

ARABIC LANGUAGE MODELING WITH  
STEM-DERIVED MORPHEMES FOR AUTOMATIC  
SPEECH RECOGNITION

DISSERTATION

Presented in Partial Fulfillment of the Requirements for  
the Degree Doctor of Philosophy in the  
Graduate School of The Ohio State University

By

Ilana Heintz, B.A., M.A.

Graduate Program in Linguistics

The Ohio State University

2010

Dissertation Committee:

Prof. Chris Brew, Co-Adviser

Prof. Eric Fosler-Lussier, Co-Adviser

Prof. Michael White

© Copyright by

Ilana Heintz

2010

## ABSTRACT

The goal of this dissertation is to introduce a method for deriving morphemes from Arabic words using stem patterns, a feature of Arabic morphology. The motivations are three-fold: modeling with morphemes rather than words should help address the out-of-vocabulary problem; working with stem patterns should prove to be a cross-dialectally valid method for deriving morphemes using a small amount of linguistic knowledge; and the stem patterns should allow for the prediction of short vowel sequences that are missing from the text. The out-of-vocabulary problem is acute in Modern Standard Arabic due to its rich morphology, including a large inventory of inflectional affixes and clitics that combine in many ways to increase the rate of vocabulary growth. The problem of creating tools that work across dialects is challenging due to the many differences between regional dialects and formal Arabic, and because of the lack of text resources on which to train natural language processing (NLP) tools. The short vowels, while missing from standard orthography, provide information that is crucial to both acoustic modeling and grammatical inference, and therefore must be inserted into the text to train the most predictive NLP models. While other morpheme derivation methods exist that address one or two of the above challenges, none addresses all three with a single solution.

The stem pattern derivation method is tested in the task of automatic speech recognition (ASR), and compared to three other morpheme derivation methods as well as word-based language models. We find that the utility of morphemes in increasing word accuracy scores

on the ASR task is highly dependent on the ASR system's ability to accommodate the morphemes in the acoustic and pronunciation models. In experiments involving both Modern Standard Arabic and Levantine Conversational Arabic data, we find that knowledge-light methods of morpheme derivation may work as well as knowledge-rich methods. We also find that morpheme derivation methods that result in a single morpheme hypothesis per word result in stronger models than those that spread probability mass across several hypotheses per word, however, the multi-hypothesis model may be strengthened by applying informed weights to the predicted morpheme sequences. Furthermore, we exploit the flexibility of Finite State Machines, with which the stem pattern derivation method is implemented, to predict short vowels. The result is a comprehensive exploration not only of the stem pattern derivation method, but of the use of morphemes in Arabic language modeling for automatic speech recognition.

For Michael

I Love You

Promise

## ACKNOWLEDGMENTS

My first and most emphatic *thank you* goes in tandem to my advisors, Dr. Chris Brew and Dr. Eric Fosler-Lussier. My head has been turned around in the last few years, and I am so grateful for it. All of my academic accomplishments are due to your teaching, encouragement, good advice, and commitment to hard work and good science. Thank you for pushing me into the engineering end of computational linguistics - equations, algorithms, proofs and all. Thank you for being patient with me as I ever-so-slowly caught on to the basics. Because of your influence, I am confident in my skills as a linguist, my unexpected knowledge of computer science and artificial intelligence, and general ability to think scientifically. Thank you for funding me through interesting and important research, including technical work that pushed me beyond my comfort zone. Thank you also for providing an extremely supportive and highly collaborative work environment. I will look fondly on these years in your lab and will not forget the lessons learned.

Thank you to Dr. Michael White for serving on my candidacy and dissertation committees, and to Dr. Joseph Zeidan, for kindly serving on my candidacy exam committee. Your commentary on drafts of this dissertation has been invaluable. Thank you both for your help in accomplishing my goals, and for making this research more accurate.

The Dayton Area Graduate Studies Initiative, principally Ray Slyh and Tim Anderson of the Air Force Research Lab in Dayton, supported my studies for far more than the year

in which they funded my research. Thanks to Brian Ore for technical support and for answering many questions about HTK both during and after my time with AFRL.

I was honored to be included in a collaborative project with Dr. Mary Beckman, Dr. Eric Fosler-Lussier, Dr. Ben Munson, and Dr. Jan Edwards on an NSF-funded Human Social Dynamics grant. That work was extremely fulfilling and worthwhile, and allowed me to explore the cognitive science side of linguistics. Funding from this and other NSF grants played an important role in my development as a computational linguist.

Many thanks go to Dr. Allen Gorin, Dr. Beckman, and Dr. Edwards for writing letters in support of the Presidential Fellowship awarded to me by the Graduate School at OSU. The funding allowed me to focus (almost) exclusively on my dissertation work, a time I found crucial to my research.

I also wish to thank Dr. Gorin for introducing me to natural language technology through a summer internship at AT&T Labs in 2001. From that time on, your sage advice to me on graduate school, career plans, and opportunities for doing good in the world have been a great help and inspiration to me.

I thank my classmates in both Linguistics and Computer Science. My SLaTe lab years will always bring good memories, as will time spent kibbitzing in the Solarium/Aquarium with linguists. A special thanks to those who answered my incessant questions, especially on computing and mathematics: Jeremy Morris, Preethi Jyothi and Rohit Prabhavalkar, and also Jon Dehdari and Jim Harmon. My colleagues in the SLaTe lab, also including Darla Shockley, Tim Weale, Billy Hartmann, Josh King, and Anna Wolf have unbelievable insight and are incredibly willing to share their knowledge and abilities. I am so lucky to have worked in such a collaborative, helpful, and fun environment. Thanks to my linguistics cohort for helping me through our coursework, and for moral support throughout

the program, especially my good friends E. Allyn Smith, Kathleen Hall, Sharon Ross, and Anouschka Bergmann, as well as Kirk Baker, Dennis Mehay, DJ Hovermale, Stephen Boxwell, Adriane Boyd, David Durian, Jeff Holliday, Crystal Nakatsu, Julia Papke, and Andy Plummer, especially for help on the proof in the Appendix. Thanks also is due to my graduate student mentors, especially Anna Feldman, Marcus Dickinson, Anton Rytting, Jianguo Li, Xiaofei Lu, Vanessa Metcalf, Jirka Hana, Ila Nagar, Stacey Bailey, Laura Stoia, and Tianfang Xu.

Thank you to the attendees of Clippers and SLaTe lab meetings. From day one of grad school, feedback at those meetings has been useful, creative, and right on target.

My participation in an OSU dissertation writing group was highly valuable to me, and I hope that my comments were valuable to the others in the group - Sue Brennan, Chu Young Chon, Ilana Maymind, and Seth Reno. Thank you, Ilana, for continued editorial and moral support after the summer session had ended. I also owe thanks to Ali Sadiqi in Morocco for skillful editing of my Arabic spelling and grammar.

While all of these people deserve credit for helping me write this dissertation, I take full responsibility for any errors found within it.

Mike, you have been the most patient, loving, generous, and understanding husband any grad student could ask for, and I love you so much for it. Thank you, and the rest of my Bromberg and Heintz family, for your patience, gentle questions about graduation timelines, and curiosity about this strange field that I've chosen. I have felt supported and loved throughout this process, and while that doesn't exactly get the chapters written, it sure helped a lot. All of you help me keep a good perspective on what is important, and also on what is possible. Thanks so much.

## VITA

September 23, 1980 .....	Born Ilana Bromberg Succasunna, NJ
May, 2002 .....	B.A., Linguistics & French Studies Rice University
December, 2007 .....	M.A., Linguistics The Ohio State University
June 2009-June 2010 .....	Presidential Fellow The Ohio State University

## PUBLICATIONS

### Journal Articles

I. Heintz, E. Fosler-Lussier, and C. Brew. “Discriminative Input Stream Combination for Conditional Random Field Phone Recognition”. *IEEE Transactions on Acoustics, Speech, and Language Processing*, volume 17, number 8, pages 1533-1546, November 2009.

### Conference Papers and Books

I. Heintz, M. Beckman, E. Fosler-Lussier, and L. Ménard. “Evaluating parameters for mapping adult vowels to imitative babbling.” *Proceedings of Interspeech 2009*, Brighton, UK, September 2009.

I. Heintz. “Arabic Language Modeling with Finite State Transducers.” *Annual Meeting of the Association of Computational Linguistics: Student Research Workshop*, Columbus, OH, June 2008.

I. Heintz and C. Brew. “Language Modeling for Local and Modern Standard Arabic.” *Language Resources and Evaluations Conference*, Marrakech, Morocco, May 2008.

I. Heintz, E. Fosler-Lussier, and C. Brew. “Latent Phonetic Analysis: Use of Singular Value Decomposition to Determine Features for CRF Phone Recognition.” *International Conference on Acoustics, Speech, and Signal Processing*, Las Vegas, March 2008.

I. Bromberg, Q. Fu, J. Hou, J. Li, C. Ma, B. Matthews, A. Moreno-Daniel, J. Morris, S. M. Siniscalchi, Y. Tsao, and Y. Wang. “Detection-Based ASR in the Automatic Speech Attribute Transcription Project.” *INTERSPEECH*, Antwerp, Belgium, 2007.

I. Bromberg, J. Morris, and E. Fosler-Lussier. “Joint Versus Independent Phonological Feature Models within CRF Phone Recognition.” *HLT-NAACL*, Rochester, NY, April 2007.

## **FIELDS OF STUDY**

Major Field: Linguistics

Studies in:

Computational Linguistics	Prof. Eric Fosler-Lussier, Prof. Chris Brew
Laboratory Phonology	Prof. Mary Beckman

## TABLE OF CONTENTS

	<b>Page</b>
Abstract . . . . .	ii
Dedication . . . . .	iv
Acknowledgments . . . . .	v
Vita . . . . .	viii
List of Tables . . . . .	xiv
List of Figures . . . . .	xviii
Chapters:	
1. Introduction . . . . .	1
1.1 Language Modeling for ASR . . . . .	4
1.2 From Words to Morphemes . . . . .	5
1.3 Cross-Dialectal Usefulness . . . . .	7
1.4 Short Vowels . . . . .	8
1.5 Method . . . . .	8
1.6 Contribution . . . . .	10
1.7 Structure of the Dissertation . . . . .	10
2. Arabic Morphology . . . . .	12
2.1 Derivational Morphology: Roots, Patterns, and Vocalisms . . . . .	12
2.2 Inflectional Morphology: Affixes and Clitics . . . . .	15
2.3 The problem of vocabulary growth . . . . .	19

3.	Previous Approaches to Sub-Word Language Modeling . . . . .	22
3.1	Introduction . . . . .	22
3.2	Deriving Sub-Word Units using Linguistic Knowledge . . . . .	23
3.3	Deriving Sub-Word Units using Statistical Information . . . . .	29
3.4	Where the proposed model fits in . . . . .	32
4.	Building Language Models for MSA . . . . .	34
4.1	Morpheme Derivation . . . . .	35
4.2	N-gram counting . . . . .	41
4.3	LM Estimation . . . . .	41
4.4	Refining the LM probabilities with EM . . . . .	43
4.5	Alternative Language Models . . . . .	46
4.5.1	Word-based LM . . . . .	46
4.5.2	Affix-derived morpheme LM . . . . .	47
4.5.3	Morfessor-derived morpheme LM . . . . .	48
4.5.4	MADA-derived morpheme LM . . . . .	49
4.6	Description of the Training Text and Pre-processing Steps . . . . .	51
4.7	Text-Based Language Model Evaluations . . . . .	54
4.7.1	Coverage . . . . .	57
4.7.2	Average Negative Log Probability . . . . .	60
5.	Testing the MSA Language Models for Automatic Speech Recognition . . . . .	69
5.1	Acoustic Model Training Regime . . . . .	70
5.1.1	Word-Based Pronunciation Modeling . . . . .	71
5.1.2	HMM training . . . . .	73
5.1.3	Decoding . . . . .	75
5.1.4	Pronunciation Models for Decoding . . . . .	76
5.2	Decoding with the Morpheme-Only LMs . . . . .	78
5.3	Developing Morpheme+Word Language Models . . . . .	79
5.3.1	Morpheme+Word N-grams for Stem- and Affix-Based Models . . . . .	79
5.3.2	Morpheme+Word N-grams for MADA-Based Models . . . . .	81
5.3.3	Morpheme+Word N-grams for Morfessor-Based Models . . . . .	81
5.4	Decoding with Mixed Word and Morpheme LMs . . . . .	83
5.4.1	Weight Application in Morpheme+Word Models . . . . .	87
5.5	Analysis of OOV prediction . . . . .	88
5.6	Enhancing Word-Based Decode Lattices with Morphemes . . . . .	92
5.7	Discussion . . . . .	95

6.	How Regional Dialects Differ from MSA	104
6.1	Understanding Diglossia	105
6.2	Phonology	105
6.3	Morphology	109
6.4	Lexicon	111
6.5	Syntax	113
6.6	Relationship to Natural Language Processing	115
7.	Previous Approaches to Modeling Regional Arabic Dialects	117
8.	Building and Testing the Levantine Language Models for ASR	122
8.1	Text-based Language Model Analysis	124
8.2	Levantine Data Training Regime	127
8.3	Speech Recognition Results	130
8.4	Rescoring Word Lattices with Morpheme Models	133
8.5	Implementing the EM algorithm with LCA data	134
8.6	Discussion	135
9.	Previous Research on Short Vowels	138
10.	Prediction of Short Vowels with Stem-Based FSMs	145
10.1	Implementation of Short Vowel Predictions	146
10.2	Counting N-grams over Voweled Morpheme Sequences	148
10.3	Pruning the Short Vowel Hypotheses	149
10.4	Building a Vowel-Enhanced Pronunciation Model	151
10.5	Discussion	152
11.	Conclusion	153
11.1	Morpheme Modeling for ASR, Arabic Dialects and Short Vowel Predictions	153
11.2	Suggestions for Future Work in Arabic ASR	157
11.3	Other Applications of Morpheme-Based Language Models	158

Appendices:

A. Counting n-grams . . . . .	161
A.1 Proof of the Efficient N-gram Counting Method . . . . .	162
A.2 Implementing the Efficient Counting Algorithm . . . . .	167
Bibliography . . . . .	171

## LIST OF TABLES

Table	Page	
2.1	Reproduction of the <i>شهر</i> /ʃhr/ entry in Wehr [1994] (abridged). The root can combine with patterns and vocalisms to produce words or stems with different definitions. . . . .	16
2.2	An example of Arabic inflectional morphology. Transliterations and Arabic affixes are written in IPA. Modern Standard Arabic uses affixes to express inflectional changes that are expressed using additional words rather than affixes in English. . . . .	17
4.1	Stem Patterns for Arabic stems, <i>R</i> stands for Root letter, <i>p</i> stands for tah marbuteh. One verbal classification for each stem pattern is given, although multiple classifications are possible for some patterns. . . . .	37
4.2	Alphabets used for finding stems in Arabic words. See Table 4.4 for the mapping between these symbols, Buckwalter encoding, and Arabic script. . . . .	38
4.3	The morpheme decomposition hypotheses of each method for the word <i>AlmVtrwn</i> , “the purchased goods,” according to all four morpheme decomposition techniques. . . . .	50
4.4	Mapping between Arabic script, the Buckwalter character transliteration, and the modification on that (Heintz) used for representing these characters to the FSM and LM tools. (Linguistic Data Consortium [2008]). . . . .	53
4.5	Size of each language model, trained on all of the MSA language modeling data, with Kneser-Ney discounting. All morpheme models except for the affix model encode the data more efficiently than the word model, with fewer unigrams. Pruning affects the number of bigrams and trigrams in each model. . . . .	56

4.6	Percent coverage of the test texts by each model, relative to each n-gram order. While unigram coverage is mostly equivalent across models, the morpheme models generally have better coverage of higher-order n-grams.	57
4.7	Percentage of word-based OOVs obtainable by combining the terms within each morpheme model. More than half of the out-of-vocabulary words can be recovered by all of the morpheme models.	58
4.8	Percentage of OOV n-grams for which a backoff n-gram (one order lower) and backoff weight exist in the model.	59
4.9	Average Negative Log Probability of the test set as calculated by each language model.	63
4.10	The average negative log probability of the test set for trigram models built on transcript data, and refined using the method described in Section 4.4.	67
5.1	Word Accuracy on the development set using word-only and morpheme-only models. The morpheme-only models are inferior to the word-only model: all morpheme results are significantly different from the word results at $p \leq .0001$ .	78
5.2	Word accuracy results on the development and evaluation sets with the morpheme+word models. The word model performs the best in all conditions - development and test, transcript and alltext language models.	83
5.3	Results on the development and evaluation sets of decoding with mixed morpheme+word language models in which illegal morpheme sequences are included with zero probability. * signifies significantly different from the word model, $p \leq .01$ . Many of the morpheme models now perform with accuracy equal to the word model.	84
5.4	Word accuracy scores on the development set for LMs that use different weight applications on sentence-wide lattices before n-gram counting. The <i>a priori</i> application of even weights across morpheme arcs was not useful; using no weighting scheme or a statistically-informed weighting scheme produces increased word accuracy.	87

5.5	How each word+morpheme model uses morphemes in predictions for the 9480-word evaluation set. The affix model predicts morphemes most often, and the morphemes that the affix model predicts are most likely to coincide with a word in the utterance. . . . .	89
5.6	Description of how the 538 OOV tokens in the evaluation set are handled by each of the word+morpheme models. The stem and affix models replace OOVs with morphemes the most, but only in $\frac{1}{4}$ of OOV instances. Rarely do morphemes recover OOVs successfully. . . . .	91
5.7	Word accuracy result for rescoreing the word-based decode lattices with morpheme+word language models. Rescoreing gives significantly better ( $p \leq .01$ ) results than words for the transcripts-only model, but the words-only lattice expansion to alltext is significantly better than rescoreing with the alltext morpheme+word LM. . . . .	94
5.8	Comparison across previous research of word accuracy improvements using morphemes, relative to comparable word models within the same study. Different combinations of system complexity, language form, and morpheme derivation method produce a large variety of results. . . . .	97
6.1	Vowel Inventory of Modern Standard Arabic (Holes [1995]) . . . . .	107
6.2	Vowel Inventory of Levantine Conversational Arabic (Hussein [1991]) . . .	107
6.3	Consonant Inventory of Modern Standard Arabic and Levantine Conversational Arabic, as presented in Holes [1995] . . . . .	108
6.4	Conjugation of the root /drs/ with Pattern 1 template, giving the perfect tense form for all person/number/gender combinations in both Levantine (Hussein [1991]) and Modern Standard Arabic (Haywood and Nahmad [1965]). . . . .	110
6.5	Vocabulary differences and similarities for some common terms in Modern Standard Arabic and Levantine Conversational Arabic, as defined in Hussein [1991], McCarus [1979], and Alish [2000], spelled with MSA orthography and in IPA symbols. . . . .	113

8.1	Size of each of the language models built on four LCA corpora. The data and therefore the language models are much smaller than for the MSA data. Morpheme models generally encode the data more effectively than words. *The affix model was pruned in order to encode it as an FSM. . . . .	124
8.2	Percentage of n-gram tokens in the test set that are included in the language model. The Morfessor model has the best coverage for all n-gram orders. . . . .	125
8.3	Percentage of OOV words that are made up of morphemes in the given model. The affix model can potentially recover the most OOVs. . . . .	126
8.4	Percentage of OOV n-grams for which a backoff n-gram (one order lower) and backoff weight exist in the model. . . . .	126
8.5	Average negative log probability of a development set as determined by summing over all paths. The Morfessor and MADA models encode the test data more effectively than the Stem and Affix models. The word model has the lowest ANLP for all n-gram orders. . . . .	127
8.6	Word Accuracy scores on the development set using transcript-only language models and combined-corpus language models. The word model is most effective. . . . .	130
8.7	Word Accuracy for LCA data, using only the acoustic modeling transcript data for language modeling. The word model is still most effective. The Morfessor and MADA models are slightly better than the Stem and Affix models in some cases. . . . .	131
8.8	Rescoring the word-based decode lattices with morpheme-based language models. Rescoring is effective to make the morpheme models achieve scores equal to the word model, but not better. . . . .	133
8.9	ANLP and Word Accuracy on LCA data using stem-derived morphemes with different weighting schemes. As with MSA data, no weighting or EM-based weighting is more effective than the evenly-applied weights. . . . .	135
10.1	Several examples of how short vowel sequences can be paired with specific stem patterns. The colon ‘:’ represents a geminate consonant. The short vowels are /a/, /i/, and /u/. The /n/ represents nunation. . . . .	146
A.1	All paths of $A$ , all have weight 1. . . . .	168

## LIST OF FIGURES

Figure	Page
1.1 The word <i>AlkitabAn</i> meaning <i>the two books</i> . The root letters are <i>k</i> , <i>t</i> , and <i>b</i> . The pattern is Root1+i+Root2+a+Root3. . . . .	7
1.2 The word “AlkitabAn” encoded as an FSM with each connection describing one morpheme. . . . .	9
2.1 Growth in vocabulary of Arabic and English whole words and stems (Kirchhoff et al. [2006]). There are many more unique word forms in Arabic than in English, but the difference is abated when we first stem the Arabic words. There are many more unique affixes and combinations of affixes, so that stemming has a more dramatic effect in Arabic than in English. . . .	20
4.1 Stem-finder transducer for the stem pattern AstRRR, representing the perfect tense of a verb in paradigm X. The prefix alphabet for this example is {A,t,s,w} and the suffix alphabet is {A,p,y,h} (A represents alif). The root alphabet includes any letter (‘...’). Affixes may be any length. The output will include brackets around the stem portion of the word, which includes ‘Ast’ and the following three root letters. . . . .	38
4.2 The union of the decompositions of the word ‘wAstslmA’. The path labeled 0-1-8-9 is the output of composition with the transducer in Figure 4.1; the other paths are the output of composition with other stem-finder transducers. . . . .	40
4.3 An affix-finder transducer including a subset of the possible prefixes, suffixes, and stem letters. Composition with a word will be constrained by the sequences of letters at the word boundaries; only those words with matching affixes will result in a morpheme sequence. . . . .	47

5.1	Pronunciation possibilities are generated with the Buckwalter Arabic Morphological Analyzer, which includes short vowels and geminates in its analyses. The pronunciations are between “voc” tags, and include phonemes like 'a' , 'u', and 'i', which represent short vowels. . . . .	72
5.2	A new pronunciation model is derived for each language model, always using BAMA to produce voweled pronunciation variants. . . . .	77
5.3	Each word in a two-word phrase has been decomposed into stem-based morphemes, and the whole-word arc added to each FSM. The whole-word arc receives half of the probability mass, and the remaining paths share the remaining probability mass evenly. . . . .	80
5.4	Each word on the decode lattice is decomposed into its component morphemes, encoded as a small FSM. The morpheme FSMs are concatenated in the same way as their word arcs were originally arranged. The acoustic model score of the word is applied to the initial arcs of the morpheme FSM. . . . .	93
10.1	An example stem-finder transducer with vowels included on the output tape. . . . .	147
A.1	One way to split up the string $w \in \Sigma^*$ . . . . .	166
A.2	The WFSA representing our corpus, all transitions have weight 1. . . . .	167
A.3	FSA representing the n-gram $x$ we wish to count occurrences of in $A$ . . . . .	167
A.4	The weighted transducer $T$ , as defined in <a href="#">Allauzen et al. [2003]</a> , that allows us to count the occurrences of $x$ of in $A$ . . . . .	168
A.5	The composition of $A$ and $T$ . . . . .	168
A.6	The WFSA in Figure A.5 having performed epsilon removal and projection onto the output symbols. . . . .	168
A.7	Composition of $A$ and $T$ in the log semiring after projection onto output labels and epsilon removal. We use log addition to sum in the log semiring; the exponent of the result equals 4. . . . .	169

## CHAPTER 1: INTRODUCTION

The combination of language and computer science comes together in the field of natural language processing (NLP). The research performed in NLP attempts to get computers to produce, understand, or otherwise usefully process the many languages spoken by people. In many cases, the research focuses on building a machine that performs some language task as well as humans can, regardless of whether the inner workings of the machine or software mimics human behavior. The behavior, for instance, might be transcribing speech: producing in text form the words encoded in a speech signal. In this dissertation, the goal is to produce a system that takes in an audio signal consisting of Arabic speech, and produce for the user a text representing the words contained in that signal. The Arabic language is chosen because of the unique linguistic challenges it poses to the speech transcription problem. The challenges are at the levels of the language's morphology, the way its use differs in casual and formal situations, and the special characteristics of its texts. It is these linguistic elements along with societal needs that have spurred on recent research into Arabic automatic speech recognition.

