F, Hirschberg J, Ellis D (2011) Dialect and accent recognition using phonetic-segmentation supervectors. In: Interspeech, Florence
7. Buckwalter T (2004) LDC2004L02: Buckwalter Arabic morphological analyzer version 2.0. Linguistic Data Consortium, Philadelphia
8. Campbell W, Sturim D, Reynolds D (2006) Support vector machines using GMM supervectors for speaker verification. *IEEE Signal Process Lett* 13(5):308–311
9. Campbell W, Sturim D, Reynolds D, Solomonoff A (2006) SVM based speaker verification using a GMM supervector kernel and NAP variability compensation. In: Proceedings of the ICASSP, France
10. Chelba C, Jelinek F (1998) Exploiting syntactic structure for language modeling. In: Proceedings of the 36th annual meeting of the Association for Computational Linguistics and 17th international conference on computational linguistics, Montreal, pp 225–231
11. Chen SF (2009) Shrinking exponential language models. In: Proceedings of the NAACL-HLT, Boulder
12. Chen S, Chu S (2010) Enhanced word classing for model M. In: Proceedings of the Interspeech, Makuhari, pp 1037–1040
13. Chen SF, Goodman JT (1998) An empirical study of smoothing techniques for language modeling. Technical report TR-10-98, Harvard University
14. Chu S, Povey D, Kuo HK, Mangu L, Zhang S, Shi Q, Qin Y (2010) The 2009 IBM GALE Mandarin broadcast transcription system. In: Proceedings of the ICASSP, Dallas, pp 4374–4377
15. Collins M, Roark B, Saraciar M (2005) Discriminative syntactic language modeling for speech recognition. In: Proceedings of the 43rd annual meeting of the Association for Computational Linguistics, Ann Arbor, pp 507–514
16. Davis SB, Mermelstein P (1980) Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans Acoust Speech Signal Process* 28:357–366
17. Emami A, Jelinek F (2005) A neural syntactic language model. *Mach Learn* 60:195–227
18. Emami A, Mangu L (2007) Empirical study of neural network language models for Arabic speech recognition. In: Proceedings of the ASRU 2007, Kyoto, pp 147–152
19. Fügen C, Rogina I (2000) Integrating dynamic speech modalities into context decision trees. In: Proceedings of the ICASSP, Istanbul
20. Gales MJF (1999) Semi-tied covariance matrices for hidden Markov models. *IEEE Trans Speech Audio Process* 7(3):272–281
21. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the AISTATS, Sardinia, pp 249–256
22. Habash N, Rambow O (2005) Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In: Proceedings of the 43rd annual meeting of the Association for Computational Linguistics (ACL'05), Ann Arbor. Association for Computational Linguistics, pp 573–580. <http://www.aclweb.org/anthology/P/P05/P05-1071>
23. Habash N, Rambow O (2007) Arabic diacritization through full morphological tagging. In: NAACL07, Rochester
24. Kingsbury B (2009) Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling. In: Proceedings of the ICASSP, Taipei, pp 3761–3764
25. Kingsbury B, Soltau H, Saon G, Chu S, Kuo HK, Mangu L, Ravuri S, Morgan N, Janin A (2011) The IBM 2009 GALE Arabic speech transcription system. In: Proceedings of the ICASSP, Prague, pp 4378–4381
26. Kirchhoff K, Vergyri D (2004) Cross-dialectal acoustic data sharing for Arabic speech recognition. In: ICASSP, Montreal
27. Kirchhoff K, Bilmes J, Das S, Duta N, Egan M, Ji G, He F, Henderson J, Liu D, Noamany M, Schone P, Schwartz R, Vergyri D (2003) Novel approaches to Arabic speech recognition: report from the 2002 Johns-Hopkins summer workshop. In: Proceedings of the ICASSP, Hong Kong, pp 344–347

28. Kirchhoff K, Vergyri D, Bilmes J, Duh K, Stolcke A (2006) Morphology-based language modeling for conversational Arabic speech recognition. *Comput Speech Lang* 20(4):589–608
29. Kuo HKJ, Mangu L, Emami A, Zitouni I, Lee YS (2009) Syntactic features for Arabic speech recognition. In: Proceedings of the ASRU 2009, Merano