The conversion of speech to text in automatic speech recognition presents a series of linguistic and computational challenges in any language. Some of these relate to language modeling: studying large amounts of text to learn about patterns of words in a language. Word patterns in English reveal certain agreement principles, for instance, the sequence 'we have' is much more common than 'we has'. In German, a language model will reveal that verbs often come at the end of a sentence. These kinds of generalizations about a language

are helpful in natural language processing tasks such as automatic speech recognition. In each language for which NLP tools are built, new challenges arise. In Modern Standard Arabic and the regional dialects of Arabic, for example, three language-specific challenges arise:

1. The morphological properties of Arabic make it so that there are many more word forms related to each semantically coherent word than is the case for English. Looking at a written text, a word form is defined as a sequence of letters delimited by whitespace. For the purpose of a semantic inquiry, some linguists may consider the terms ‘fact’ and ‘facts’ to be essentially the same *word*, but they are different *word forms*. Unless the way in which distinct word forms are grouped into more abstract words is made explicit, an algorithm that counts words in a text will actually count word forms. Without grouping the word forms in some way, it is difficult to collect reliable statistics across Arabic word sequences. Two word forms that are semantically similar and have similar contextual distributions will be counted separately, and their contexts counted separately. Each *word form* will occur less often than the more abstract *word* to which it is related.
2. The regional dialects of Arabic differ in ways that hinder the sharing of linguistic resources. Building NLP tools usually requires some amount of written text. Because most of these languages are primarily spoken and not written, acquiring resources for NLP tools is difficult. Researchers would therefore find it useful to incorporate available texts from one dialect into NLP tools for another dialect, but differences between the dialects’ words, pronunciations, and grammar make the resulting tools unreliable in the new dialect.

3. The orthography (written form) of Arabic lacks information about some vowel sounds. In Arabic natural language processing, this is called the *short-vowel problem*. In order to build reliable acoustic models, the vowels are either given explicit symbols in the text, or the symbols representing consonants are modeled as consonant+vowel sequences. The former method produces more accurate speech recognition, but the process of re-introducing short vowels into the text is quite difficult (e.g. [Stolcke \[2006\]](#)). The missing vowels make the text data less sparse, which is beneficial: two word forms that differ only in their short vowels are very likely to be semantically and contextually related, and would likely be grouped by an algorithm designed to form coherent classes of word forms. However, without the grammatical information that the short vowels encode, important information is absent from the models, that is, the model becomes less specific. Furthermore, the acoustic model may not be reliable in predicting the correct short vowel, due to the duration and acoustic confusability of the sounds. A language model that includes short vowels may be able to correct mistakes made by the acoustic model.

The goal of this dissertation is to address all three of these Arabic language modeling challenges by introducing a novel way of splitting words into smaller parts. The units used to build the language models are *morphemes*, meaningful parts of words, rather than whole words. The morphemes are derived using a method and single grammatical resource that are useful across different dialects. The method of morpheme derivation also provides information to address the problem of missing vowels. This research builds on recent language modeling studies such as [Afify et al. \[2006\]](#), [Xiang et al. \[2006\]](#), and [Kirchhoff et al. \[2006\]](#). The new language models are tested in a speech recognition application.

## 1.1 Language Modeling for ASR

The decoding process of automatic speech recognition (ASR) involves three main steps: converting the speech signal to a sequence of hypothesized phonemes using an acoustic model, converting those phoneme sequences into hypothesized word sequences using a pronunciation model, and determining from the resulting list of phrases which one was intended, using a language model. For instance, using the acoustic model and pronunciation model, a speech recognizer may give a series of possible phrases as solutions to some segment of speech:

1. me pauls in this kitty
2. meet buzz instant getty
3. meatballs and spaghetti

A language model could then be used to choose the most likely of these intermediate outcomes. A language model is a table of statistics about how words co-occur in the language. Those statistics are used to give a score to each possibility. The third solution above would likely get the best score, because it best reflects the kinds of word sequences that are spoken in English. The statistics of word sequences are learned by counting groups of words called *n-grams*. For every set of two, three, or four words, a record is made of the word sequence, as well as how many times it occurs in a large corpus of text. The *n-grams* and their counts are assumed to comprise a probability distribution, and as such, *n-gram* counts are converted into probabilities. An allowance is made for new *n-grams* to occur under some circumstances. Taking these probabilities into account, the language model is able to assign scores to new phrases in that language ([Chen and Goodman \[1996\]](#)).

Using a language model to choose the best phrase from a list of possible phrases is a task performed often in natural language processing. While the end task of automatic speech recognition is considered here, language models are also used to similar ends in machine translation and natural language generation (e.g. [Kirchhoff and Yang \[2005\]](#), [White \[2004\]](#)).

## 1.2 From Words to Morphemes

The main language modeling challenge posed by Arabic is brought on by its complex *morphology*, the way that words are built up from smaller parts. In Arabic, as in English, most words can be broken down into morphemes. Three morphemes comprise each word: a prefix, a stem, and a suffix. Taken together, prefixes and suffixes are called *affixes*. For instance, in the English word *unthinkable*, there is a prefix: *un-*, a stem: *think*, and a suffix: *-able*. The affixes *un-* and *-able* are used in many English words with reliable meaning. There are two ways in which Arabic differs from English in its morphology: there are many more affixes, which change across regional dialects, and the stem morphemes have an additional internal structure.

The number of different affixes in Arabic leads to a large number of word forms: the combinations of prefix, stem, and suffix multiply as the number of affixes increase. Consequently, there are a larger number of word forms. In an Arabic text, a reader is likely to come across many more unique word forms than in a text of the same length in English. Statistics of word sequences are less accurate when repetition of words and word sequences is less common. There are fewer examples of each word form, therefore fewer contexts from which to learn about it. A reader is also more likely, in new text, to come across a word form that was previously unseen: the *out-of-vocabulary problem*. Compared

to whole words, individual morphemes tend to repeat more frequently and have less variation. Therefore, for learning statistics over pieces of the text, it is sometimes useful to break the text into morpheme-sized chunks rather than word-sized chunks. Building language models over morpheme sequences has been shown to be an effective way to reduce the out-of-vocabulary problem, including in Arabic (e.g. [Vergyri et al. \[2004\]](#)). While some other works show that modeling over morphemes is not necessary for Arabic (e.g. [Creutz et al. \[2007\]](#)), it remains an active area of research in the Arabic NLP community. In particular, tools that perform morphological decomposition in dialects other than Modern Standard Arabic are of interest ([Diab and Habash \[2007\]](#)).

While the use of morphemes in language model building is sometimes effective, the process of deriving morphemes from words is problematic. It is not always clear whether the first or last few characters of a word represent an affix or a part of the stem. This particular problem is directly addressed in this study by exploiting the inner morphology of the stems. In Arabic, a stem can be separated into what are called the *root* and the *pattern*. English words do not have this structure; it is a special property of words in Semitic languages like Hebrew and Arabic. The root is usually a sequence of three consonants and has some abstract meaning. The pattern is a template that defines the placement of each root letter among vowels and possibly other consonants. Interweaving the root and the pattern, as in [Figure 1.1](#), results in a stem with a specific meaning ([Haywood and Nahmad \[1965\]](#)).

The number of patterns in Arabic is finite and relatively small; there are about 30 patterns (if variations in short vowels are ignored). The algorithm proposed in this study searches each word for every pattern. Wherever a sequence of letters within the word matches a pattern, that sequence is labeled as a possible stem, and any letters on either

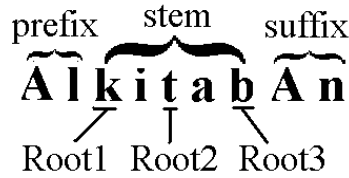


Figure 1.1: The word *AlkitabAn* meaning *the two books*. The root letters are *k*, *t*, and *b*. The pattern is Root1+i+Root2+a+Root3.

side of it as affixes. All of the possible prefix-stem-suffix combinations for each word are recorded and used in building the morpheme-based language model.

### 1.3 Cross-Dialectal Usefulness

Sharing data about specific word forms is not very useful cross-dialectally in Arabic (Kirchhoff et al. [2002]). Similar problems would arise in testing an American English speech recognizer with the English spoken in Scotland; the words and pronunciations are quite different in the two dialects. Tools that rely on affixes and stems common to Modern Standard Arabic for processing a dialect such as Levantine Conversational Arabic will produce many mistakes. However, the stem patterns mentioned above are reasonably consistent across dialects (Cuvalay-Haak [1997]), and are defined in Arabic grammar textbooks. As discussed above, this dissertation uses the *stem patterns* as the key to breaking down words into morphemes. Instead of sharing specific morpheme data across language types, only the method of morpheme derivation is shared. New stems and affixes emerge from the language data specific to each dialect.

## 1.4 Short Vowels

For most words in written Arabic, some of the sounds that are spoken do not appear as letters. If English were written in an analogous way, sentence 1 might be written as in 2.

- (1) Steve spoke with the librarian yesterday.
- (2) Stv spok wth th librarn yestrdy.

In some words, no vowel appears, or only the most emphasized vowel appears. In other cases, clues to pronunciation such as the final *e* in *spoke/spok* are missing from the written form. These phenomena occur in Arabic orthography. A fluent reader of Arabic has no trouble understanding an Arabic text lacking the vowel markings. Contextual clues disambiguate polysemous words or homophones; in the example above, we would guess that the first word is ‘Steve’ and not ‘Stove’, because stoves generally don’t speak with librarians. While human readers can re-insert vowels without a problem, the same is not true for computers. One of the challenges of training an Arabic automatic speech recognizer is to learn how to account for the unwritten sounds (Buckwalter [2004a]). One solution is to take advantage of the root-and-pattern morphology discussed above. The *patterns* used to describe the stem forms also define the short vowels that are part of that stem. Therefore, using the same method of searching for stem forms within words, predictions may be drawn about short vowels. These predictions may help to refine the training data and make the acoustic models more reliable.

## 1.5 Method

In this study, a special structure called a *finite state machine* (FSM) is used to combine the information about stem patterns with words in order to derive morphemes. An FSM

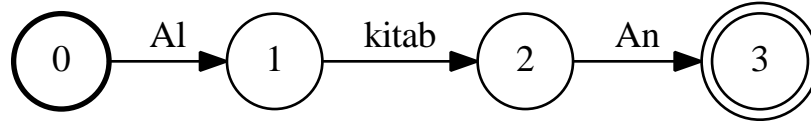


Figure 1.2: The word “AlkitabAn” encoded as an FSM with each connection describing one morpheme.

is a way of organizing information as a sequence of nodes and transitions. For example, Figure 1.2 shows a simple FSM encoding the word “AlkitabAn” in terms of its morphemes; the numbered circles represent nodes, and the lines between them are transitions. In moving from one node to the next, a new morpheme is found. At the last node, the FSM has accepted the whole word.

FSMs are used to encode words, stem patterns, morphemes,  $n$ -grams, and language models (Allauzen et al. [2003]). The stem pattern FSMs and word FSMs are composed with each other to determine what patterns are found in each word, thus efficiently searching every word for every hypothetical combination of prefix-stem-suffix. Short vowels may be added to the stem pattern FSMs, and through the FSM composition, short-vowel hypotheses are added to the word FSMs. Encoding the language model statistics in an FSM format allows for a tight integration between the morpheme derivation and the language model itself. This lends greater efficiency to the process. FSMs have been used in the past to study Arabic morphology (Al-Sughaiyer and Al-Kharashi [2004], Kiraz [2000], Beesley [1998]). This particular use of FSMs, starting with information only about stem patterns, is new, and its use for a tight integration between morpheme derivation and language modeling is also new.

## 1.6 Contribution

Using FSMs as a technological solution, this dissertation addresses three problems in Arabic language modeling. We attempt to address the Arabic out-of-vocabulary problem by modeling over morphemes rather than words, and deriving those morphemes using information about possible forms of the stem. This differs from other research in computational Arabic morphology that starts by defining possible affixes (Afify et al. [2006], Xiang et al. [2006]). By starting with the stem, the framework becomes cross-dialectally viable, because stem *patterns* are shared across dialects, even though particular affixes and stems may not be shared. Furthermore, by focusing on the form of the stem, the opportunity arises to address the problem of short vowels. The language models may be useful in tasks including speech recognition, but also machine translation and natural language generation, and extending language technologies to many dialects of Arabic. This work will also discuss the challenges inherent in the interactions between acoustic modeling and morpheme-based language modeling, which can counteract the proposed benefits of the morphemes. We will find that, despite sound theoretical groundwork, the effort made in deriving morphemes may not be worthwhile, given other parameters of the complex systems created for the speech recognition task.

## 1.7 Structure of the Dissertation

The first part of the dissertation addresses Modern Standard Arabic morphology and computational approaches that account for its regularities and idiosyncrasies in language modeling for ASR. Chapter 2 gives a discussion of the relevant parts of Arabic morphology, both inflectional and derivational. Chapter 3 includes a review of previous literature concerning the use of sub-word units in ASR. Chapter 4 describes in detail the FSM-based

morpheme derivation and experimental language model, as well as several baseline models. Chapter 4 also includes coverage and perplexity evaluations of these language models. Chapter 5 describes the process of building acoustic models for the Modern Standard Arabic ASR experiments, and describes the results of using the experimental and baseline language models for decoding.

The second part of the dissertation focuses on the problem of diverse Arabic dialects. Chapter 6 gives a more comprehensive view of the issues, and Chapter 7 gives a review of ASR literature addressing the problem. This is followed by a series of experiments comparing the novel language model to others as they relate to performing ASR in Levantine Conversational Arabic in Chapter 8.

Finally, the third part of the dissertation addresses the short vowel problem. A review of previous attempts to solve this problem is given in Chapter 9, followed by the solution given by the experimental language model in Chapter 10.

A discussion and conclusion are given in Chapter 11.

## CHAPTER 2: ARABIC MORPHOLOGY

The word-to-morpheme derivation proposed and evaluated in this dissertation takes into account both derivational and inflectional aspects of Arabic morphology. This chapter discusses how this morphology works to create verbs and nouns by combining root morphemes with pattern and vocalism morphemes. Arabic morphology and grammar are complex; many aspects, especially as regards irregular forms, will be left out of this discussion, as they are not accounted for in the computational framework. The explanation below is intended to help the reader better understand the derivation of sub-word units for use in language modeling described in Chapter 4. This discussion uses as its resources the Hans Wehr dictionary (Wehr [1994]) and a grammar textbook of modern written Arabic (Haywood and Nahmad [1965]).

### 2.1 Derivational Morphology: Roots, Patterns, and Vocalisms

An example letter sequence useful for illustrating the morphological processes is /شهر/ /ʃhr/<sup>1</sup>. This sequence is a *root*: a series of three, or sometimes four, letters (radicals) that signifies some broad set of concepts. The set of concepts expressed by words of a given root may be related, but may also vary widely. A root alone is not a fully-formed word; it has no part of speech, and is usually highly ambiguous in its meaning. In the theory of McCarthy [1981] and others, a root is a morpheme. In order to become a word, the root

<sup>1</sup>All Arabic examples will be written first in the Arabic script, followed by a transliteration into IPA. In this chapter, strings surrounded by slashes ‘/’ are morphemes, strings surrounded by brackets ‘[-]’ are complete words.

must be combined with two other morphemes, a pattern and a vocalism. For instance, the root /شهر/ /ʃhr/ can combine with the *pattern* /فعل/ /fʕl/, where ‘f’ represents the first radical, ‘ʕ’ represents the second radical, and ‘l’ the third radical. By combining the root with this pattern, the stem /شهر/ /ʃhr/ is realized. The *vocalism* is the third morpheme, crucial both to the pronunciation of the word and to giving the word its specific grammatical properties. The vocalism /a-a-a/, when combined with this root and pattern, forms [شَهَرَ] [ʃahara], now a fully-formed word with a dictionary-defined meaning: “made well-known, famous, renowned, notorious.” The verb is in the perfect tense, and agrees with a subject that is third person, singular, and masculine.

Each pattern and vocalism is part of a paradigm. There are approximately 14 paradigms, each with its own pattern and set of vocalisms. These paradigms are integral to the morphological processes of Arabic. The choice of paradigm determines the shape of the word (regarding whether there are consonants in the stem other than the root letter) as well as the meaning of the word. The pattern /فعل/ /fʕl/ used in the last example is that of the first *paradigm*, also considered a base form, the pattern from which all others are derived.

If the root /شهر/ /ʃhr/ instead combines with the pattern of paradigm III, /فاعل/ /fa:ʕl/, the letter alif is inserted between the first and second radicals to derive the stem /شاهر/ /ʃa:hr/. Together with the same vocalism as before, the word [شَاهَرَ] [ʃa:hara] is created, a perfect tense word with the meaning “hired or rented on a monthly basis”. The short vowel after the first radical is assimilated into the long vowel of the pattern.

By combining the same root with the pattern of paradigm VIII, /افتعل/ /ʔiftʕl/, and the same vocalism, [اشتهر] [ʔiʃtahara] is derived, with the meaning “to be or become well-known, famed, famous, to be known; to be widespread, common”. Here the first short

vowel comes after the ت /t/ of the pattern because the pattern has caused a change in the syllable structure: the letters ا /ʔI/ and ت /t/ inserted around the first radical ف /f/ cause the ف /f/ to be at the end of the syllable, rather than the beginning. The short vowel is therefore moved to the second syllable.

Some of the patterns have a predictable semantic effect on the word. Stems formed from paradigm II are often transitive or causative, for instance, the word عَلَّمَ [ʔalama] “to know” in paradigm I becomes عَلَّمَ [ʔal:ama] “to teach” in paradigm II. Stems formed with paradigm V tend to have a reflexive meaning: the same example becomes تَعَلَّمَ [taʔal:ama] “to learn” in paradigm V.

Other vocalisms are also possible, and these can change the word either inflectionally or derivationally. For instance, the vocalism /a-u/ is used to form stems in the imperfect tense. This vocalism combined with شَهَرَ /ʃhr/ and the pattern فَعَلَ /fʕl/ produces the stem شَهَرُ /ʃharu/, meaning “to make famous...” in the imperfect tense. However, this stem is not yet a grammatical word; inflectional affixes must be added to specify the person, number, and gender of the verb. For instance, the prefix /ي/ /ja-/ is added to the stem to derive the third person singular masculine form يَشَهَرُ [jaʃharu].

Alternatively, the vocalism /a/ creates the verbal noun شَهْرٌ [ʃahr], which has several possible meanings: “announcement, declaration” as well as “new moon; month”. Further inflectional morphology may be used to form the plural. There are two different plural

forms for this noun; the form is dependent on the meaning. The word meaning “announcement” takes a regular or *sound* plural ending, the affix /ون/ /uwn/, to create [شَهْرُونَ] [ʃahruwn]. The word meaning “month” takes a *broken* form for its plural, which is [أَشْهُرٌ] [ʔa:ʃhur]. This is a case of inflectional morphology produced by a non-concatenative function: rather than adding affixes to either end of the word to produce the plural, a new pattern is combined with the root to change the form of the word.

Other vocalisms create yet other specific meanings of the word. For instance, the pattern and vocalism combination /مُفَاعِل/ /mufa:ʕal/, the active participle of paradigm III, together with the root /شهر/ /ʃhr/ produces [مُشَاهِرٌ] [muʃa:har], meaning “monthly salary.”

As a further exploration of the ways in which a single root can be transformed into many different words, Table 2.1 reproduces the entries for the root /شهر/ /ʃhr/ as given in the Hans Wehr Arabic-English Dictionary, annotated with information about patterns, vocalisms, and pronunciations.

## 2.2 Inflectional Morphology: Affixes and Clitics

Just as the prefix /ي/ /ja-/ attaches to the imperfect stem /شهرُ/ /ʃharu/ to create the third person singular masculine form of the verb, other affixes may be added to achieve other grammatical details. Inflectional affixes are used to encode person, number, gender, tense, and mood information on verbs, and gender, number, definiteness, and case information on nouns. Affixes are a closed class of morphemes, and they encode predictable information. In addition to inflection, cliticization is commonplace in Arabic text. Prepositions, conjunctions, and possessive pronouns are all expressed as proclitics or enclitics.

Stem	Pattern	Vocalism	Pronunciation	Meaning
شهر	I, فعل [fʕl]	a-a	ʃahara	to make well-known, famous, renowned, notorious
شهر	II, فَعَلَ [fʕ:l]	a-a	ʃahhara	to revile publicly, expose, pillor, denounce
شَاهِر	III, فَاعِل [fa:ʕl]	a-a	ʃa:hara	to engage on a monthly basis, rent by the month
اشهر	IV, افعل [ʔa:fʕl]	a-a	a:ʃhara	to make known, proclaim, announce; to unsheathe, draw; to sell at auction
اشتهر	VIII, افتعل [ʔa:ftʕl]	a-a	ʔa:ʃtahara	to be or become well-known, famous, notorious; to be known; to be widespread, common
شهر	I, فعل [fʕl]	a-u	ʃahr	announcement, declaration
شهر	I, فعل [fʕl]	a-u	ʃahr	new moon; month
شهري	I, فعل [fʕly]	a-ij	ʃahrij	monthly
شهرياً	I, فعل [fʕlyɑ]	a-ij-a:n	ʃahrija:n	per month, by the month
شهرية	I, فعل [fʕl]	a-ij	ʃahrija	monthly salary
شهرة	I, فعلة [fʕla]	u	ʃuhra	reputation, fame, celebrity; notoriety; surname
شهير	II, فَعِيل [fʕi:l]	a-ij	ʃahijr	widely known, well-known, famous; notorious
اشهر	IV, افعل [ʔa:fʕl]	a	ʔa:ʃhar	better known, more widely known
تشهير	II, تفعيل [tfʕi:l]	i	taʃhijr	public exposure, pillorying
مشاهرة	III, مفاعلة [mfa:ʕla]	u-a	muʃa:hara	monthly salary
اشهار	IV, افعال [ʔa:fʕl]	i-a:	ʔiʃha:r	announcement, proclamation, declaration; auction
اشتهار	VIII, افتعال [ʔiftʕa:l]	i-a:	ʔiʃtiha:r	reputation, fame; notoriety
مشهور	I, مفعول [mfʕu:l]	a-uw	maʃhuwr	well-known, famous; notorious; a celebrity
مشهر	II, مفعّل [mfʕ:l]	u-a-a	muʃahhar	well-known, famous, celebrated; notorious

Table 2.1: Reproduction of the /شهر/ /ʃhr/ entry in Wehr [1994] (abridged). The root can combine with patterns and vocalisms to produce words or stems with different definitions.

Arabic	Transliteration	Translation	Arabic Affixes	English Affixes
أدرس	?adrusu	I study	a-	
ندرس	nadrusu	we study	na-	
تدرس	tadrusu	you (ms) study	ta-	
تدرسين	tadrusina	you (fs) study	ta- , -ina	
تدرسان	tadrusa:n	you (dual) study	ta-, -a:n	
تدرسون	tadrusun	you (mp) study	ja-, -n	
تدرسن	tadrusna	you (fp) study	ta-, -na	
يدرس	jadrusu	he studies	ja-	-ies
تدرس	tadrusu	she studies	ta-	-ies
يدرسان	jadrusa:n	they (dual) study	ja-, -a:n	
يدرسون	jadrusun	they (mp) study	ja-, -n	
يدرسن	jadrusna	they (fp) study	ja-, -na	

Table 2.2: An example of Arabic inflectional morphology. Transliterations and Arabic affixes are written in IPA. Modern Standard Arabic uses affixes to express inflectional changes that are expressed using additional words rather than affixes in English.

An example of affixational, inflectional morphology in Arabic is shown in Table 2.2. The Arabic stem /درس/ /drs/, meaning *to study*, combines with the imperfect tense verb pattern and vocalism /فَعُلُ/ /fʔulu/ to form the imperfect tense stem /دَرُسُ/ /drusu/. This stem can be combined with 11 different combinations of inflectional affixes, creating as many unique word forms. The English counterpart stem, however, can take on the same meanings using far fewer unique word forms. Instead, English uses separate words - I, they, he, etc. - rather than affixes to express person and number. The verb itself has only two forms, *study* and *studies*.

Table 2.2 can be expanded with stems from the same root representing different tenses. For instance, the stem (and word) [دَرَسَ] [darasa] means *he studied*. Or, the root can be combined with a different pattern to obtain different meanings, for instance, to teach or to learn. Each of these stems can combine with the same or different affixes to create a similar number of varying word forms.

Adding a single clitic to the words in Table 2.2 will double the number of word forms. For instance, the word [أَدْرُسُ] [ʔadrusu], meaning *I study*, can take on the enclitic /ه/ /ha/ to form [أَدْرُسُهُ] [ʔadrusuha] *I study it*. Various clitics can be combined, both before and after the stem. This process quickly increases the number of possible word forms.

Some stems differ in ways that do not surface in the Arabic orthography. For instance, the pattern /فَعِلُ/ /fʔilu/ differs from /فَعُلُ/ /fʔulu/ only in one short vowel, which is encoded orthographically as an optional and normally unused diacritic. Thus, [أَدْرِسُ] [ʔadrisu] and [أَدْرُسُ] [ʔadrusu] are homographs, but not homophones. While this property helps decrease the number of written word forms, it contributes to the ambiguity in morphological analyses.

## 2.3 The problem of vocabulary growth

The combination of concatenative inflectional morphology, cliticization, and derivational morphology using roots and patterns results in a large variety of word forms. This abundance of word forms is particularly impressive in contrast to the slower growth in number of English word forms. Figure 2.1 and Table 2.2 show this contrast. Figure 2.1 displays the growth in number of unique word forms, in English and Arabic, as more speech tokens are encountered. It is clear that the number of unique Arabic word forms far exceeds that of English.

Many of the mistakes in automatic speech recognition for Arabic can be attributed to the large number of word forms. For instance, in spoken word recognition, the decoding process makes use of a language model to predict the words that best fit the acoustic signal. In a language model comprised of whole word forms, and assuming no post-processing is performed on the result of decoding, only terms defined in the language model will be included in the decoded hypotheses. Because of the large number of possible word forms in Arabic, it is highly probable that some of the words in a new acoustic signal will not have been present in the language model's training text, and incorrect words will be predicted at all of those points in the test signal. The more unknown words there are in the test signal, the more mistakes are made. Unknown words become more likely as the size of the language's vocabulary - the number of word forms - increases.

If the variation in word forms that is due to concatenative morphology is removed via *stemming*, the number of unique word forms decreases, dramatically in Arabic. Stemming is a process in which words determined to have the same stem or base form are grouped into a single word class. For NLP tasks such as information retrieval, statistics of sequences in

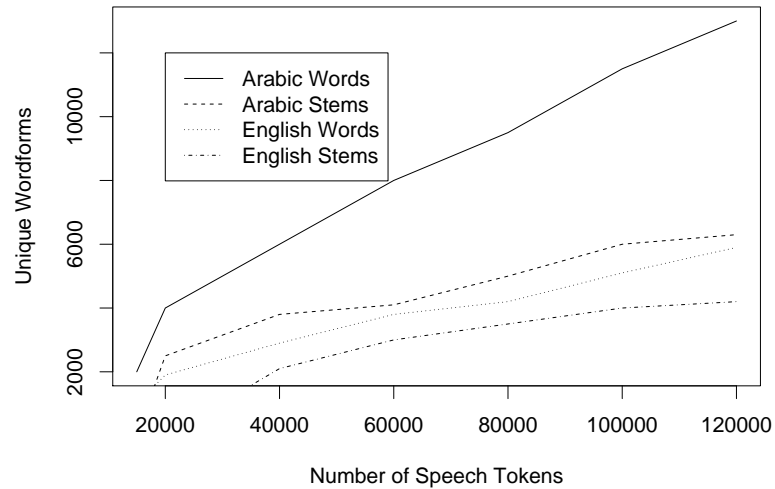


Figure 2.1: Growth in vocabulary of Arabic and English whole words and stems (Kirchhoff et al. [2006]). There are many more unique word forms in Arabic than in English, but the difference is abated when we first stem the Arabic words. There are many more unique affixes and combinations of affixes, so that stemming has a more dramatic effect in Arabic than in English.

a corpus can be collected over these word classes, rather than individual word forms. Alternatively, for tasks like language modeling, whole words may be broken up into their component morphemes, and statistics collected over sequences of these smaller terms. There is usually less variation among morphemes than among whole word forms. The stemming done in Arabic natural language processing tasks (such as in Afify et al. [2006], Xiang et al. [2006], and in this dissertation) may not provide prescriptively correct stems and affixes in every case, especially as irregular morphological processes change words in unexpected ways. Despite these potential errors, it is often possible to collect more reliable statistics concerning repeating patterns in a text when the hypothesized morphemes are used instead of whole words, due to their more frequent repetition.

In this dissertation, information about the morphology of Arabic is used to create a more reliable language model. This model should encounter fewer unseen forms. The units used to model the language are the more frequent and predictable morphemes. The statistics collected about the morphemes and their contexts will be more robust than those of words. Fewer terms in the test set will have been unseen in the training text, and the predictions made about the sequences of terms will be more reliable. As a result, once the predicted morphemes are re-concatenated to form words, there should be fewer errors made by a system that incorporates a morpheme-based language model, as opposed to a word-based language model. The exact process used to derive morphemes from words in Arabic uses information about the morphological processes described in this chapter. We will show that encoding these morphological processes in the form of finite state machines (FSMs) in order to derive morphemes for the purpose of language modeling is a highly practical method, and one that offers other advantages within the language modeling and natural language processing realm. These advantages include the ability to build a tool that is useful cross-dialectally, and a tool that has the ability to make predictions about the unwritten vocalisms. The exact method for using FSMs to derive morphemes from words is described in Chapter 4.

## CHAPTER 3: PREVIOUS APPROACHES TO SUB-WORD LANGUAGE MODELING

### 3.1 Introduction

Many of the world's languages exhibit a degree of morphological complexity that is not seen in English. Turkish (Guz et al. [2009]), Finnish (Hirsimäki et al. [2006]), Estonian (Creutz et al. [2007]), German (Geutner [1995]), and other languages in addition to Arabic, all use a wide variety of bound inflectional affixes, compound affixes, or compound words to create new terms. For instance, German speakers concatenate existing nouns to derive nouns with new meaning (Geutner [1995]). The same process occurs in Finnish, which also uses a large number of bound inflectional affixes to create phrase-like words specific to the semantic or syntactic context (Hirsimäki et al. [2006]). By counting each unique word form as a separate term in the lexicon, these morphological processes can produce a very large lexicon.

Large lexica are problematic for NLP tasks. The goal of NLP is to provide some analysis for each term in a given text or utterance. When a term found in a new text is unaccounted for in a given training text, it is called an out-of-vocabulary (OOV) term. OOV terms typically do not receive an analysis, as no information is given for them in the model. A successful way of approaching the OOV problem is to break the words into sub-word units (like morphemes) under the assumption that carefully chosen sub-word units have fewer unique types, thereby limiting the OOV problem. In addition, well chosen sub-word

units should repeat more often than whole words in a given text, making their frequency statistics more reliable.

However, replacing words with sub-word units can result in new problems: shorter units often cause n-gram models to have a smaller amount of contextual information, and may be harder to model acoustically or semantically. To counteract these deficiencies, model designers may change parameters such as the order of n-gram model or the size of the sub-word unit.

Sub-word units often provide a good, if partial, solution to the OOV problem in automatic speech recognition (ASR) (El-Desoky et al. [2009], Geutner et al. [1998], Pellegrini and Lamel [2009]). Given that solution, another question arises: if a training text is segmented at the word level, what is the best way of determining the sub-word units that make up those words? This chapter discusses previous work that uses both linguistic knowledge and statistical estimation to approach this question, as it pertains to Arabic and other morphologically rich languages.

## **3.2 Deriving Sub-Word Units using Linguistic Knowledge**

The tools described in this section were designed to automatically produce morphological analyses of Arabic words. The analyses may be useful for ASR language modeling, natural language generation, and machine translation, among other applications. In all of these resources, some computational structure, such as a finite state machine (FSM), encodes the information necessary for decomposing each word into morphemes, sub-word units that express semantic or syntactic meaning.

Habash et al. [2005] describe a multi-dialectal morphological analyzer that includes information about lexemes, morphemes, roots and patterns, orthography, and phonology.

Built for both analysis and generation, the multi-tape finite state machines include information about multiple dialects. In the resulting analyses, dialect switching can occur even word-internally. The implementation requires manually encoding hand-crafted rules and lexical entries. A great deal of linguistic knowledge is collected from texts or native speakers of the dialects in order to build a system that can perform automatic analyses. Once such a resource is complete, it can be of great use to the NLP community. However, updating and maintaining it can be quite challenging and time-consuming, and the information contained in the resource may not be generalizable past the dialects specified. If the goal is to document each dialect as fully and accurately as possible, then this approach may be quite successful. If generality among dialects is the goal, then a different approach to describing the dialects, perhaps using more abstract structures or definitions, should be taken.

Beesley [1998] and Kiraz [2000] also use linguistic knowledge and FSMs to create Arabic morphological analyzers. In Beesley [1998], the goal is to show that finite state morphology can be used to express the root-and-pattern morphology of Semitic words. The author advocates over-generation of morphological hypotheses using general rules and filtering the output with subsequent FSMs, rather than writing very specific rules, such as those for agreement between affixes. Roots and patterns are combined, a process called *interdigitation*, through concatenation and composition of FSMs. Interdigitation also requires careful definition of consonant, vowel, and verbal paradigm categories. As stated in Beesley [1998], “The rules are relatively difficult to write,” nevertheless, the analyses are achieved. Combining the multitude of rules with a sufficient dictionary results in the possible analysis of 72 million words. The rules must describe both regular and irregular morphology, and the dictionary must include roots, patterns, affixes, and their grammatical

meanings. This resource, like the previous, requires maintenance and upkeep for the dictionaries of roots and affixes, and may not generalize well to new dialects. The analyses of a single word, while possibly useful in their being very specific about grammatical functions and morphological composition, will be unranked, which could be problematic for downstream processing. It might be possible to rank the multiple analyses using the language modeling approach described in this thesis.

In Kiraz [2000] the goal is to encode the *autosegmental grammar* defined in McCarthy [1981] in multi-tape finite state automata. The method is to encode rules regarding how specific roots, patterns, and vocalisms combine. Affixational morphology as well as orthographic and phonological rules are also encoded in the same computational structure by introducing the notion of *tiers*. There are several tiers of lexical components - roots, patterns, and vocalisms - and another tier of surface components - words, orthography, and phonology. Alternating between tiers allows implementation of all of the rules in the proper order with accurate results. The previously stated issues regarding defining and maintaining the rules and dictionaries, as well as the lack of generalization to other dialects, apply to this method of analysis.

Similarly, Habash and Rambow [2005] use linguistic information to create a morphological analyzer for Arabic. Labels from the Penn Arabic Treebank (Maamouri et al. [2005a]) are used to train multiple classifiers for morphological analysis. The treebank was created with the aid of the Buckwalter Arabic Morphological Analyzer (BAMA) (Buckwalter [2004b]), which uses a dictionary of Arabic prefixes, stems, and suffixes, together with combination rules, to hypothesize all morphological analyses of a word. Human annotators marked the correct BAMA analysis for thousands of words in context in the treebank. The morphological analyzer in Habash and Rambow [2005] is based on the multiple BAMA

analyses and the correct choices given in the Penn Treebank, two rich sources of linguistic knowledge. Statistical information is also included via classifiers trained over the treebank labels. These classifiers then provide the best analysis of each word. The resulting tool, called *MADA*, has recently been used successfully in several ASR systems for both morpheme generation and diacritization (El-Desoky et al. [2009], Diehl et al. [2009], Vergyri et al. [2008]). There is an inherent limitation with this tool in that it was built on MSA data and definitions; the results may not be useful when applied to dialects other than MSA. The use of the *MADA* tool as applied to both Modern Standard Arabic and Levantine Conversational Arabic for building language models is explored in the experiments in this dissertation.

In Yaghi and Yagi [2004], a stem generator is built with linguistic knowledge of Arabic. Lists of roots from an Arabic dictionary and known stem patterns are combined with known morphosemantic and morphosyntactic rules to implement the processes of deletion, gemination, and affixation within words. Regular expressions, a type of FSM, encode the rules and are used to transform words to morphological analyses. Grammatical and morphological information remains intact throughout the process. Upon encoding all of the information and compiling the regular expressions, the tool generates a stem by combining a root with a pattern and transforms, or it hypothesizes a stem for a given word by comparing the word to the database of known combinations. No quantitative analysis of the tool is given aside from its ability to generate stems very quickly. The authors claim that the tool would be useful in NLP tasks that benefit from stem analysis such as word searching and concordancing. With its reliance on lists of roots, it is unlikely that this resource is generalizable beyond the specific dialect for which it was built.

[Attia et al. \[2009\]](#) build an annotation tool that allows annotators to manually choose the best morphological analysis of a given word. The tool automatically produces a complete set of analyses without over-generation, and gives additional lexical and semantic information as part of the analysis. A number of linguistic resources are put to use to perform the analyses. Each morpheme is classified into one of nine categories: prefix, derivative root, regular derivational pattern, irregular derivational pattern, fixed root, fixed pattern, Arabized root, Arabized pattern, or suffix. Each morpheme is also coded with a series of features describing its morphological and semantic properties; these are corpus-derived features. After linguistic knowledge is encoded, statistical estimation is used as well: an n-gram model with backoff estimates and the MAP (maximum a posteriori) technique serve to rank the multiple analyses of each word. The authors report an error rate on morphological disambiguation of less than 5%, and use the algorithm as the basis of an annotation tool. As is the case with MADA and other methods mentioned above that require large MSA resources and definitions, the tool is likely limited to the formal dialect only.

In the following studies, linguistic knowledge is combined with statistical estimation to derive sub-word units specifically for use in language modeling.

In [Afify et al. \[2006\]](#), the authors pre-define Iraqi Arabic affixes and use them to perform blind segmentation over the training corpus. Heuristics, including setting a minimum stem length and checking stem accuracy against the BAMA stem dictionary, limit the number of segmentations of each word. Linguistic knowledge of both Iraqi Arabic and Modern Standard Arabic is thus applied. Each word in the training text is mapped to a single segmentation, and the resulting segments comprise the language model and pronunciation dictionaries. [Afify et al. \[2006\]](#) find that, as expected, they are more successful at decoding with morphemes than decoding with words, as the morpheme vocabulary includes fewer

out-of-vocabulary terms in the test set. The authors also find that interpolating the morpheme model with a word model is successful; this is likely due to the increased contextual knowledge held by the word model. This method uses only a small amount of knowledge of a dialect to tailor the ASR system to that language variety, and uses the larger resources available for MSA to refine their models. Combining information sources in this way may be fruitful for other tasks or dialects for which linguistic resources are lacking. Using information from multiple dialects as resources for a low-resource dialect is useful in this study and may prove fruitful in other studies, a result also shown in [Kirchhoff and Vergyri \[2005\]](#).

[Emami et al. \[2008\]](#) use neural networks to include part of speech and vowel information as extra contextual data. Morphemes are derived using a three-part FSM that segments based on affixes, then refines the hypotheses based on dictionary entries and a segment-based LM. The linguistic knowledge in use here is a set of pre-defined affixes and a set of pre-defined stems. Statistical information is used to infer which are the most likely morphemes. The contribution of this work is the introduction of a neural network language model to Arabic ASR, but it is found that the gain in word error rate is negligible.

[Kirchhoff et al. \[2006\]](#) describes factored language models (FLMs), often considered to be state-of-the-art in Arabic ASR. FLMs encode many types of information for each word, including morphological information where available. During the language model estimation and subsequent decoding, backoff for unseen n-grams occurs using any of the features encoded in the FLM. The morphological information in [Kirchhoff et al. \[2006\]](#) comes from the lexicons provided by the LDC for the ECA corpus, and using an MSA root finder ([Darwish \[2002\]](#)). Because FLMs can encode diverse information and make the best use of that information at any point in an analysis via statistical algorithms like

genetic programming, they may prove to be very useful. However, in order to populate the FLM with information, it is necessary to either collect that information from available annotations or derive it with relevant tools. Useful information might include part-of-speech tags, semantic information, and morphological information. The lack of resources - annotated corpora and tools - for dialectal Arabic make this method somewhat less viable. It may be for this reason that FLMs have not been highly visible in the Arabic NLP literature since the cited publication.

### **3.3 Deriving Sub-Word Units using Statistical Information**

The following studies make more use of statistical estimation than linguistic knowledge to segment the morphemes that comprise each word. The goal of these methods is to produce morphemes useful to natural language processing tasks, meaning that strict grammatical accuracy may suffer; however, heuristics are often used to ensure that the analyses do not stray too far from theoretical “correctness”.