30. Kuo HKJ, Mangu L, Emami A, Zitouni I (2010) Morphological and syntactic features for Arabic speech recognition. In: Proceedings of the ICASSP 2010, Dallas
31. Kuo H, Mangu L, Arisoy E, Saon G (2011) Minimum Bayes risk discriminative language models for Arabic speech recognition. In: Proceedings of the IEEE ASRU, Waikoloa
32. Maamouri M, Bies A, Buckwalter T, Mekki W (2004) The Penn Arabic Treebank: building a large-scale annotated Arabic corpus. In: Proceedings of NEMLAR conference on Arabic language resources and tools, Cairo, pp 1137–1155
33. Messaoudi A, Gauvain JL, Lamel L (2006) Arabic broadcast news transcription using a one million word vocalized vocabulary. In: Proceedings of the ICASSP, Toulouse, pp 1093–1096
34. Povey D, Woodland PC (2002) Minimum phone error and I-smoothing for improved discriminative training. In: Proceedings of the ICASSP, Orlando, vol I, pp 105–108
35. Povey D, Kanevsky D, Kingsbury B, Ramabhadran B, Saon G, Viswesvariah K (2008) Boosted MMI for model and feature space discriminative training. In: Proceedings of the ICASSP, Las Vegas, pp 4057–4060
36. Povey D, Burget L, Agarwal M, Akyazi P, Feng K, Ghoshal A, Glembek O, Goel NK, Karafiat M, Rastrow A, Rose RC, Schwarz P, Thomas S (2010) Subspace Gaussian mixture models for speech recognition. In: Proceedings of the ICASSP, Dallas, pp 4330–4333
37. Reichl W, Chou W (1999) A unified approach of incorporating general features in decision tree based acoustic modeling. In: Proceedings of the ICASSP, Phoenix
38. Reynolds D, Quatieri T, Dunn R (2000) Speaker verification using adapted Gaussian mixture models. *Digit Signal Process* 10:19–41
39. Sainath T, Kingsbury B, Ramabhadran B (2010) Auto-encode bottleneck features using deep belief networks. In: Proceedings of the ICASSP 2012, Kyoto
40. Saon G, Chien JT (2011) Bayesian sensing hidden Markov models for speech recognition. In: Proceedings of ICASSP, Prague, pp 5056–5059
41. Saon G, Chien JT (2011) Discriminative training for Bayesian sensing hidden Markov models. In: Proceedings of ICASSP, Prague, pp 5316–5319
42. Saon G, Chien JT (2011) Some properties of Bayesian sensing hidden Markov models. In: Proceedings of IEEE ASRU, Waikoloa
43. Saon G, Soltau H, Chaudhari U, Chu S, Kingsbury B, Kuo HK, Mangu L, Povey D (2010) The IBM 2008 GALE Arabic speech transcription system. In: Proceedings of the ICASSP, Dallas, pp 4378–4381
44. Schwenk H (2007) Continuous space language models. *Comput Speech Lang* 21(3). doi:<http://dx.doi.org/10.1016/j.csl.2006.09.003>
45. Soltau H, Waibel A (2000) Phone dependent modeling of hyperarticulated effects. In: Proceedings of the ICSLP, Beijing
46. Soltau H, Saon G, Kingsbury B, Kuo HKJ, Mangu L, Povey D, Emami A (2009) Advances in Arabic speech transcription at IBM under the DARPA GALE program. *IEEE Trans Audio Speech Lang Process* 17(5):884–894
47. Soltau H, Saon G, Kingsbury B (2010) The IBM Attila speech recognition toolkit. In: Proceedings of the IEEE workshop on spoken language technology, Berkeley
48. Stolcke A (1998) Entropy-based pruning of backoff language models. In: Proceedings of the DARPA broadcast news transcription and understanding workshop, Lansdowne, pp 270–274
49. Stolcke A (2002) SRILM – an extensible language modeling toolkit. In: Proceedings of the ICSLP, Denver, pp 901–904
50. Stolcke A, Bratt H, Butzberger J, Franco H, Gadde VRR, Plauche M, Richéy C, Shriberg E, Sonmez K, Weng F, Zheng J (2000) The SRI March 2000 Hub-5 conversational speech transcription system. In: Proceedings of the NIST speech transcription workshop, College Park
51. Vergyri D, Kirchhoff K, Gadde R, Stolcke A, Zheng J (2005) Development of a conversational telephone speech recognizer for Levantine Arabic. In: Interspeech, Lisbon