An example of combining a small amount of linguistic knowledge with statistical estimation is the work of [Lee et al. \[2003\]](#). Here, a small segmented corpus is used to bootstrap a morpheme finder. A trigram language model is built over a corpus wherein the words have been segmented into (prefix)-stem-(suffix) sequences. For each word in a new corpus, a set of segmentations are hypothesized. The segmentation with the best language model score is retained. To prevent over-generation, only those segmentations that correspond to a pre-defined set of affixes are permitted. Similarly, a list of stems can be referenced to further limit the segmentation possibilities. If a stem dictionary is used to constrain the initial segmentations, it can be grown by applying this algorithm to new text and adding new stems found to have a high probability. [Choueïter et al. \[2006\]](#) use this method of morpheme

derivation in ASR for Modern Standard Arabic, and find that these morphemes are useful when the decoding vocabulary is small and the OOV rate is large. Unsurprisingly, the effect is mitigated when there is a large decoding vocabulary that already counteracts the OOV problem. However, more recent studies including [Nguyen et al. \[2009\]](#) have found that the use of morpheme language models in Arabic ASR is useful even when the vocabulary is large.

[Hirsimäki et al. \[2006\]](#) and [Creutz et al. \[2007\]](#) use only statistical information to derive sub-word units from words. The algorithm, which will be explained in more detail in [Chapter 4](#), is language independent. First, a lexicon is derived from the word-segmented corpus. This lexicon is evaluated in terms of the bits required to encode it, and the corpus is evaluated by measuring its likelihood given the lexicon. Together, these measurements define the cost of the model. To find sub-word units, the words in the lexicon are recursively split, and the cost of the model is re-calculated. Segmentations that lower the model cost are retained. Segmentations that include units common to many words are likely to be retained, as they tend to lower the cost of the model. No linguistic knowledge is taken into account; every segmentation of every word is evaluated. This algorithm is called *Morfessor*. Because the segmentations are not necessarily the same as theoretically “correct” morphemes, the sub-word units are called *morphs* rather than *morphemes*. [Hirsimäki et al. \[2006\]](#) use this algorithm to split Finnish words into segments for ASR with good results compared to whole word and grammatically segmented models. [Creutz et al. \[2007\]](#) use this method for a number of other languages, including Egyptian Arabic. While the method works well for Turkish and Estonian, the method is less valuable for Egyptian Arabic, for which fewer data are available and for which the morphological proliferation of words is

less extreme. The word model is equally as effective on Egyptian Arabic as the statistically-derived *morph* model, achieving a word accuracy rate of 41.8%. In comparison, [Kirchhoff and Vergyri \[2005\]](#) use a more linguistically sophisticated HMM tagger for morphological analysis of the same Egyptian text, and perform automatic diacritization. They achieve 45.6% and 58.3% accuracy on two evaluation sets; it is unclear which of these sets is used in the [Creutz et al. \[2007\]](#) study. The lack of data for Egyptian Arabic surely was a factor for the low scores in both cases. The effectiveness of the Morfessor algorithm is compared to linguistically-informed morpheme derivation methods in Chapters 5 and 8 of this dissertation.

[Kiecza et al. \[1999\]](#) describe a morpheme-based language modeling technique for Korean, which, like Arabic, shows a high out-of-vocabulary rate when words are the modeling unit due to a high degree of inflectional morphology. Agglutination occurs over syllabic units, which tend to be just a few phonemes each. The OOV rate can be reduced to zero by modeling over syllables, but the resulting acoustic confusability is high. Therefore, the units used for recognition in [Kiecza et al. \[1999\]](#) are combinations of syllables. Syllables are merged by finding pronunciation transitions that occur frequently in the corpus, and merging those syllables that, when together, produce that particular transition. Merging of syllables stops when the OOV rate on a test text reaches a threshold of 5%. Here, the linguistic knowledge used is the set of Korean syllables. The pronunciation probabilities and merged units are corpus-based, and stochastically derived. Using these statistically-derived syllables in language models reduces the OOV rate dramatically and results in an increase in syllable accuracy of up to 6.5% absolute. Although the data-derived syllables may not have a clear semantic value, using heuristics related to acoustic confusability to define the

syllables proved valuable in this study. We learn from this that the theoretical “correctness” of a morphological analysis may not be crucial to the function of an NLP task. We can take advantage of this fact to avoid some of the challenges of creating and maintaining linguistically detailed resources described in the previous section. Instead, we can use a small amount of linguistic knowledge to create useful NLP resources. In some cases, the lack of linguistic detail will have the added benefit of creating tools that are generalizable to dialects or varieties of a language other than the one originally described.

In [Shalnova et al. \[2009\]](#) the goal is to implement a morphological analyzer for inflectional languages, agglutinative or fusional. The authors build trees that relate single-letter suffixes to each other, and then use entropy-related measurements to determine which suffixes ought to be merged. Applying these algorithms to a morphological tagging task produces F-measure scores as high as 99.5% for Russian and 61.2% for Turkish verbs, compared to much lower scores (21.0% and 56.7% respectively) when using Morfessor for the same task. As few linguistic resources are needed to implement these algorithms, the analysis can (in theory) be performed successfully with under-resourced languages as well.

### **3.4 Where the proposed model fits in**

The model proposed in this dissertation uses both linguistic knowledge and statistical estimation to derive morphemes from words, drawing on techniques used in the work just described. The linguistic knowledge is the set of verbal *patterns* that combine with roots to form stems. In addition, a pre-defined set of affix characters can be included to limit the amount of over-generation. These definitions are intended to be as dialect-specific as are available, but allowing for generalization to other dialects without major changes to the framework. Frequencies of segment sequences are collected to statistically determine

which are the most useful for further NLP processing. As in [Beesley \[1998\]](#), [Kiraz \[2000\]](#) and [Habash et al. \[2005\]](#), finite state machines are used to encode the linguistic information. The language model itself determines the relative validity of the hypotheses, and influences the derivation of new segmentations in subsequent processing. In this way, we hope to overcome the lack of generality of some of these systems, while simultaneously working to overcome the out-of-vocabulary problem and addressing the problem of lacking resources.

## CHAPTER 4: BUILDING LANGUAGE MODELS FOR MSA

In this chapter we present a novel method for deriving morphemes from Arabic words. We use morphemes rather than words for building language models in order to reduce the out-of-vocabulary problem (see Section 2.3). The morpheme derivation method proposed here will have the following properties: it will use the structure of the Arabic stem patterns to find possible stems and affixes within each word; by using the stem patterns, the method will be useful on a cross-dialectal basis; and, because the stem patterns also provide information about short vowel quality, we will be able to use this method to aid in short vowel prediction. This chapter will describe the morpheme derivation method in detail. The cross-dialectal and vowel-prediction properties will be discussed in subsequent chapters.

The derived morphemes are used to build n-gram language models (LMs), which are later applied to an automatic speech recognition (ASR) task. The derivation and LM-building steps are described in the following sections: morpheme derivation using stem pattern finite state machines (Section 4.1), n-gram counting over sentence-long morpheme finite state machines (Section 4.2), LM estimation given morpheme n-gram counts (Section 4.3), and refining of the LM probabilities by repeating the n-gram counting with weighted paths (Section 4.4). Alternative morpheme language models are described in Section 4.5. The data used for training all of the LMs is described in Section 4.6, and text-based evaluations of all of the LMs are given in Section 4.7. ASR-based evaluations are described in Chapter 5.

## 4.1 Morpheme Derivation

As discussed in Chapter 3, morphemes can be derived from words by first defining the affixes, then assuming that whatever character sequences fall between the affixes are stems. Here we will take the opposite approach, examining the center portion of the word to find a stem, then assuming character sequences that fall outside this stem are affixes. According to [Cuvalay-Haak \[1997\]](#), “the verbal patterns [stem patterns] of the Modern Arabic dialects are identical to those of Classical Arabic/Modern Standard Arabic in most cases.” Therefore this derivation method should prove to be useful in multiple dialects, without the need to redefine the character sequences that we search for in each word.

As a simple example of the stem derivation method, let *English'* be a language like English, except that word stems are defined by specific sequences of consonant and vowel letters. One stem form might be the sequence CVVC (where each *orthographic* vowel in the set {a, e, i, o, u} counts as a separate V). This stem pattern matches a sub-word unit in each word of the following subset of *English'*:

unsuitable	unseemly
unwearable	unseen
<b>fearless</b>	<b>square</b>
<b>weedless</b>	<b>soothe</b>

The word parts that fall outside of the bold ‘stems’ are possible affixes. Those that repeat - *un*, *able*, *ly*, *less* - are likely to be genuine affixes, regular in their usage and meaning. Additionally, the stems found with these repeating affixes - *suit*, *wear*, *fear*, *weed*, *seem*, *seen* - likely represent valid stems that can be found in semantically related word forms with different affixes. Less evidence is available in this list for the ‘affixes’ *s-*, *-e*, and *-he*, therefore, it is unclear whether they should be considered affixes, and whether the bold parts of those words - *quar*, *soot* - are really stems that will be found in related word

forms in English'. Of course, in real English, word stems are not restricted to specific CV sequences, especially orthographically. In contrast, the stems of Modern Standard Arabic are regular in their orthography and are constrained by these kinds of patterns, as discussed in Chapter 2. We use the stem patterns that define the Arabic stem character sequences to look for possible stems in each word. We then examine the frequency characteristics of the sequences of proposed stems and affixes to determine which terms are most likely to represent productive morphemes in the language.

We draw a set of 30 stem patterns from Haywood and Nahmad [1965], taking into account all patterns for the active and passive verbs and for active and passive participles of the ten most common verbal paradigms (verbal paradigms are described in detail in Chapter 2). The 30 unique stem patterns are shown in Table 4.1. Some of the paradigms share patterns; this is why not all verbal forms for each paradigm are described. Table 4.1 shows one possible classification of each unique stem pattern. These stem patterns represent the sequences of characters that we aim to find in each word.

The process of matching the stem patterns to words is performed with finite state machines (FSMs). We design one finite state transducer (*stem-finder transducer*) for each of the 30 unique stem patterns. An example stem-finder transducer is shown in Figure 4.1. The stem-finder transducers describe the sequences of root letters and other specific stem pattern consonants, such as A, n, s, t, and m, that occur in the augmented stem patterns shown in Table 4.1. An affix may occur on either or both sides of the stem, represented as self-loops on the first and last state. On the output tape of the transducer, the stem portion of the word is surrounded by brackets to help us identify where the morphemes begin and end.

Paradigm	Word Type	Stem Pattern
I	Perfect Verb	RRR
I	Verbal Noun	RRARp
I	Past Participle	mRRwR
II	Verbal Noun 1	tRRyR
II	Verbal Noun 2	tRRRp
II	Verbal Noun 3	tRRAR
II	Past Participle	mRRR
III	Perfect Verb	RARR
III	Imperfect Verb	tRARR
III	Verbal Noun 1	RRAR
III	Verbal Noun 2	mRARRp
III	Active Participle	mRARR
IV	Perfect Verb	ARRR
IV	Verbal Noun	ARRAR
V	Perfect Verb	tRRR
V	Past Participle	mtRRR
VI	Active Participle	mtRARR
VII	Perfect Verb	AnRRR
VII	Imperfect Verb	nRRR
VII	Verbal Noun	AnRRAR
VII	Active Participle	mnRRR
VIII	Perfect Verb	ARtRR
VIII	Imperfect Verb	RtRR
VIII	Verbal Noun	ARtRAR
VIII	Past Participle	mRtRR
IX	Verbal Noun	ARRRAR
X	Perfect Verb	AstRRR
X	Imperfect Verb	stRRR
X	Verbal Noun	AstRRAR
X	Active Participle	mstRRR

Table 4.1: Stem Patterns for Arabic stems,  $R$  stands for Root letter,  $p$  stands for tah marbuteh. One verbal classification for each stem pattern is given, although multiple classifications are possible for some patterns.

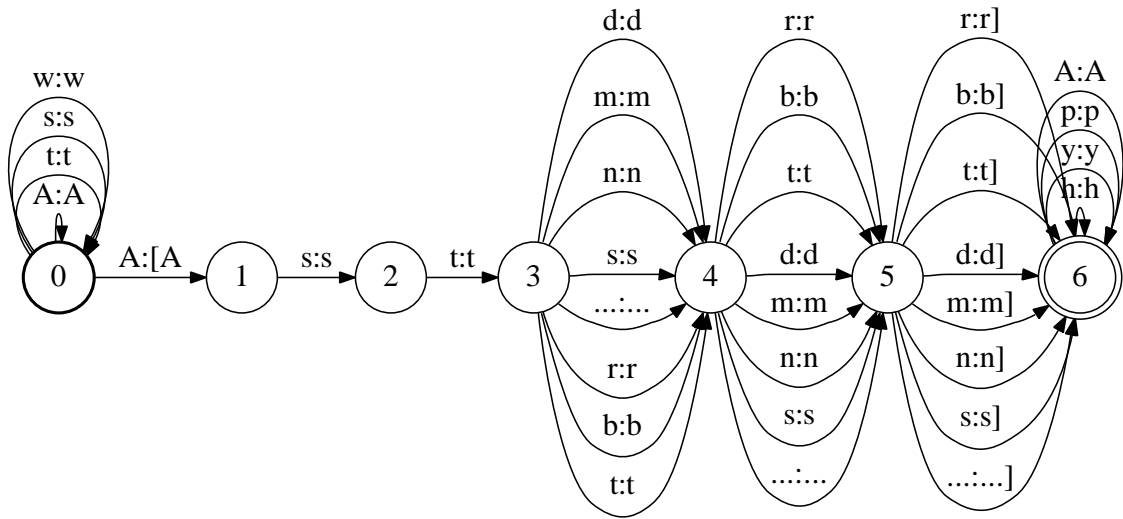


Figure 4.1: Stem-finder transducer for the stem pattern AstRRR, representing the perfect tense of a verb in paradigm X. The prefix alphabet for this example is {A,t,s,w} and the suffix alphabet is {A,p,y,h} (A represents alif). The root alphabet includes any letter ('...'). Affixes may be any length. The output will include brackets around the stem portion of the word, which includes 'Ast' and the following three root letters.

Prefix letters	Suffix letters	Root letters
A	A	b g v
l	n	D k z
b	h	H s A
f	m	L w E
k	k	T B M
w	y	d F Q
s	t	h J U
	w	l R Y
	Q	t V m
	p	x Z q
		G f y
		O j P
		S n
		W r

Table 4.2: Alphabets used for finding stems in Arabic words. See Table 4.4 for the mapping between these symbols, Buckwalter encoding, and Arabic script.

As shown in Figure 4.1, the affixes may be of any length (including length zero). In a scenario with no information about a language’s affixes, the affix alphabet may be unrestricted. The characters that fill the root slots may also be unrestricted. In practice, we take a more informed view by restricting the alphabet for affix letters. We have found that an unrestricted affix alphabet results in an unmanageable number of hypothesized morphemes. However, if little or no information were known about possible affixes, an unrestricted approach might be useful. The alphabets we use are shown in Table 4.2. We do not restrict the size of the possible affixes, or the sequencing of the affix characters. Our approach is thus less specific about affix definitions than those in studies such as Afify et al. [2006] and Zitouni et al. [2009], but we do assume some knowledge of possible affix shapes.

To find possible stems within a given word, we compose each stem-finder transducer with a finite state acceptor representing that word. The word acceptor describes the word with one transition per character. When composed with the word acceptor, each stem-finder transducer may match zero, one, or more instances of possible (prefix)-stem-(suffix) sequences within the word. All of the resulting paths are retained. The result of the compositions (one word acceptor composed with multiple stem-finder transducers) is therefore the enumeration of all possible (prefix)-stem-(suffix) decompositions for that word. The paths are modified by consolidating the arcs so that each morpheme is traversed over a single transition.<sup>2</sup> In the final representation, depicted in Figure 4.2 for the word *wAstslmA* “and they (dual) submitted,” each affix and stem are traversed over a single arc. In addition, the stems are identified by surrounding brackets. The FSM shown in Figure 4.2 is the union of all of the decompositions of a single word. If the word does not compose successfully

<sup>2</sup>The arcs are consolidated by traversing all of the single-character arcs, noting where the brackets are, and creating new FSMs with one morpheme per arc. In other words, this is not a defined FSM operation like composition or determinization.

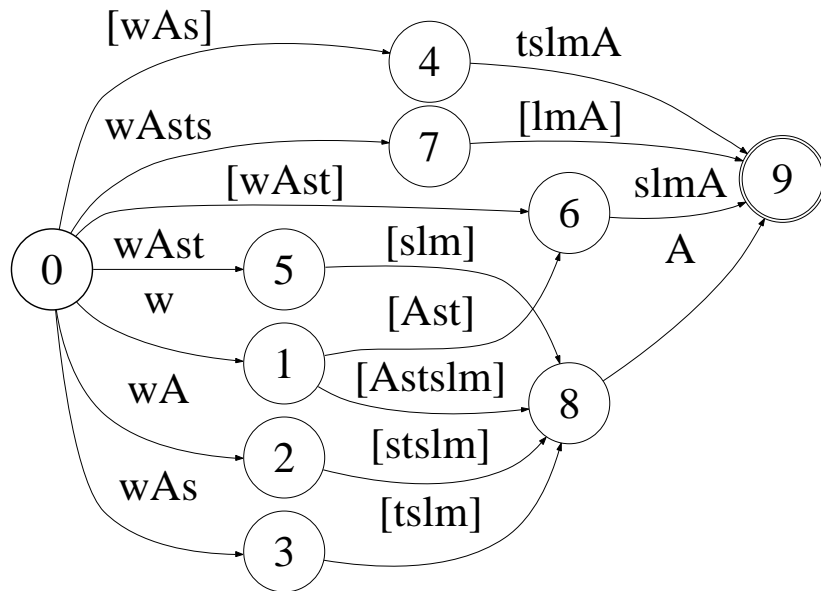


Figure 4.2: The union of the decompositions of the word ‘wAstslmA’. The path labeled 0-1-8-9 is the output of composition with the transducer in Figure 4.1; the other paths are the output of composition with other stem-finder transducers.

with any of the stem-finder transducers, then we create an FSM encoding the entire word on a single transition.

## 4.2 N-gram counting

Each word is decomposed separately by the stem-finder transducers. In order to collect frequency information about the proposed morphemes, we decompose each word in the vocabulary. Furthermore, we wish to collect information about sequences of morphemes, including sequences of morphemes across word boundaries. To do so, we calculate the frequencies of the n-grams that occur in a sentence-long concatenation of morpheme FSMs.

We transform a sentence in the language model training text into an FSM by composing each word in the sentence with the stem-finder transducers. We concatenate the resulting FSMs (similar to the one in Figure 4.2) in the order the words occur in the sentence. Then we count the morpheme n-grams over these sentence-long FSMs. The count of an n-gram in an FSM is defined, as in [Allauzen et al. \[2005\]](#), as the weighted sum of the number of occurrences of the n-gram in all paths of the FSM. [Allauzen et al. \[2005\]](#) describe an efficient method for calculating the weighted sum. We implement this efficient method, which is further described in the Appendix. For each language model described in this chapter, each path is equally weighted in the utterance-length FSMs.

## 4.3 LM Estimation

The counts collected from the sentence-length morpheme FSMs are used to estimate an n-gram language model. The goal of n-gram language modeling is to predict what item from the vocabulary is most likely to occur after a given history of  $n - 1$  items. We use the history of items  $(w_{i-1} \dots w_{i-n+1})$  to predict the next item,  $w_i$ . That is, we wish to

maximize the probability

$$P(w_i|w_{i-1} \dots w_{i-n+1}) \quad (4.1)$$

by choosing the item from the vocabulary that is most likely to follow the given history. A large value of  $n$  provides greater predictiveness, as more accurate predictions result from greater context. However, data sparsity is more severe when  $n$  is large; 4-grams are repeated less frequently than smaller order  $n$ -grams. In practice, a compromise is reached by using many differently-sized units (unigrams, bigrams, and trigrams) within the same model.

Another way to understand  $n$ -gram models is that we calculate the probability of a sequence having been produced by a given source. If we know all of the types of  $n$ -grams that the source has produced, and the frequency of each of those types, then we can assign a probability to each type:

$$P(w_i \dots w_{i-n+1}) = \frac{\text{freq}(w_i \dots w_{i-n+1})}{Y_n} * 100 \quad (4.2)$$

where  $Y_n$  represents the total frequency of all  $n$ -grams of size  $n$  in the vocabulary.

Given a training corpus, the language model assigns probabilities to each  $n$ -gram present in that corpus, based on their frequencies. The maximum likelihood estimates are usually refined using a *smoothing* method. The point of smoothing is to attribute some probability to possible  $n$ -grams that do not appear in the training data. A small amount of probability mass is subtracted from some or all of the  $n$ -grams present in the training data, and applied to any  $n$ -gram in the test data that was not seen in training. At test time, a *backoff* technique may be employed. If the language model is built on 4-grams, and an unseen 4-gram is encountered in the test set, backoff allows the model to apply the probability of the relevant trigram, bigram, or unigram instead. We use the SRILM toolkit (Stolcke [2002]) to estimate smoothed language models given the  $n$ -gram counts calculated over morpheme

sequences. We use Kneser-Ney discounting in all models described in this chapter. The calculation of the discount is described in [Stolcke \[2008\]](#).<sup>3</sup>

In addition, when the LM is estimated, less likely n-grams are pruned. We expect that morpheme hypotheses that do not fit the canonical definition will be pruned or given low probability in the model due to being less frequent than the more viable morphemes, as described above.

#### 4.4 Refining the LM probabilities with EM

The stem-finder transducers do not introduce any probabilities or ranking of the proposed morphemes. The initial n-gram counts assume that all paths through the sentence-long morpheme LM are equally likely (unweighted). The LM estimates describe which of the proposed stems and affixes are more likely to be productive morphemes in the language given the unweighted frequency information. As described below, we can apply these initial estimates to the n-gram counting algorithm, produce refined counts, and subsequently re-estimate the language model with these new weighted counts. This creates an Expectation-Maximization (EM) iteration, where with each new probabilistic language model, we can influence the estimated counts of the n-grams before creating a new language model.

To apply the language model estimates to the n-gram counting algorithm, we must first encode the LM as an FSM. Such an encoding is described in [Allauzen et al. \[2003\]](#), and summarized here. In a language model finite state machine, each node represents one n-gram  $w_i \dots w_{i-n+1}$ . A transition out of that node is labeled either with a new term  $w_k$  or a backoff symbol  $\phi$ . Following the first type of transition leads to a node representing the

<sup>3</sup>We use the `-ukndiscount` flag in SRILM.

n-gram  $w_k \dots w_{i-n+2}$ , that is, the previous n-gram with the most distant term dropped, and the new term added. Following the backoff transition leads to the n-gram  $w_i \dots w_{i-n+2}$ , the next lower-order n-gram. The nodes also define the probability of the n-grams. The backoff transitions define the backoff weight assigned to the lower-order n-gram. The  $\phi$  transitions are an extension of the usual definition of an FSM, in that they may only be traversed if none of the other transitions out of the given node are applicable. They are different from  $\epsilon$  transitions in that respect. The  $\phi$  transitions therefore implement an if-else statement within the FSM. The OpenFST toolkit ([Allauzen et al. \[2007\]](#)) includes the ability to use  $\phi$  transitions in its C++ libraries.

The following steps are iterated in order to refine the LM estimates:

1. For each sentence in the training text, decompose each word with the stem-finder transducers, and concatenate the resulting lattices to form a sentence-long morpheme FSM.
2. If a morpheme language model already exists, encode it as an FSM. Compose the sentence FSM with the language model FSM. This produces a sentence FSM with weighted paths.
3. Count n-grams on the sentence FSMs using the efficient counting method described in [Allauzen et al. \[2005\]](#). Collect the count estimates over all n-grams in all sentences in the training text.
4. Estimate a new language model from the n-gram counts.

At the end of each iteration, the new language model can be evaluated using any of the methods described in Section 4.7, and the process is repeated until convergence. For instance, we can continue refining the language model estimates using this method until the average negative log probability of a development text stops changing.

In the evaluations below, we include a model that has undergone two rounds of estimation: initial n-gram counts were calculated over unweighted FSMs, a language model estimated, then those model probabilities applied back to the FSMs for weighted n-gram counting. This was done with the stem-finder transducers over the whole of the training corpus. It is necessary here to further discuss the handling of unknown tokens. When the initial language model is estimated, not all morpheme sequences proposed in the sentence-wide FSMs may be included in the model, due to minimum-count pruning performed by SRILM. Therefore, when the corpus is decomposed the second time for composition with the LM, the sentence FSM may contain terms that do not exist in the LM. In this case, the composition with the LM fails outright. To avoid failure, we first find all terms in the sentence FSM that do not appear in the LM, and change their representation on the input and output tape. On the input tape of the sentence FSM, the unknown word or morpheme appears as <unk>. On the output tape of the sentence FSM, we append +unk to the original morpheme. When we compose the LM and the sentence, the <unk> symbol in the LM and its associated probability compose with the <unk> symbols on the input tape of the sentence FSM, while the output retains the original morpheme information. By including the morpheme and +unk tag on the output tape, we do not lose information about the morpheme or word on the sentence-FSM, nor do we lose information about where unknown words and morphemes occurred. Before performing the evaluations described below, we remove the +unk tag from all n-grams.

## 4.5 Alternative Language Models

In our proposed language model, the terms over which we calculate n-gram estimates are morphemes derived via stem definitions. We compare this stem-derived morpheme LM to several other language models, each of which represents a type of model used in the Arabic speech recognition literature. Each model defines a different type of term over which n-gram probabilities are estimated. These terms include:

1. words,
2. morphemes derived via affix definitions,
3. morphemes derived via character sequence statistics (Morfessor), and
4. morphemes derived by performing a data-driven, context-dependent, complete morphological decomposition (MADA).

Each of these models is described in turn below.

### 4.5.1 Word-based LM

For the word model, we build and count n-grams using whitespace-delimited words as our terms. Each unique word form counts as a separate vocabulary item. This is the usual way to build an LM for languages with little morphological variation or concatenative processes, such as English. In Arabic, morphological variation usually causes the size of the vocabulary to increase quickly with the number of tokens in a text. This causes the n-gram estimates to be less reliable, as each n-gram is seen fewer times, and causes the out-of-vocabulary rate to increase. Morpheme-based language models are often used to overcome these obstacles.

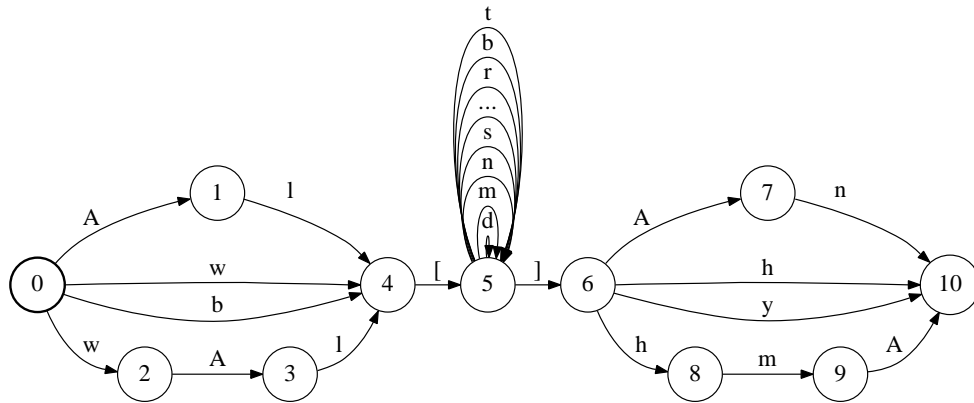


Figure 4.3: An affix-finder transducer including a subset of the possible prefixes, suffixes, and stem letters. Composition with a word will be constrained by the sequences of letters at the word boundaries; only those words with matching affixes will result in a morpheme sequence.

#### 4.5.2 Affix-derived morpheme LM

This LM is designed to replicate those built by [Afify et al. \[2006\]](#), [Lee et al. \[2003\]](#), [Sarikaya et al. \[2007\]](#) and others, who segment words into morphemes by defining a list of affixes that may occur at either word boundary. Usually some heuristics are used to constrain the word segmentations. Often, a language model is applied to choose the best segmentation. For our comparison model, we use the list of affixes defined in [Xiang et al. \[2006\]](#), and build FSMs similar to our stem-finder transducers. A subset of the final affix-finder transducer is shown in Figure 4.3. Each word is decomposed into morphemes based on the presence or absence of the defined affixes. The word may have zero or more prefixes, followed by a stem of any length and characters, followed by zero or more suffixes. As in the stem-derived morpheme model described above, the morpheme decompositions are

encoded as FSMs, and we count n-grams over FSM sentences using the efficient counting method described in the Appendix.

### 4.5.3 Morfessor-derived morpheme LM

We replicate the work in [Creutz and Lagus \[2005\]](#) by using *Morfessor* to derive *morph* segmentations from the words. The segments are called *morphs* rather than *morphemes* because they are derived through statistical estimation rather than linguistic knowledge. *Morfessor* uses the MAP (Maximum *a Posteriori*) algorithm to find the most likely morpheme boundaries. Given the lexicon of a corpus, each word is recursively broken down into all possible segmentations. Where  $M$  is a lexicon of morphs,  $W$  is a lexicon of words, each of which has  $n$  segments,  $\mu$  is a morph,  $f_\mu$  is a function of the frequency of a morph, and  $s_\mu$  is a function of the length of a morph, the model is defined as follows:

$$P(\text{corpus}|M) = \prod_{j=1}^W \prod_{k=1}^{n_j} P(\mu_{jk}) \quad (4.3)$$

$$P(\text{lexicon}) = M! * P(f_{u1} \dots f_{uM}) * P(s_{u1} \dots s_{uM}) \quad (4.4)$$

$$P(M|\text{corpus}) = P(\text{corpus}|M) * P(\text{lexicon}) \quad (4.5)$$

For each segmentation, the new cost of the corpus (Equation 4.3) and the cost of the lexicon (Equation 4.4) are combined to calculate the cost of the model (Equation 4.5), as described in [Creutz and Lagus \[2005\]](#). That is, the corpus cost is based on the product of the probabilities of each of the morphs in it, independent of the order in which they occur. The cost of each morph is determined by its token frequency in the segmented corpus, and its length. In our experiments, no priors are used to refine the morph frequency or length probabilities.<sup>4</sup> Those segmentations that lower the cost of the model are retained. Words

<sup>4</sup>Although the algorithm allows for external probability distribution functions to be defined for the length and frequency of morphemes in the corpus, [Creutz and Lagus \[2005\]](#) show these to have minimal effect, and so we leave them out for simplicity.

and morphs are recursively segmented until the decrease in the model cost reaches a given threshold. The algorithm requires no linguistic knowledge, and can therefore be applied to any language with a sufficient amount of textual data. The algorithm could be used to derive the morph probabilities of one dialect and apply them to segment another unknown, sparsely represented dialect.

We apply *Morfessor* to the vocabulary of the corpus, minus the singleton word types. This means the morphs are derived from the more common words, rather than relying on rare words for segmentations, resulting in more reliable morphs. We use word types rather than their token frequencies as the basis for the decomposition, as recommended in [Creutz and Lagus \[2005\]](#). The resulting dictionary is used to segment the entire corpus: each word is deterministically decomposed into its *Morfessor*-determined morphs, or where no decomposition has been defined (e.g. for singleton word types), the entire word is used. N-grams are counted over these morphs. We include morpheme- and word-boundary markers in the decomposition so that morphs can be correctly concatenated back into words for evaluation purposes.

#### 4.5.4 MADA-derived morpheme LM

We replicate the LMs described in [El-Desoky et al. \[2009\]](#) and [Diehl et al. \[2009\]](#) by using *MADA* ([Habash and Rambow \[2005\]](#)) to derive morphemes from words. The algorithm underlying the *MADA* software relies on a dictionary of MSA affixes and stems, tables defining their possible combinations, and a model that judges the most likely morphological decomposition given a word in context. First, all possible morpheme decompositions for a word as defined by the databases in the Buckwalter Arabic Morphological Analyzer ([Buckwalter \[2004b\]](#)) are enumerated. Each decomposition is accompanied by

Word	<i>AlmVtrwn</i>
Stem	Al [mVtrw] n
	Al [mVtr] wn
Affix	Al [mVtrwn]
	Al [mVtr] wn
	[AlmVtr] wn
MADA	Al -mVtr- wn
Morfessor	Al- -mV- -trwn

Table 4.3: The morpheme decomposition hypotheses of each method for the word *AlmVtrwn*, “the purchased goods,” according to all four morpheme decomposition techniques.

a detailed description of the part-of-speech and other grammatical properties of the decomposed word. Next, a support vector machine model, previously trained for MADA on broadcast news data, is used to determine which of those decompositions is most likely given the context and grammatical information. The most likely decomposition is marked by MADA. For each sentence in the corpus, we replace each word with its most likely decomposition, then count morpheme n-grams to estimate the MADA-based language model.

The MADA algorithm requires a substantial amount of linguistic knowledge to perform the morphological analyses, and annotated data to train the model that predicts the correct morphological decomposition. These resources have been utilized in building MADA, which is designed to work with MSA data. We expect the MADA-based morpheme derivation method to work well for the MSA data, as it was trained on MSA broadcast news. Its usefulness when applied to other dialects will be explored in Chapter 8.

Table 4.3 shows how the single word *AlmVtrwn* “the purchased goods” is decomposed by each morpheme derivation method.

## 4.6 Description of the Training Text and Pre-processing Steps

The training, development, and test corpora used for the MSA portion of this study, as well as the text preprocessing steps applied to the corpora, are described in this section.

The primary training text corpus for language modeling is the TDT4 Multilingual Broadcast News Speech, Text and Annotations, distributed by the Linguistic Data Consortium (Kong and Graff [2005]). We use the Modern Standard Arabic portion of the corpus, which includes 1,088 files of broadcast news text. The news sources included are *Agence France Presse*, *Al-Hayat*, *An-Nahar*, *Voice of America*, and *Nile TV*. All of the data files are news articles collected between October 2000 and January 2001. The first three sources are newspaper text, while the latter two are transcripts of radio and TV news programs. The chronologically last four transcripts are reserved as testing data (19K words), and the four previous to that are used as development data (17K words), leaving 1080 files (14 million words) for LM training. Note that the development and test data include transcripts only, not newspaper text.

The text files are formatted with SGML; the tags break up the corpus into articles and give information about the source and data of each story. The content is encoded as Arabic script in UTF-8, and thus readable as Arabic in many text editors or browsers. However, this encoding is not amenable to manipulation with the various morphological and language modeling tools we are working with to build our LMs, such as SRILM (Stolcke [2002]) and OpenFst (Allauzen et al. [2007]). We change the original texts from UTF-8 into a representation using only the 7-bit ASCII characters. We use the Buckwalter transliteration scheme, slightly modified to exclude characters that are incompatible with the FSM and LM software. Buckwalter encoding can be unambiguously reverted back to Unicode. The

mapping between the Arabic script, Buckwalter transliterations, and our transliterations is reproduced in Table 4.4.

We perform tokenization to separate punctuation from surrounding characters. We use periods to delimit the sentences when counting n-grams. To that end, we change all exclamation points, question marks, and colons to periods. Periods, quotation marks, and commas are surrounded with whitespace so that they are not interpreted as characters within a word. Otherwise, the words “ktb,” and “ktb” would be different, increasing the number of unique word forms and exacerbating the out-of-vocabulary problem. The forward slash appears exclusively in dates in this corpus and can cause problems with our downstream software, so we remove it entirely from the training corpus. Numbers that are attached to words (no whitespace between) are removed. The percent sign is also deleted. The Al-Hayat text uses the characters « and » rather than quotation marks; we convert them to quotation marks for consistency across the corpus.

The problem of inconsistently used hamzas is described in [Buckwalter \[2004a\]](#). The hamza symbol, representing a glottal stop in some places, and no phonological value in others, can be written with a number of different glyphs. These varieties are not used consistently, and would be a source of noise in our models. Any instance of hamza occurring at the beginning of a word or after the two most frequent prefixes are changed to the letter Alif. That is, any instance of  $\text{أ}$ ,  $\text{إ}$ , or  $\text{آ}$  occurring at the beginning of a word, following  $\text{أل}$  at the beginning of a word, or following  $\text{و}$  at the beginning of the word, are changed to  $\text{ا}$ .

Glyph	Buckwalter	Heintz	Glyph	Buckwalter	Heintz
ء	'	Q	ع	E	E
آ	—	W	غ	g	g
أ	>	U	ـ	-	-
ؤ	&	R	ف	f	f
أ	<	O	ق	q	q
أ	}	M	ك	k	k
ا	A	A	ل	l	l
ب	b	b	م	m	m
پ	p	p	ن	n	n
ت	t	t	ه	h	h
ث	v	v	و	w	w
ج	j	j	ى	Y	Y
ح	H	H	ي	y	y
خ	x	x	fatHatayn	F	F
د	d	d	fatHatayn	ð	e
ذ	*	L	kasratayn	N	N
ر	r	r	Dammatayn	K	K
ز	z	z	fatHa	a	a
س	s	s	Damma	u	u
ش	\$	V	kasra	i	i
ص	S	S	shadda	~	I
ط	D	D	sukun	o	o
ظ	T	T	dagger 'alif	'	C
ظ	Z	Z	waSla-on-alif	{	B

Table 4.4: Mapping between Arabic script, the Buckwalter character transliteration, and the modification on that (Heintz) used for representing these characters to the FSM and LM tools. ([Linguistic Data Consortium \[2008\]](#)).

## 4.7 Text-Based Language Model Evaluations

In this section, the stem-derived morpheme language model is compared to the alternative language models through a series of text-based evaluation measures, including coverage of a test set and average negative log probability. These measurements will give us initial insight into how the different models may perform in a task setting before building a speech recognition system on which to test them more thoroughly. These evaluations serve as a check that the models are properly built and that the theories we have about the relationships between words and morphemes are confirmed by the models. We also gain some insight into the differences between the morpheme models.

The test set is converted from words to the appropriate kind of morphemes for each language model before evaluation. For the stem, stem+EM, and affix models, we decompose each word in the test set using the same stem-finder or affix-finder FST that was used to decompose the training text. The decompositions of each word in the test sentence are concatenated into a sentence-long FSM. The sentence FSM is composed with the FSM representation of the stem- or affix-based language model, and the best path is returned. The best morpheme path of each sentence in the test text is used to calculate the coverage and average negative log probability statistics described below. For the MADA model, we use the word-to-MADA dictionary that was compiled during the training step. For each word in the test sentence, we extract from the dictionary all of the decompositions that were chosen for that word throughout the training text (usually there is a single decomposition, but for about 5% of the words, multiple decompositions were chosen throughout the text). The decompositions are compiled as FSMs, which are concatenated in order. The sentence-wide FSM is composed with the MADA-based language model FSM, and the best path is chosen. For the Morfessor model, since only one decomposition is possible per

word, we simply use the same word-to-morph dictionary used in training to convert each sentence into morphemes. In this way, we evaluate each morpheme language model on a text comprised of the correct type of morpheme.

Some pruning is performed when estimating the language models from counts. All models are pruned according to the minimum count indices provided by SRILM for Kneser-Ney discounting. The stem+EM model initially proposes far more unigrams than the stem model, due to application of weight to the morpheme paths, which makes some morphemes frequent enough to avoid pruning by the second model. In order to make more consistent comparisons between the stem and stem+EM model, we prune the stem+EM model to include only the same unigrams as the stem model. In addition, the affix model was pruned using entropic pruning in order for it to be a manageable size. No unigrams were pruned, and only as many bigrams and trigrams were pruned as to make the LM small enough to encode as an FSM. Although we prune the affix model to a larger extent than the others, it remains the largest LM, as shown in Table 4.5.

Table 4.5 describes the five language models in terms of how many n-grams of each order exist in each model. We see that the stem model has much fewer unigrams than any of the other LMs. We know that all of the 460K word types encountered in the training text can be created by concatenating these morphemes; therefore, the stem-derived morphemes provide the most compact encoding of the words. The Morfessor and MADA models also have fewer unigrams than the word model. The affix model, however, is highly permissive in its judgement of what is a possible morpheme, as it proposes more unique morphemes than words. This means that many words have been decomposed in multiple ways, or many of the derived morphemes are not repeated for many words. As mentioned above, the unigrams in the stem+EM model are pruned to match those in the stem model. The

Model	Unigrams	Bigrams	Trigrams
Words	461,650	4,672,078	1,328,770
Stem	99,353	843,016	731,600
Stem+EM	99,353	1,110,594	142,213
Affix	612,790	4,853,110	6,356,298
Morfessor	283,018	2,325,287	1,772,926
MADA	232,221	1,895,422	1,597,788

Table 4.5: Size of each language model, trained on all of the MSA language modeling data, with Kneser-Ney discounting. All morpheme models except for the affix model encode the data more efficiently than the word model, with fewer unigrams. Pruning affects the number of bigrams and trigrams in each model.

bigrams and trigrams are also pruned to include only these vocabulary items, and the LM probabilities redistributed among the remaining n-grams. The application of LM weights to the FSMs, unigram pruning, and subsequent LM estimation cause the stem+EM model to be quite different from the stem model in size. It is shown below that the resulting changes are not necessarily useful when EM is applied over all of the language modeling text.

On average, how many morpheme paths do the different language models hypothesize for each word? The stem-finder and affix-finder transducers can both hypothesize more than one path through each word, depending on how the letters in the word compose with the patterns in the transducers. For the vocabulary comprising all of the training text, the stem-finder produces 1.28 and the affix-finder produces 2.19 hypotheses per word on average. Although the MADA method can predict different morpheme sequences for the same word if it appears multiple times, in this study, the top-ranked morpheme sequence is the same for most instances of a given word. Only about 5% of the words have multiple MADA decompositions, resulting in an average of 1.017 decompositions per word. The Morfessor method, by definition, proposes only one decomposition per word. The evaluations below

Model	Size of Text	Unigram	Bigram	Trigram
Words	12235	98.40	68.26	21.60
Stem	20015	<b>99.08</b>	84.06	44.73
Stem+EM	19274	99.00	69.30	11.59
Affix	19757	99.04	77.72	42.98
Morfessor	19356	98.69	<b>90.59</b>	63.35
MADA	21596	93.21	84.36	<b>64.07</b>

Table 4.6: Percent coverage of the test texts by each model, relative to each n-gram order. While unigram coverage is mostly equivalent across models, the morpheme models generally have better coverage of higher-order n-grams.

as well as the ASR results in the next chapter show that the multiple decompositions, while they could provide increased recovery of out-of-vocabulary terms, in fact do not result in increased coverage of the test text.

### 4.7.1 Coverage

Coverage statistics measure how many words or n-grams in an unseen text have probability mass in the language model. We compare the test and LM vocabularies:

$$Coverage(text|LM) = 100 - \left( \frac{\# \text{ tokens in text not in LM}}{\# \text{ tokens in text}} * 100 \right) \quad (4.6)$$

For larger n-gram orders, it is possible that some of the unseen tokens appeared in the training text but were pruned in estimating the language model. The main goal of morpheme language modeling is to reduce the number of unseen words in a test set, therefore, we expect to see higher coverage for the morpheme LMs than the word LMs. Table 4.6 shows how well each of the language models covers the test text.

We see that the unigram coverage is nearly identical for all models except the MADA model. For bigrams and trigrams, all of the morpheme models except stem+EM cover

Model	Percent word OOVs covered by morphemes
Stem	60.14
Stem+EM	60.14
Affix	54.05
Morfessor	68.24
MADA	74.32

Table 4.7: Percentage of word-based OOVs obtainable by combining the terms within each morpheme model. More than half of the out-of-vocabulary words can be recovered by all of the morpheme models.

more of the test set than the word model, confirming the hypothesis of reduced out-of-vocabulary (OOV) rates when modeling with morphemes. For bigrams, the Morfessor model performs best, followed by the MADA and stem models. The MADA model has the best trigram coverage, followed by Morfessor. Note that there is very little difference in the amount of trigram coverage provided by the Morfessor and MADA models, despite their very different methods for deriving morphemes from words. On this measure, the resource-light Morfessor method works about as well as the resource-heavy MADA method. This result will be repeated throughout the experiments in this work. The stem+EM model does not produce bigrams and trigrams that are more attuned to the test set than the stem model before EM. This may be because the iteration over the training set serves to emphasize those n-grams that occur frequently within it, but the higher-order n-grams are being pruned too aggressively after the LM weight application, causing n-grams in the test text to be under-represented in the model. Because the test text is first transformed into morphemes, the coverage is calculated over a different size corpus for each model. Therefore, calculating the statistical significance of the differences between models is not meaningful.

Model	Bigram	Trigram
Word	<b>95.51</b>	41.35
Stem	90.89	51.88
Stem+EM	81.42	36.12
Affix	85.90	38.47
Morfessor	85.51	<b>61.42</b>
MADA	62.40	51.91

Table 4.8: Percentage of OOV n-grams for which a backoff n-gram (one order lower) and backoff weight exist in the model.

We are interested in morpheme models because we assume that we can construct OOV words by combining morphemes. We measure the extent to which this assumption is true by counting how many of the OOV words are obtainable through combinations of the morphemes within each model. Table 4.7 shows, for each language model, what percentage of the 148 out-of-vocabulary words in the word-based test set can be recovered by combining morphemes within each model. The MADA model produces the most useful morphemes for re-composition of OOV words, followed by the Morfessor model. The stem and stem+EM models perform with equal usefulness, as they contain the same unigrams. The affix model has the least helpful morphemes. At best we will recover 75% of the OOV terms by employing morphemes. The OOV words not recoverable by the morpheme models may be named entities like people or organizations which are less likely to be subject to the inflectional varieties of regular nouns and verbs, or are foreign words, which do not follow the stem pattern rules described above.

We can also analyze the coverage results more closely by examining, for each of the higher-order OOVs, whether a backoff weight exists in the model for the next-lower-order n-gram. That is, if n-gram  $x y z$  does not exist in the language model, is there a backoff

weight associated with the n-gram  $x y$ ? When backoff is necessary, we want to be able to backoff as little as possible, as the longer n-grams are more informative, and have more reliable estimates. For instance, if we are looking for the probability of a trigram, we would prefer to backoff only to the bigram (if the trigram is unseen), than to back off to the unigram. The results given in Table 4.8 show that both the word and stem-based morpheme models provide backoff weights for nearly all bigrams. The Morfessor model provides the most backoff weights for the trigrams, followed by the stem and MADA models. The affix and stem+EM models show about the same results. The MADA model will have the least smooth estimates for higher-order n-grams. These differences may appear in the models' scores on perplexity evaluations and when applied to the speech recognition task. We would expect models with the smoothest distributions to make the most reliable predictions.

Considering all of the coverage measurements - size of LM, n-gram coverage, OOV recovery, and backoff coverage - no clear best model emerges from among the morpheme models.

### 4.7.2 Average Negative Log Probability

Language models are often evaluated by the perplexity of a test text given the model. Perplexity measures the average branching factor of the model, and is calculated as:

$$PP(x_1 \dots x_n) = 2^{\frac{1}{n} \sum_{x_i=4}^n \log P(x_i | x_{i-3})} \quad (4.7)$$

where  $n$  is the number of items in the test set. A lower perplexity indicates a stronger model. However, calculating this measure for each of our models would lend bias towards the morpheme models, as  $n$ , the number of morpheme tokens, will invariably be larger for those models, therefore making the perplexity lower. In order to evaluate the morpheme

models, we use an adapted measure of perplexity introduced in [Kirchhoff et al. \[2006\]](#), called average negative log probability (ANLP):

$$PP(x_1 \dots x_n) = \frac{1}{n} \sum_{x_i=4}^n \log P(x_i | x_{i-1}^{i-3}) \quad (4.8)$$

where  $n$  is the number of *words* in the test set, always as determined by the word model. This allows for a less biased comparison among the models. As with perplexity, we look for the lowest ANLP given to a test text by different models to indicate the model that encodes the text most effectively. The log probabilities of Equation 4.8 are calculated using backoff; where a trigram is unknown, the bigram probability and backoff weights are used, and an unknown bigram will back off to the unigram probability and backoff weight. In each language model, a small amount of probability mass is attributed to an unknown symbol based on the number of unigrams in the model; that is, the discounting method applied to the model is used to apply probability mass to the unseen unigram. The unknown symbol is not included in any higher-order n-grams. When an out-of-vocabulary word or morpheme is encountered, it is replaced by this symbol and given this probability (as a unigram). In this way, no zero-probability terms are included in the calculation of ANLP. Subsequent terms are also measured as unigrams, bigrams, and then trigrams as informative context becomes available.

When working with multiple morpheme hypotheses, as with the stem, stem+EM, affix, and MADA models, each sentence in the test text can be encoded as an FSM, where each word is represented by all of its hypothesized morpheme sequences. Weights are applied to the paths by composing it with an FSM encoding the language model. There are two possible ways of calculating ANLP at this point. We can calculate the probability of the sentence by finding the shortest (least-cost) path of the associated FSM. This best path may

or may not be the one chosen as the best path in a task setting, where other scores like acoustic or translation probabilities may influence the best path calculation. Alternatively, we can sum the probabilities of all paths to find the total probability that the model gives to all morpheme paths. While summing over all paths is not practical in a single-pass decoder, and therefore will not be used in our ASR experiments, the calculation is a good way to calculate the model's efficiency at encoding the test text. Results of both methods are shown in Table 4.9. In the first three columns, marked Best Path, the minimum cost path is chosen. In the columns marked All Paths, the weights of all of the paths of the sentence-wide FSM are summed to produce the ANLP for that sentence. The word and Morfessor models have a single path for each sentence; rather than apply the FSM method, we simply use the `ngram -ppl` command in SRILM to calculate log probabilities for each sentence. In all cases, the log probabilities are summed across utterances and divided by the number of *words* in the test text. The affix model was too large to encode as an FSM with 4-grams and 5-grams, so the ANLP is unavailable for those models.

Beginning with the first column of Table 4.9, we see that the diminished context of the morpheme models cause them to assign a higher average negative log probability to the test text than the word model, indicating that the morpheme unigram models are weaker than the word unigram model. The predictive context of the morpheme-based models is diminished because each morpheme contributes only a portion of the context that a whole word can contribute. The morphemes alone are less predictive; for instance, one might imagine that the definite article morpheme is less predictive of what will follow it than a whole word form. Therefore the word model best encodes the test text when only unigrams are considered.

Model	Unigram Best Path	Bigram Best Path	Trigram Best Path	Trigram All Paths	4-gram All Paths	5-gram All Paths
Words	<b>4.18</b>	<b>3.35</b>	<b>3.26</b>	3.26	3.32	3.31
Stem	5.39	3.98	4.08	3.88	3.79	3.59
Stem+EM	5.87	5.27	5.38	5.28	5.30	5.29
Affix	5.14	3.92	3.54	<b>3.09</b>	–	–
Morfessor	5.63	3.94	3.31	3.31	<b>3.29</b>	<b>3.25</b>
MADA	5.99	4.17	3.76	3.61	3.47	3.56

Table 4.9: Average Negative Log Probability of the test set as calculated by each language model.

In the second column, we see that adding bigrams to the models greatly improves the predictiveness of all of the models except for the Stem+EM model. Although a word is often broken into a sequence of three morphemes in the morpheme models, it appears that sequences of two morphemes recover enough of the predictive context to make the ANLP scores of the morpheme bigram models equal to or less than the ANLP of the word unigram model.

As for trigrams, the Morfessor morpheme model assigns an ANLP to the test text very close to that of the word trigram model. Even though the expected predictive context of the Morfessor trigram model is less than that of the trigram word model, the test text is encoded with the same entropy by both models. The Morfessor model is able to overcome the lack of context by having a lower out-of-vocabulary rate and possibly by having more reliable statistics over its trigrams, which are bound to repeat more often in the training text than word-based trigrams.

The addition of trigrams to the Stem model exhibits an odd behavior: the ANLP increases in comparison to the Stem bigram model. This indicates that the Stem-based trigram statistics are in some way unreliable, or that the best-path algorithm is not providing

a good representation of the model. In the third column we also notice that the ANLP of the Stem model is greater than that of the Morfessor model. This may be due to a better encoding given by the Morfessor model, but it may also be due to the use of the best path to calculate ANLP. Whereas all of the probability mass of the Morfessor model is in the single path, the probability mass of the stem-based morpheme model is spread across multiple paths, and only one such path is used to calculate ANLP. In order to recover the lost probability mass, we can use the sum of all of the paths rather than the best path to calculate ANLP. None of the information encoded by the morpheme model is lost in the calculation of all-paths ANLP.

The use of all-paths lowers the ANLP compared to best paths for all multiple-path trigram models (column 4). The Affix morpheme model assigns the lowest ANLP to the test text. The Affix model produces an ANLP slightly lower even than that of the Word trigram model. Again, using morphemes produces a lower out-of-vocabulary rate which makes up for the diminished context of the morphemes. As for the Stem morpheme model, using the all-paths calculation recovers the lost probability mass, so that the model now encodes the test text with trigrams more efficiently than using the best-path method, but the encoding is not as efficient as that of the Word, Affix, or Morfessor models. The MADA model also benefits from the all-paths calculation even though there are fewer paths through each sentence; the ANLP it assigns to the test text is between that of the Morfessor and Stem morpheme models. While we do not use the all-paths method for decoding in the following chapters, the ANLP results indicate that a more sophisticated system in which such a decoding process was reasonable may be able to make better use of these models than a Viterbi decoder.

In most cases, increasing the n-gram order past trigrams produces lower ANLP, as expected. However, the statistics for higher-order word n-grams and MADA morpheme n-grams are sometimes unreliable enough to increase rather than decrease ANLP relative to the lower n-gram order. Increasing the n-gram order to 6- or 7-grams may be useful in the Stem and Morfessor model.

The Stem+EM model, built using stem-derived morphemes with first-round LM probabilities applied to the FSMs, gives a higher ANLP to the test text than all other models. While its unigram results are in line with those of the other morpheme models, it is the only model for which adding bigrams only results in a slight decrease in ANLP. As in the Stem model, adding trigrams worsens ANLP, and using the all-paths method helps only a small amount. Let us consider this in light of the Stem+EM's results in other evaluations in this chapter. In Table 4.5 (page 56), we see that there are far fewer trigrams in the Stem+EM model than in the Stem model (this is due to pruning during LM estimation, described below). In Table 4.6 (page 57), we see that the coverage provided by the bigrams and trigrams in the Stem+EM model is the worst of all of the morpheme models, and because of its small number of trigrams, the test set coverage is very poor for that order n-gram. The backoff coverage (Table 4.8, page 59) of the Stem+EM model is lower than that of the other morpheme models as well. All of these measurements point to the outcome seen in Table 4.9, that the Stem+EM model does not encode the test text as well as the other morpheme models.

Why are there so few trigrams in the Stem+EM model? To create the Stem+EM model, we apply the language model probabilities from the Stem model to the morpheme sentence FSMs. When we subsequently count the trigrams, the counts are weighted by those probabilities. In some cases, the weight applied to the trigram, summed over all instances, will

remain less than one, or less than the threshold required by the LM estimation algorithm for trigrams to be included in the subsequent LM. Therefore, trigrams that were included in the original stem model are not included in the Stem+EM model, because they appeared few times in the corpus and were given low probability by the Stem language model. This is not surprising given that the trigrams in the Stem language model were not highly beneficial (compare columns 2 and 3 of the Stem model in Table 4.9).

The effect of EM seems to have had an odd effect on the bigrams. Although the bigrams encoded in the original Stem model were useful, their coverage and encoding of the test text is not as high as expected in the Stem+EM model. The weighted counting of bigrams seems to have emphasized those that are less helpful to the model, and excluded many that were useful. It is possible that with the large vocabulary in this data set, more iterations of EM are necessary to achieve the desired results. With a large vocabulary, more uncertainty is present, so it may take longer to reach convergence. A set of experiments below shows that EM works well on a smaller data set. Given these results, however, future research might focus on whether setting the count thresholds differently for language model estimation would improve the results.

The EM algorithm for language model refinement is far more practical over a small data set. We can therefore see what happens to the average negative log probability of the test text as we continue to refine the model estimates using the iterated counting method discussed in Section 4.4 if we restrict the data to only the transcripts (455 tokens), eliminating the newswire text. Table 4.10 shows the ANLP results for the trigram stem-derived morpheme model over several iterations, with the previous iteration's LM estimates used to

Iteration	Stem Model	
	Unigrams	ANLP
0 (unweighted)	37561	5.53
1	36970	4.67
2	37507	4.58
3	37597	4.55
4	37507	4.55

Table 4.10: The average negative log probability of the test set for trigram models built on transcript data, and refined using the method described in Section 4.4.

weight the new counts each time. In this setup, the refinements to the language model appear immediately. We see that for the stem model, the scores converge after five iterations, for a total 17.72% relative decrease in ANLP.

In the results shown in Table 4.10, the EM process is proven to be useful for the stem-based trigram model trained only on transcript data. The number of unigrams in each model remains relatively stable, in contrast to the models built over all of the LM data, in which the number of unigrams increases significantly when EM is applied. In the larger LM instance, the number of unigrams grows because in the first application of LM probabilities, weight is added to spurious morpheme sequences that were pruned in the initial model. The added weight pushes them to a level at which they are not pruned from the subsequent language model. The probability mass attributed to these spurious hypotheses is taken from the more viable n-grams, thus making the model less predictive, and the ANLP higher. Even pruning away the less likely n-grams, the large LM to which EM has been applied is not as predictive as the original large stem-based model (see Table 4.9). It is possible that, for the larger stem model, further EM iterations will push the probability mass away from spurious and towards correct morphemes, so that correct morpheme sequences would

receive higher probabilities and encode the test text more efficiently. However, performing these LM compositions with the sentence FSMs is very time consuming over all of the language modeling data, taking up to three weeks to complete on four processors. Iterative application of LM weights is feasible and useful for the smaller language model, likely because the incorrect morpheme hypotheses are repeated fewer times across the data. These continue to be pruned or given low probability in all iterations of the EM process, as shown in Table 4.10, and the probability mass shifts more quickly to the most likely morphemes. The EM iteration over the smaller language model is also useful in the ASR task, as shown in the next chapter.

These probability scores are a good indication that morpheme-based language models may be as effective as word models in the NLP tasks to which they are applied, if not more effective. These scores do not, however, give a good indication of which among the morpheme models will be most effective. The specific tasks in which the models are used - speech recognition, language generation, machine translation - will require different properties, and therefore may prefer different models. In the next chapter, we apply these models in the decoding step of an Arabic speech recognizer.

## CHAPTER 5: TESTING THE MSA LANGUAGE MODELS FOR AUTOMATIC SPEECH RECOGNITION

The results of text-based evaluations in Chapter 4 suggest that the stem-derived morpheme language model will work as well or better than word-based or other morpheme language models when applied to a specific natural language processing (NLP) task. The task described in this study is automatic speech recognition (ASR). The language models are used in the decoding step, when we attempt to recognize the words contained in test utterances. This chapter describes how we build acoustic and pronunciation models from Arabic speech data, then test the various language models during decoding.

The automatic speech recognition problem is formulated as follows:

$$P(W|A) = \frac{P(A|W)P(W)}{P(A)} \quad (5.1)$$

where  $W$  is the spoken word sequence and  $A$  is the acoustic signal (Holmes and Holmes [2001]). Using this Bayesian equation, we try to find the most likely word sequence  $W$  given the acoustic signal.  $P(A|W)$  is the acoustic model, which we will build as a generative model with triphone HMMs. The acoustic model describes the relationship between the features in the acoustic signal and the labels (phonemes) that we use to describe the phonetic components of the language.  $P(W)$  is the language model, which will be built with both words and morphemes as the terms making up the n-grams. We use a pronunciation model to bridge the gap between the phonemes hypothesized by the acoustic model and the words and morphemes represented in the language model. We can ignore the  $P(A)$

term because, for any given utterance, it is constant and does not change the probability of a word sequence being the correct hypothesis.

Section 5.1 describes the makeup of the ASR system. This includes a description of the training and test corpora, the procedure to train the HMMs that comprise the acoustic model, and the method of creating an appropriate pronunciation model for each language model. Section 5.1 also describes the decoding procedure. The results of an initial round of decoding are described in Section 5.2. Because these experiments were not successful, in Section 5.3 we describe language models that comprise a mixture of words and morphemes, which allow us to better make up for the context lost when working with morphemes, while still taking advantage of the possibility of recovering OOV words. The results of decoding with the mixed LMs are described in Section 5.4. Analyses of how the models handle OOV words and how morphemes are employed in decoding are given in Section 5.5. To increase the word accuracy rate given the particular challenges faced with the morpheme+word models, we try expanding the word-based lattices with morpheme hypotheses, discussed in Section 5.6. Further discussion of the speech recognition results, including a comparison of the results of the different morpheme+word models and a comparison to the results in previous literature, is given in Section 5.7.

## 5.1 Acoustic Model Training Regime

The data used for building MSA acoustic models are from the TDT4 corpus, and consist of the Nile TV and Voice of America broadcasts and transcriptions. There are 74.18 hours of data. This corpus is broken up into a 70.87 hour (455K word) training set, 1.65-hour development set, and 1.66-hour test set. The development and test sets represent the chronologically last eight data files, and contain about 10K words in their transcripts. The

development set was used to tune parameters such as pruning level (beam width), word insertion penalty, and language model weight.

### 5.1.1 Word-Based Pronunciation Modeling

A pronunciation model is derived with the help of the Buckwalter Arabic Morphological Analyzer (BAMA, [Buckwalter \[2004b\]](#)), which hypothesizes short vowels and geminates for many of the words, given that the diacritics for these phonemes do not appear in the transcripts. An example of output from the BAMA tool is shown in [Figure 5.1](#). All possible pronunciations hypothesized by the analyzer are included in the pronunciation model. For the 44K unique words in the training transcript, there are a total of 224K pronunciations, indicating an average of about 5 pronunciations per word. In the experiments below, we supplement the BAMA pronunciations with grapheme-only pronunciations for words or morphemes that are not found in the BAMA dictionary. The possibility of using the stem-based finite state machines to generate vowel predictions in place of the use of BAMA is discussed in [Chapter 10](#).

There are 71 monophones represented in the pronunciation model. This is a large number compared to approximately 45 monophones used to describe English sounds. In Arabic, the monophones include doubled consonants (geminates), in addition to consonants, long vowels, short vowels, and silence. The small number of vowels is countered by a large number of consonants, some of them unused in English, such as pharyngeal fricatives and ‘emphasized’ or pharyngealized consonants.

For studies in which a voweled (as opposed to a graphemic) pronunciation model is used, the application of the Buckwalter Arabic Morphological Analyzer, its successor

```

<token_Arabic> يحتوي
  <variant>yHtwy
    <solution>
      <lemmaID>{iHotawaY_1</lemmaID>
      <voc>yaHotawiy</voc>
      <pos>ya/IV3MS+Hotawiy/IV+(null)/IVSUFF_MOOD:I</pos>
      <gloss>he/it + contain/include + [ind.]</gloss>
    </solution>
    <solution>
      <lemmaID>{iHotawaY_1</lemmaID>
      <voc>yaHotawiya</voc>
      <pos>ya/IV3MS+Hotawiy/IV+a/IVSUFF_MOOD:S</pos>
      <gloss>he/it + contain/include + [sub.]</gloss>
    </solution>
    <x_solution>
      <voc>yHtwy</voc>
      <pos>yHtwy/NOUN_PROP</pos>
      <gloss>NOT_IN_LEXICON</gloss>
    </x_solution>
  </variant>
  <variant>yHtwY
    <solution>
      <lemmaID>{iHotawaY_1</lemmaID>
      <voc>yuHotawaY</voc>
      <pos>yu/IV3MS+HotawaY/IV_PASS+(null)/IVSUFF_MOOD:I</pos>
      <gloss>he/it + be contained/be included + [ind.]</gloss>
    </solution>
    <solution>
      <lemmaID>{iHotawaY_1</lemmaID>
      <voc>yuHotawaY</voc>
      <pos>yu/IV3MS+HotawaY/IV_PASS+(null)/IVSUFF_MOOD:S</pos>
      <gloss>he/it + be contained/be included + [sub.]</gloss>
    </solution>
    <x_solution>
      <voc>yHtwY</voc>
      <pos>yHtwY/NOUN_PROP</pos>
      <gloss>NOT_IN_LEXICON</gloss>
    </x_solution>
  </variant>
</token_Arabic>

```

Figure 5.1: Pronunciation possibilities are generated with the Buckwalter Arabic Morphological Analyzer, which includes short vowels and geminates in its analyses. The pronunciations are between “voc” tags, and include phonemes like ‘a’, ‘u’, and ‘i’, which represent short vowels.

MADA, or a similar morphological analyzer with pronunciation output is common. For instance, [El-Desoky et al. \[2009\]](#) use the pronunciations and morphemes provided by MADA for a group of experiments. [Xiang et al. \[2006\]](#) use BAMA together with other “decompounding” constraints to build a morpheme language model with voweled pronunciations. [Messaoudi et al. \[2004\]](#) finds a best path through BAMA pronunciation variants by applying acoustic model probabilities. [Nguyen et al. \[2009\]](#) use a morphological analyzer similar to BAMA to generate pronunciation hypotheses, then limit their morpheme language models to only those terms that have pronunciations. [Biadisy et al. \[2009\]](#) use a series of linguistically-informed rules to craft a pronunciation model for both words and morphemes. Other studies, especially those involving non-MSA data, will use grapheme-only models ([Diehl et al. \[2009\]](#), [Choueiter et al. \[2006\]](#), [Billa et al. \[2002\]](#)), or where available, phonetic transcriptions provided with the corpus ([Kirchhoff et al. \[2006\]](#)). More information on these and other methods of vowel insertion can be found in Chapter 9.

### 5.1.2 HMM training

We use HTK ([Young et al. \[2002\]](#)), the HMM Toolkit, to build the acoustic models. A hidden Markov model (HMM) is a stochastic, sequential model that emits probabilities of feature vectors according to a distribution learned from labeled data. Each state of an HMM represents a possible label in the outcome. In a speech recognition system, the labels are the set of monophones, or combinations of monophones possible in the language (e.g., triphones or quinphones, etc). The emissions of the states are the probabilities that the given acoustic feature vectors would have been generated by the model. We train the parameters of the model by associating sequential frames of acoustic feature vectors with correct labels, using an Expectation-Maximization algorithm to learn the distribution of

acoustic features corresponding to each label. In this way, each state of the HMM learns a particular probability density function of the values of the acoustic feature vectors, given its label. For a sequence of acoustic feature vectors with unknown labels (the test utterance), we determine the set of states that was most likely to have emitted that sequence. By applying the Viterbi algorithm, we can predict the sequence of labels corresponding to that most likely state sequence (Rabiner [1989]).

The acoustic signal is converted into 13 Mel Frequency Cepstral Coefficients (MFCCs) with delta and double-delta derivatives, resulting in 39 features per 25ms window with a time step of 10ms. These features are applied to the basic 3-state HMM training procedure described in the HTK manual and summarized here. First, we find a global average for each of the 39 features by extracting their means across a subset of the acoustic data.<sup>5</sup> These averages are designated as the initial values for each of the 71 3-state HMMs, one model per monophone. Baum-Welch re-estimation is used to train the models with monophone transcripts. After the first round of re-estimation, a one-state short pause HMM is introduced, which allows for silence to be hypothesized between each word. The models are re-estimated with this short pause included. The transcripts are realigned, allowing alternative pronunciations to be chosen for each word, after short pause re-estimation. We then create a cross-word triphone transcription, which includes 26K unique triphones, and re-estimate the models. Triphone HMMs with low probability, as well as the set of unseen possible triphones, are then tied to more stable models using the decision tree method, where a series of phonological questions are considered to determine which triphones are most alike (Young et al. [1994]). The trees used for our Arabic data were supplied by the

<sup>5</sup>Through monophone training, we use a subset of the corpus made up of the smallest utterances that add up to a total of 34 hours. By using only the shorter utterances in the first part of training, we were able to build more stable models with fewer run-time errors.

Air Force Research Lab (Slyh et al. [2007]). The tied triphone HMMs are re-estimated. The number of Gaussian mixtures in each HMM is then increased in increments of 2 and re-estimated after each increase until 16 mixtures per HMM have been estimated.

We generate one set of acoustic models with which to test all of the language models. In many previous studies, researchers have trained separate acoustic models with transcripts and pronunciation dictionaries comprised of morphemes. In initial testing, this method was not found to be fruitful, so instead, we simply change the pronunciation model and language model at decode time. This reduces training time and allows for a more direct comparison among language models, as the acoustic model remains the same throughout all experiments.

### **5.1.3 Decoding**

Decoding is performed with HDecode, a component of HTK. The Viterbi algorithm is used to find the best path through the labels generated by the HMMs at each frame. This is done by considering not only the state emission probabilities, but also the associations in the pronunciation model between phonemes and terms, and the scores in the language model concerning sequences of those terms. The best path through the lattice, accounting for both acoustic and language model scores, is retained as the recognized utterance. In the case of morpheme models, adjacent morphemes in the best path are concatenated using the dash annotations on the morphemes as guides to concatenation points, in order to compare the recognized utterance to the word references.

We employ a two-step process for decoding: in the first step, we create decode lattices using a trigram LM trained only on the transcription data. We find the best path of these lattices to calculate an initial decoding accuracy. We use this small LM in order to stay within

the memory bounds of the computing resources. Then, we expand the decode lattices with a trigram or higher-order LM built on all of the training data, including the newswire text. This second step is performed with the *lattice-tool* component of SRILM (Stolcke [2002]). The two kinds of language models will be referred to as LM-transcript and LM-alltext.

#### 5.1.4 Pronunciation Models for Decoding

When we decode with the word-only language model, we can use the same pronunciation model that was employed to train the HMMs. For the morpheme LMs, a new pronunciation model is built using the BAMA tool. The process is shown in Figure 5.2. We first derive a vocabulary of morphemes from our trained language model, then present the vocabulary to the morphological analyzer. Many of the morphemes are unknown to the analyzer, nevertheless a set of hypothesized pronunciations is generated for them using the consonants in the morpheme, that is, a grapheme pronunciation is generated for unknown morphemes. Although the resulting pronunciations are less reliable for morphemes than whole words, especially due to missing phonological information from neighboring phonemes, using the same procedure for deriving word and morpheme pronunciations allows us to share the acoustic model between experiments. A new pronunciation model is built to accommodate the morphemes in each language model.

A related note is that an additional spelling change was made to best incorporate the language models into the decoding process. Rather than using brackets to mark morpheme boundaries as described in Section 4.1, instead we simply mark morpheme boundaries with a dash “-”. While the brackets are more informative as to which terms exactly match the stem patterns, the dashes are easier to work with in the decoding process. In any decoded morpheme sequence, if two adjacent terms have matching dashes, they are concatenated

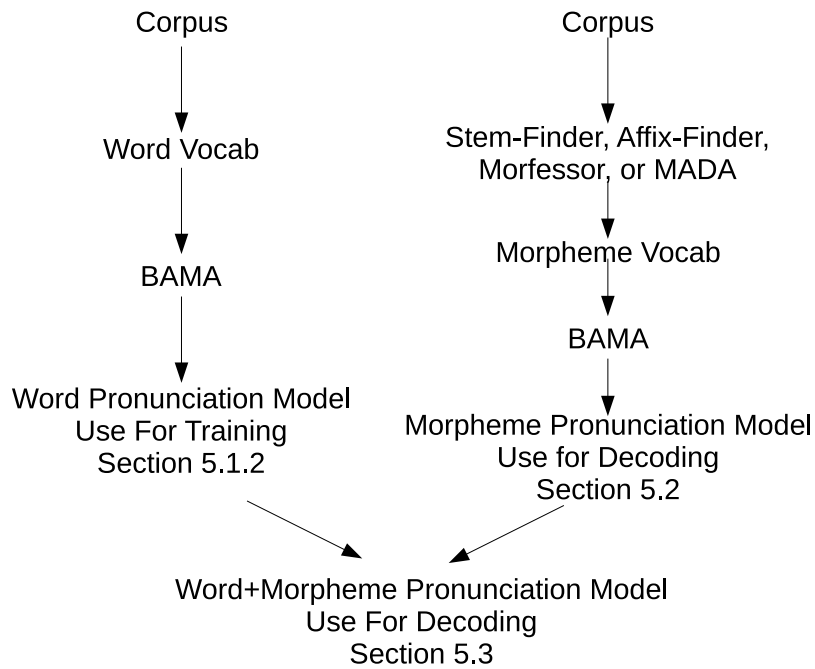


Figure 5.2: A new pronunciation model is derived for each language model, always using BAMA to produce voweled pronunciation variants.

Model	3-grams	4-grams	5-grams	6-grams
Word	74.95			
Stem	58.02	58.02	55.03	54.71
Affix	58.07	58.85		
Morfessor	55.91	44.99	53.11	53.01
MADA	42.48	41.28		

Table 5.1: Word Accuracy on the development set using word-only and morpheme-only models. The morpheme-only models are inferior to the word-only model: all morpheme results are significantly different from the word results at  $p \leq .0001$ .

to form a word. If a stray morpheme is found, for instance, a morpheme with a dash at the start but no dash on the preceding term, then the dash is removed and we assume the morpheme is a stand-alone word. The problem of hypothesizing stray morphemes, or illegal morpheme sequences, is discussed further in Section 5.4 below.

## 5.2 Decoding with the Morpheme-Only LMs

Table 5.1 gives the results for decoding the development utterances with the morpheme-only language models described in the previous chapter. The lattices were created using the LM-transcript and expanded with LM-alltext. We see that the morpheme models cause a significant reduction in word accuracy as compared to using the word model for decoding.

The accuracy reduction is due in part to the lack of context in the morpheme models: because a word can be broken into as many as three morphemes (and possibly more for the Morfessor model), a sequence of three morphemes is often equal in context to a single word. The predictive ability of a morpheme such as ‘Al’ at the beginning of a word, which may be followed by many different stem morphemes, may not be as strong as the predictive ability of a stem, after which it is likely that a suffix or the prefix of a following word

will appear. The same may be said of function versus content words in the word model; however, because nearly every word is split into morphemes in the morpheme model, the frequency of affixes will out-pace that of function words, making the predictions of the morpheme model weaker. It is interesting to see that the morpheme models do not improve as more context is added with higher-order n-grams. This means that the lattices may be of poor enough quality given the initial trigram decoding strategy and acoustic model, or the estimates of the higher-order n-grams are not high quality.

The reduction in word accuracy is also likely in part due to the mismatch in pronunciations in the switch from words to morphemes. The acoustic models are trained on word transcripts, then tested with morpheme pronunciation models. The duration statistics and end-of-word models may be different in words and morphemes, causing a decrease in accuracy. Phonological processes sometimes occur over multiple adjacent sounds; these effects are not always modeled properly when the pronunciation model is based on morphemes. Furthermore, the short length of the morphemes means there is less room for error, and more opportunity for ambiguous pronunciations, than when we decode with words.

### **5.3 Developing Morpheme+Word Language Models**

To address the problem of context and reduce the phonological mismatch, we create models that are mixtures of word and morpheme n-grams. Each n-gram may have both whole words and morpheme sequences within it. The four morpheme derivation methods require different procedures for incorporating words with the morphemes.

#### **5.3.1 Morpheme+Word N-grams for Stem- and Affix-Based Models**

In the two models where morphemes are derived via finite state machines (FSMs), we add a whole-word arc to each lattice representing a decomposed word. See Figure [5.3](#)

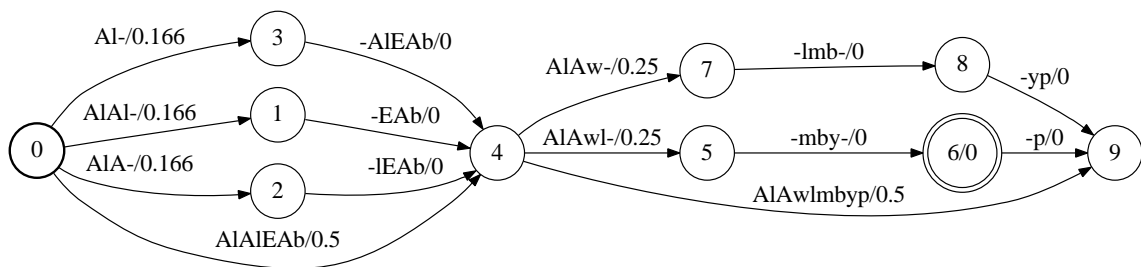


Figure 5.3: Each word in a two-word phrase has been decomposed into stem-based morphemes, and the whole-word arc added to each FSM. The whole-word arc receives half of the probability mass, and the remaining paths share the remaining probability mass evenly.

for an example of two such decomposed words concatenated. To summarize the earlier discussion of using FSMs to decompose a word, we build either a stem-finder FSM or an affix-finder FSM, using stem patterns or affix definitions respectively, and compose that finder FSM with an acceptor representing the word. The outcome of the composition is an FSM representing all of the possible morphological decompositions of the word. Adding a new step, we produce the union of the morpheme FSM and a whole-word, single-arc FSM. Furthermore, we assign weights to the new FSM. The word arc receives half of the possible weight of all paths through that word, and the morpheme paths share the remaining weight equally. This is done to prevent the morpheme sequences, of which there may be multiple types for each word, from dominating the language model probabilities in comparison to the less-frequent words. The weights are also shown in Figure 5.3. Other weighting schemes, including the use of EM to determine the weights, are discussed in Section 5.4.1 below. When these morpheme+word FSMs are concatenated into a sentence-level FSM, the n-gram counting algorithm creates estimates for sequences that include both words and morphemes.

As before, we build language models by first applying the efficient n-gram counting method [Allauzen et al. \[2005\]](#) to count morpheme+word n-grams over the sentence-wide FSMs, then estimating the language model from those counts. In this case, we use Witten-Bell discounting, as it allows for partial counts, whereas Good-Turing and Kneser-Ney discounting do not.

### **5.3.2 Morpheme+Word N-grams for MADA-Based Models**

We do not use finite state machines to derive the morpheme possibilities with the MADA tool, so we must use a different procedure for incorporating words into the n-grams along with morphemes. In this case, we replicate a method described in [El-Desoky et al. \[2009\]](#), where the most frequent words are left whole. For each utterance in the training text, we use MADA to find the most likely decomposition of each word. Then, we create a new sentence wherein the infrequent words are replaced with their decomposition, and the frequent words remain intact. We define frequent words as those in the top quarter of the frequency list, slightly more than [El-Desoky et al. \[2009\]](#) use in their smaller vocabulary system. N-gram counts are calculated over these mixed sequences, and language models estimated in course. No weighting scheme is necessary, as there is only a single path through each utterance.

### **5.3.3 Morpheme+Word N-grams for Morfessor-Based Models**

The procedure for incorporating words into the Morfessor-based language model follows the lead of the MADA procedure, such that we leave the most frequent words whole. To do this, we rely on a feature of the Morfessor algorithm: in calculating the cost of the model and frequency of proposed morphs, we take into account the word frequencies. The frequent words are rarely decomposed because their whole-word status is less costly than

the morph sequences that comprise them (Creutz and Lagus [2005]). Therefore, we let the algorithm decide which of the words are decomposed. The result is that 18K words remain whole, 38% percent of the vocabulary. We can compare this to the 13% that remain whole when word frequencies are not taken into account. The procedure is otherwise the same as before, with each sentence transformed into its morphs and words. N-grams are counted across each sentence, and the language model is estimated from those counts.

Initial experiments showed that these mixed models are not sufficient on their own. The decoding is greatly enhanced by interpolating the estimates from the word-only model with the mixed models described above. The SRILM toolkit provides the facility to mix estimates from several language models. Therefore, in the experiments described below, each language model is an unweighted mixture of two separate models:

1. A model built of mixed word and morpheme n-grams
2. Word-only n-grams

Experiments in which the models were weighted differently in the interpolation did not produce better results. Also, interpolating the morpheme-only language model as well as the word-only model did not change the results, we therefore leave out this option for simplicity.

To reiterate, these mixed models, which will be called morpheme+word models, allow us to encode greater context than the morpheme-only models allow, and also provide greater cohesion between the acoustic, pronunciation, and language models.

Model	Development Set		Evaluation Set	
	LM-transcript	LM-alltext	LM-transcript	LM-alltext
Word	71.67	74.95	72.75	76.55
Stem+Word	68.14	72.21	69.11	72.34
Affix+Word	68.20	72.34	68.80	73.34
MADA+Word	70.09	73.54	71.59	74.87
Morfessor+Word	70.27	73.41	71.64	74.66

Table 5.2: Word accuracy results on the development and evaluation sets with the morpheme+word models. The word model performs the best in all conditions - development and test, transcript and alltext language models.

## 5.4 Decoding with Mixed Word and Morpheme LMs

In all of the experiments presented in Table 5.2, the word model performs significantly ( $p \leq .05$ ) better than any of the morpheme+word models, using either the LM-transcript or LM-alltext models, for both the development and evaluation sets. Why is the word-based accuracy not surpassed or even met by the mixed models? Upon studying the decode lattices, we found that many illegal morpheme sequences are predicted by the mixed models. Even though they are not directly represented in the language model, sequences such as two consecutive prefixes, or a stem followed by a prefix, can occur in the output due to backoff. Choueiter et al. [2006] found that constraining the morpheme predictions to only legal sequences using FSMs is useful to counteract this problem. We could not change HDecode to add this constraint, but we did try to compose the decode lattices with a constraining FSM; this method was not successful in our system. Instead, we collect all illegal bigrams that occur in the initial lattices, and add them into the language model with zero probability. The language model is re-normalized with the additional bigrams in order to return it to a

Model	Development Set		Evaluation Set	
	LM-transcript	LM-alltext	LM-transcript	LM-alltext
Word	71.67	74.95	72.75	76.55
Stem+Word	70.64	74.16	69.74*	73.09*
Affix+Word	70.56	74.67	71.21*	75.11*
MADA+Word	71.40	74.56	72.53	75.88
Morfessor+Word	71.95	74.83	72.83	75.88

Table 5.3: Results on the development and evaluation sets of decoding with mixed morpheme+word language models in which illegal morpheme sequences are included with zero probability. \* signifies significantly different from the word model,  $p \leq .01$ . Many of the morpheme models now perform with accuracy equal to the word model.

probability distribution. The additional zero-probability bigrams serve the purpose of discouraging backoff to illegal sequences, shifting probability mass to sequences we know to be more likely. The test utterances are decoded again with the enhanced transcript-only language model. The results of decoding with the illegal sequence enhanced morpheme+word models are shown in Table 5.3.

The addition of the zero-probability illegal sequences changes the results. For both the development and evaluation sets, we see that the morpheme+word models now produce results that are non-significantly different from the word model, except in the case of the stem+word and affix+word model on the evaluation set. For the development set LM-alltext experiments, the morpheme+word models are non-significantly different from each other. The MADA+word and Morfessor+word models produce results that are statistically equal in all experiments. The stem+word and affix+word models perform the same on the development set, but the affix+word model performs significantly better than the stem+word model on the evaluation set.

An additional iteration of finding illegal sequences in the lattices and adding them into the model did not produce different decoding results. We also note here that it was necessary to use a beam width of 200 to achieve scores that were statistically the same for word and morpheme+word models. (The default beam width for decode lattices in our work is 150.) The interaction between the morpheme+word models and the acoustic model is such that more predictions are required from the initial decode in order for the language model to choose a good path. Increasing the beam width further did not improve results, and causes the processing time and space to increase. It might seem beneficial to expand the decode lattices with 4-grams from the LM-alltext, however, this was not always feasible on our system. In the instances where the 4-gram expansion did finish, the results were not different from the LM-alltext 3-gram experiments. This indicates that the higher-order n-grams are too sparsely represented in the corpus to provide more useful probability estimates than the 3-grams.

The lack of improvement with the addition of morphemes to the language model indicates that the deficiencies in the pronunciation modeling procedure were harmful to the experimental results. Recall that we model the morpheme lexica as well as the word-only lexicon with the Buckwalter Arabic Morphological Analyzer (BAMA). The morphemes that do not exist as independent items in the BAMA lexicon are not modeled with short vowels, making them different and less consistent with the acoustic model than the whole words modeled with short vowels. Furthermore, we do not model the way in which pronunciations may change when the morphemes are concatenated; in other words, because the morphemes pronunciations are not modeled in the context of the whole word, we miss the phonological changes that can occur at morpheme boundaries. We believed that using this consistent method of pronunciation modeling would allow us to repeat the use of the same

acoustic model with all language models, rather than re-training the acoustic model with morpheme-based transcripts for each language modeling experiment. Initial experiments indicated this to be the case, however, the final results indicate that the mismatch between acoustic, pronunciation, and language models is problematic for recognition.

Why did the two FSM-based models, the stem- and affix-derived morpheme models, perform worse than the Morfessor and MADA models? The most obvious difference between the two pairs is that the FSM models hypothesize multiple morpheme sequences per word, and the Morfessor and MADA models only predict one morpheme sequence per word. (The MADA model initially considers multiple hypotheses, but in each instance, chooses one sequence to represent the word.) As we showed in Section 5.3.1 above, the probability mass for each word is spread among the hypotheses in the FSM-based models. The word arc receives half of the probability mass for that word, and the remaining morpheme paths receive an equal share of the remaining probability mass. The motivation for this weighting scheme was to prevent the morphemes from acquiring too much probability mass in comparison to the words, given that for any word there may be multiple morpheme paths, but only a single whole-word arc. However, the particular weight application that we chose resulted in an undesirable distribution of probability mass. Morpheme sequences in words for which there are many paths have a smaller probability than morpheme sequences in words with few paths (compare the weights on the morpheme paths of the two words in Figure 5.3 on page 80). Words that have fewer decompositions can then dominate the model. This kind of prior on the morpheme sequences is not well motivated linguistically, which may make the FSM-based models less predictive. Moreover, given that we do not test for the viability of the morphemes against a lexicon, we must assume that probability mass is being given to some wrongly hypothesized morpheme sequences. In Chapter 4 we

Model	LM-transcript
Stem+Word, Even Weight	68.14
Stem+Word, No Weight	69.14
Stem+Word, EM Weight	70.62

Table 5.4: Word accuracy scores on the development set for LMs that use different weight applications on sentence-wide lattices before n-gram counting. The *a priori* application of even weights across morpheme arcs was not useful; using no weighting scheme or a statistically-informed weighting scheme produces increased word accuracy.

described how our expectation is that the language model probabilities will winnow out the less viable morphemes in favor of those that repeat with more reliability. Apparently, the weighting scheme or multitude of n-grams is preventing this from happening effectively. Additional experiments shown in the following section support this hypothesis.

#### 5.4.1 Weight Application in Morpheme+Word Models

To build the stem+word language model in Table 5.2, we partition the weight for each word evenly between the word arc and the derived morpheme paths. We apportion one half of the probability of the word to the word arc, and split the remaining weight among the morpheme sequence paths. We do this for each decomposed word in the sentence, then concatenate the FSMs before counting n-grams across the sentence-wide FSM. At least two other ways exist to apportion weight across the word and morpheme paths for each word. First, we can assign no weight to any path, thereby considering the paths to be equiprobable. The count of the n-gram sequences will be only dependent on the number of times the sequence occurs in the sentence, and not on how many paths exist through each word. Second, having estimated an LM on these equiprobable paths, we can apply the LM weights to the sentence-wide lattices, and count n-grams a second time to create

an EM-enhanced language model, as described in Section 4.4. We estimated two such models on the transcript data, and decoded the development data with these new language models. The results are shown in Table 5.4, along with the score of the evenly weighted model used above. We see that the language model estimated with EM-applied counts gets the best score on the development set, using the transcript-only LM. Unfortunately, there was not time to pursue using this model instead of the even weight model with the LM-alltext, due to the time and resources required to collect these n-grams over the entire corpus. However, these experiments show promise that the EM application of weights to the morpheme hypotheses produces better language models, and has the sought-after effect of removing or more severely discounting the spuriously hypothesized morpheme sequences.

## 5.5 Analysis of OOV prediction

The results in Table 5.3 do not support the claim that morpheme models help reduce the OOV error rate. Before attempting once more to use the morpheme+word models successfully, we will look more closely at how each of the models makes use of morphemes and how each model handles out-of-vocabulary words. The analysis will focus on the mixed morpheme+word models, where illegal morpheme sequences have been removed, as these produced the best results. These analyses correspond to the evaluation set.

Table 5.5 shows that the affix model uses the most morphemes overall: 10.57% of its predictions are morphemes rather than words. This might have been expected given the results shown in Chapter 4, where we saw that the affix model was most permissive in its morpheme derivation. That is, the affix model has the most morphemes to choose from, and also uses them to the greatest extent in decoding with a mixed morpheme+word model.

Model	% Predictions = Morpheme	% Morpheme = Label	% Morpheme Sequence = Word in Reference	%Morph Seq. = OOV Word
Stem	9.62	2.61	33.15	2.01
Affix	10.57	6.79	41.02	3.66
MADA	3.28	3.92	19.90	7.46
Morfessor	3.57	10.24	33.78	7.11

Table 5.5: How each word+morpheme model uses morphemes in predictions for the 9480-word evaluation set. The affix model predicts morphemes most often, and the morphemes that the affix model predicts are most likely to coincide with a word in the utterance.

The stem-based model uses morphemes for 9.81% of its predictions, slightly fewer than the affix model, while the MADA and Morfessor models use morphemes for only 3-4% of their predictions. The number of morphemes predicted by any of the models is far fewer than the number of words predicted. There are at least two ways to try to explain the preference for words. We might expect the acoustic model to give higher scores to the words than the morphemes, especially considering that the acoustic model was trained on word transcripts; however, this prediction is not borne out by an analysis of the acoustic model scores on the decode lattices. Another explanation may be that, because word sequences exist in both the morpheme+word language model and the word-only language model used for interpolation, the language model statistics regarding the words are stronger and overpower the morpheme statistics to some extent.

The second column of Table 5.5 describes the percentage of morphemes that exactly match their paired label. The matching is considered without the morphemes' dashes. It is the case that some morphemes can also represent whole words; for instance, a term with a preceding dash prefers to be attached to a prefix, but may be a grammatical word without that prefix. If the term exists in the pronunciation model with and without the dash, then it

will have the same pronunciation for both, and it will be up to the LM to choose which one is preferred given the context. The affix model has the most predicted morphemes matching the aligned label, although the number of exact matches is low, at 5.6%. The stem-based model performs the worst in this regard, with 2.7% of morphemes matching the aligned label. Again, the higher percentage for the affix model is likely due to the fact that the derivation method allows for more morpheme types, therefore there is a higher chance that such a morpheme might be valid as a stand-alone word, and a higher chance that it will match the label. The low number of matches overall is not surprising since we design the model so that *sequences* of morphemes are predicted to match OOV (and other) words. Column 3 of Table 5.5 shows the percentage of morpheme sequences predicted by each model that represent any valid word in the evaluation set. The affix model's predictions most closely match the evaluation set vocabulary, followed by the stem model, both at about 41%. Morpheme sequences that do not match a word in the evaluation set vocabulary may be valid words, but they will not have produced a 'hit' for word accuracy. The fourth column of Table 5.5 shows the percentage of predicted morpheme sequences that represent OOV words. These figures, all below 10%, are lower than we may have predicted, as we specifically make use of morphemes to try to recover the OOV words. The MADA and Morfessor models have the most morpheme sequences matching OOV words; this may be one reason that they provide slightly better word accuracy than the stem- and affix-based models in some experiments.

We can look at the same information from the point of view of the OOV tokens. Of the 9480 words in the evaluation set references, 538 (5.68%) are OOV words. How many of the OOV tokens are aligned with a morpheme? The first column of Table 5.6 shows that the affix and stem models replace about a quarter of the OOVs with morphemes. The next

Model	% OOVs Replaced With Morpheme	% Replacements Match OOV	% OOVs deleted
Stem	24.16	1.54	9.29
Affix	24.72	6.02	13.94
MADA	11.52	3.23	13.57
Morfessor	13.01	8.57	13.38

Table 5.6: Description of how the 538 OOV tokens in the evaluation set are handled by each of the word+morpheme models. The stem and affix models replace OOVs with morphemes the most, but only in  $\frac{1}{4}$  of OOV instances. Rarely do morphemes recover OOVs successfully.

column describes the percentage of those morphemes that exactly matched the OOV with which they are aligned. Very few OOVs are exactly matched by a morpheme, but, this value does not take into account whether that OOV is exactly matched when adjacent morphemes are concatenated. The MADA and Morfessor models replace only half as many OOVs with morphemes, but according to the analysis above, these replacements are more likely to be accurate than those of the stem- and affix-based models. The stem and affix models use deletion for OOVs about as often as the MADA and Morfessor models.

This analysis of morpheme prediction and OOV handling by each of the language models reveals that the morphemes are indeed detrimental to the decoding process, as shown in Tables 5.2 and 5.3 above. We know that there is a mismatch between the acoustic model, built on word-only transcripts, and the pronunciation of the morphemes. Still, we hope to address the OOV problem by predicting morphemes rather than words in places where OOV words occur in the signal. As the analysis shows, this does not often happen. OOV words are resolved with morpheme predictions only 27% of the time at most, and in these cases, there is a low probability that the morpheme or sequence of morphemes will match

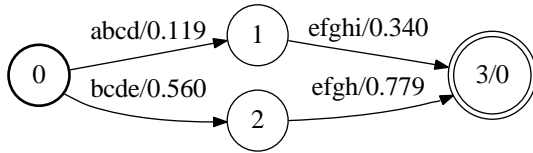
the OOV word. We see that this is the case regardless of which morpheme derivation technique we choose. Therefore, we must look elsewhere in the system for ways to better utilize the morpheme sequences.

## 5.6 Enhancing Word-Based Decode Lattices with Morphemes

In an effort to make better use of the morpheme models and avoid the pronunciation model mismatch, we perform a set of experiments that use the morpheme models to rescore the word-only decode lattices. The steps are as follows:

1. Decode the test utterance with the word-only language model and pronunciation model. Save the decoded lattice for each utterance.
2. Encode each lattice as a finite state acceptor with the acoustic model scores included as weights on the acceptor.
3. For each word arc on the acceptor, find the stem-based or affix-based decomposition. Concatenate the outcome FSMs in the same arrangement as the original word lattice. This process is described further below.
4. Encode a morpheme+word language model as a finite state machine, with the language model probabilities and backoffs included as FSM weights.
5. Compose the morpheme-enhanced decode lattice FSM from Step 3 with the language model FSM.
6. Find the best path of the composition, and concatenate the morphemes into words. Compare the outcome to the reference labels.

Word decode lattice:



After decomposing each word arc and re-applying AM scores:

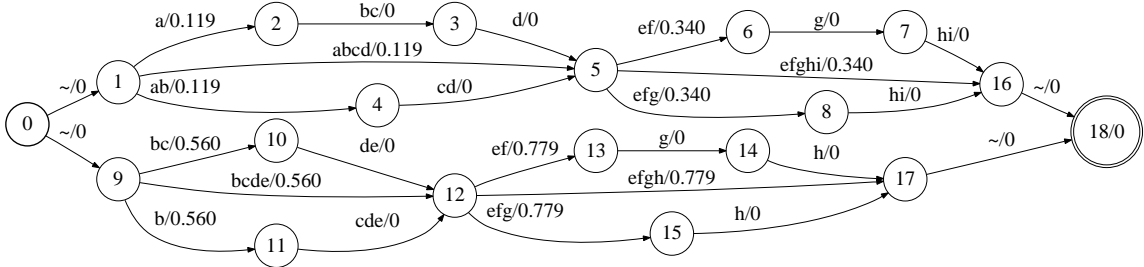


Figure 5.4: Each word on the decode lattice is decomposed into its component morphemes, encoded as a small FSM. The morpheme FSMs are concatenated in the same way as their word arcs were originally arranged. The acoustic model score of the word is applied to the initial arcs of the morpheme FSM.

The word to morpheme decompositions performed in Step 3 are necessary for successful composition of the word-based lattices with the morpheme+word language models. For each arc on the decode lattice, we create an acceptor representing just that word, and compose it with a stem-finder or affix-finder transducer (as described in Section 4.1). The resulting morpheme sequences, as well as the original whole-word arc, are encoded as a single finite state machine which is inserted into the sentence-length FSM in place of the original word. The acoustic score associated with the word in the original lattice is applied to the arcs emanating from the first state of the morpheme FSM. A simple example of this is shown in Figure 5.4. We do not use the MADA or Morfessor models to enhance the lattices because they do not fit well into the FSM architecture designed for the stem- and

Model	Development Set		Evaluation Set
	LM-transcript	LM-alltext	LM-transcript
Word	67.87	74.95*	72.75
Stem+Word	71.59*	71.83	72.42
Affix+Word	71.62*	71.96	72.42

Table 5.7: Word accuracy result for rescoring the word-based decode lattices with morpheme+word language models. Rescoring gives significantly better ( $p \leq .01$ ) results than words for the transcripts-only model, but the words-only lattice expansion to alltext is significantly better than rescoring with the alltext morpheme+word LM.

affix-based models, and also used to create the enhanced decode lattices. While it is possible to create FSM models reflecting the output of the MADA and Morfessor algorithms, doing so made the process too slow for inclusion in this work.

In these experiments, we have removed the main reason for applying a morpheme-based instead of a word-based language model: by starting with the word-based decode lattices, we cannot improve on the OOV rate. Only the words originally proposed in the word-based decode lattice will be hypothesized in the rescored lattice; however, a different best path may be chosen based on the ways in which the words decompose, and the likelihood of those morphemes in context.

The word accuracies for these experiments are significant in the case of the transcript-only LMs on the development set, as shown in Table 5.7. Composing the morpheme-enhanced decode lattice with a morpheme+word language model built on transcripts produces a significantly better score ( $p \leq .01$ ) than the best path given by the word-only LM-transcript (column 1). When we expand the decode lattices with the word-only LM-alltext (row 1, column 2), the score improves even further, but not when we rescore with

a morpheme+word LM-alltext (rows 2 & 3, column 2). The use of morphemes in the alltext case, even when interpolated with the word model, is not as powerful as the context given by the trigram words alone. The lack of effect with the LM-alltext models may be due to the pruning performed on the morpheme+word LMs.<sup>6</sup> We might also come to a similar conclusion as Choueiter et al. [2006], that larger vocabulary language models cause morpheme estimates to be less useful. Because the LM-alltext experiments were not successful for the development set, and because the rescoring is slow to complete, we ran only the LM-transcript experiment on the evaluation set (column 3). In the case of the evaluation set, rescoring with morpheme+word models does not produce better results than the word-only model - no useful information is added to the decode lattices.

If the results on the development set are indicative that this decoding method will be useful for some data sets, it might mean that the probability estimates made for the more oft-repeated morpheme sequences are more reliable than the probabilities estimated for the word sequences, at least over a small corpus. The small language model limitation is not necessarily a downfall: for many dialects of Arabic, only a small amount of language modeling text is available, so this method may prove to be more useful in those domains than in the MSA domain.

## 5.7 Discussion

Why do the morpheme-based models not significantly increase the word accuracy of the system? The main cause is likely the interaction between the morphemes and the pronunciation model: the pronunciations derived for the morphemes do not take into account

<sup>6</sup>We use the SRILM flag `-prune` with threshold  $1.0 \times 10^{-6}$  to prune these models as minimally as possible to allow for FSM encoding.

the phonological processes that occur at morpheme boundaries. Consider the word الطالب, AITAlb in Buckwalter encoding, meaning *the student*. It is pronounced [ət<sup>ʕ</sup>:ɑ:lɪb]. The coronal ط, (T) pronounced /t<sup>ʕ</sup>/, is considered a “sun letter” which causes the /l/ in the definite article to assimilate to its place of articulation. The first consonant in the word becomes the geminate /t<sup>ʕ</sup>:/ . Furthermore, ط is an emphatic consonant, and it causes the subsequent sound represented by the alif (A), normally pronounced as /æ:/, to become a back vowel, /ɑ:/. In the word-only pronunciation model, consonant assimilation is explicitly modeled so that الطالب (AITAlb) has the correct pronunciation [ət<sup>ʕ</sup>:ɑ:lɪb], as above. The /l/ of the definite article does not appear in this pronunciation. In the morpheme-based model, the word will be in two parts: ال (Al), with pronunciation /əl/ or /æɪl/, and طالب (TAlb), with pronunciation /t<sup>ʕ</sup>ɑ:lɪb/. No assimilation is accounted for. Sun letter assimilation does not occur across word boundaries, so any cross-word triphones with the l-#-t<sup>ʕ</sup> combination from the word model will not help in this situation. The same is true for the backing effect that the ط has on the alif ( ا ). If the word happened to be split into morphemes between the ط (T) and the ا (A), there would be no indication to the acoustic model that the alif should be the backed kind, as this phenomenon does not occur across word boundaries. It is possible that applying one of the more sophisticated pronunciation modeling methods discussed in Section 5.1.1 or Chapter 9 would improve the results of these experiments. By accounting for these and other phonological processes that occur across morpheme boundaries but not word boundaries, all of the morpheme models used in this study might achieve higher word accuracy rates.

How do the results for the current system compare to the results in previous literature for morpheme modeling in Arabic ASR? Table 5.8 shows a comparison of the levels of WER improvement across 13 studies using morpheme-enhanced Arabic language models.

Publication	Language	Morpheme/Model Type	Best Improvement (% Relative)
Afify et al. [2006]	Iraqi	Affix	14.60
El-Desoky et al. [2009]	MSA	MADA	12.64
Nguyen et al. [2009]	MSA	Combination	11.70
Xiang et al. [2006]	MSA	Affix	9.84
Choueiter et al. [2006]	MSA	Affix, med vocab	7.57
Diehl et al. [2009]	MSA	MADA	7.46
Sarikaya et al. [2007]	Iraqi	Affix	7.45
Nguyen et al. [2009]	MSA	Sakhr	6.36
Nguyen et al. [2009]	MSA	Affix	3.72
Vergyri et al. [2004]	Egyptian	FLM	3.34
Kirchhoff et al. [2006]	Egyptian	FLM	3.11
El-Desoky et al. [2009]	MSA	MADA, voweled	3.00
Kirchhoff et al. [2002]	Egyptian	Combination	2.54
Kirchhoff et al. [2006]	Egyptian	Stream, Class	1.94
Stolcke [2006]	Levantine	Affix	1.41
Choueiter et al. [2006]	MSA	Affix, large vocab	0.74
Emami et al. [2008]	MSA	Rich NN	0.00
Creutz et al. [2007]	Egyptian	Morfessor	-4.06

Table 5.8: Comparison across previous research of word accuracy improvements using morphemes, relative to comparable word models within the same study. Different combinations of system complexity, language form, and morpheme derivation method produce a large variety of results.

As discussed in the previous chapter, we see that there is precedent for using morpheme-based models; many studies gain significant improvements in word accuracy by enhancing the models in this way. However, there are two studies that show no improvement, and for some studies where morphemes were helpful, the effect is only slight. Given these poorer published results, and given the bias for publishing only positive, significant results, we might assume that other studies concurring with the results shown in this chapter were performed but remain unpublished. Another indication of this assumption is in [Vergyri et al. \[2008\]](#): in building an Arabic ASR system, morphemes were discarded as the models grew large, because “as more data became available the improvement from these models diminished.” Some publications indicate that the amount of benefit is highly dependent on the corpus. In the [Nguyen et al. \[2009\]](#) study, five evaluation sets are used; for the best model, the improvement in word accuracy across those five sets ranges from 2.2% to 0.9% absolute.

There does not seem to be a clear correlation between the complexity of the system or the size of the model and the utility of morphemes, judging from the list in [Table 5.8](#). Highly sophisticated and relatively simple systems, as well as large and small corpora, can be found in studies at both the top and bottom of the list. Furthermore, there does not seem to be a correlation between complexity of morpheme derivation and success of the resulting model. For instance, the [Afify et al. \[2006\]](#) study uses the relatively simple affix definition method, also constraining predicted stems using the BAMA tool. It is highly successful (14.60% relative improvement over word models) within its domain of Iraqi Arabic. Likewise the knowledge-rich MADA method appears high on the list with 12.64% improvement shown on MSA data. On the other hand, the Factored Language Models

of Kirchhoff et al. [2006], which take into account a multitude of linguistic data and sophisticated backoff algorithms, provide a much smaller (3.11%) improvement for Egyptian Arabic, as does the simple affix method when used with a large vocabulary on MSA data (0.74% improvement).

We might conclude that each study in Table 5.8 proves the utility of morpheme models and morpheme derivation methods *for that system*. It is difficult if not impossible to predict whether a complex, simple, or lack of morpheme derivation method will be most useful for a particular implementation of Arabic automatic speech recognition.

The results shown in Sections 5.3 and 5.4 add to the argument that there is little correlation between the complexity of morpheme derivation method and benefit to word accuracy. The Morfessor algorithm uses the least amount of linguistic knowledge of the four derivation techniques explored in this study, requiring only a vocabulary list and optional frequency counts, whereas the MADA model requires the most amount of linguistic knowledge, including a corpus of parsed data over which to collect part-of-speech sequence statistics. Despite this disparity, the morphemes and morpheme sequences resulting from these two models were about equally useful in the ASR task, producing similar results in all of the experiments shown. Unless the grammatical information about words and morphemes provided by the MADA model is to be used in downstream processing, it seems that the additional work required to implement the resource-heavy MADA model does not provide any benefit beyond the resource-light derivation method provided by Morfessor.

In addition to this new perspective, we have learned that an important parameter in designing a morpheme derivation method is to properly account for multiple morpheme hypotheses. The two methods in which a single hypothesis is made for each word (MADA and Morfessor) generally perform better than the two methods in which multiple hypotheses are

made (stem and affix). Properly ranking the multiple hypotheses, in particular by using the EM method for refining the resulting language models, is crucial. Further improvements to the stem-based and affix-based morpheme models might be achieved if EM were carried through more iterations or introduced over the larger data set (although see Section 4.7.2 for caveats). It is unclear whether this method would overcome the mis-match between the language and acoustic models, but we expect that the scores from the affix and stem-based morpheme models would at least be equivalent to, if not better than, those of the MADA and Morfessor models. In this instance, we will have combined the linguistic insights used to create the affix and stem models with statistical information gathered directly from the data to produce an effective language model.

Given that the morpheme models produce approximately the same word accuracy results in this ASR system, are there properties of the models that make them more or less useful or attractive for other tasks? Some models provide morphological information beyond the morphemes themselves, which could be used to make more knowledge-rich LMs or class-based models. These richer n-grams could be used in other tasks or downstream processing such as machine translation (MT), MT performed on ASR output, or natural language generation. As to the information each model could provide:

- The Morfessor algorithm is fast, resource-light, and language independent. However, it is capable only of providing “morph” hypotheses, which may or may not correspond to meaningful grammatical linguistic units. No further analysis of those units can be made without applying knowledge-based processing or statistical clustering to induce meaningful categories.

- The affix definition model is nearly as resource-light, requiring only a list of likely affixes and optionally some heuristics for narrowing the hypothesized word decompositions based on the length of the remaining stem or inclusion of the stem in a given lexicon. With this method, we are able to label each sub-word unit as either a stem or an affix. More specific information is not available.
- The stem-based method requires about the same amount of linguistic knowledge as the affix method, only requiring a list of stem patterns rather than a list of affixes. Again, we can label each term as an affix or stem. Additionally, because the stem patterns also define possible short vowel sequences, we have the opportunity to predict this missing information as well as the hypothesized morpheme sequences (see Chapter 10). Similarly, because each stem pattern is associated with specific grammatical paradigms and word classes, we could include on the output tape some grammatical information about each of the hypothesized stems (see Figure 4.1). This possibility is not explored within this thesis, but, it is possible that such word class information could be used to develop class-based n-gram language models. However, because each stem pattern is possibly associated with multiple grammatical paradigms and parts of speech, the ambiguity of the word classes may render them unsuitable for the task. The stem-based method will most likely over-generate analyses for each word due to its liberal definition of possible stems and multiple analyses for each stem pattern. Further work could be done to analyze the extent to which the grammatical information assigned to each stem by the stem patterns matches that given by the Buckwalter Arabic Morphological Analyzer. The two examples of adding short vowels or grammatical information to the stem pattern FSMs highlight the flexibility of the Finite State Machines. It is a straightforward task to add or remove constraints

whenever new linguistic knowledge or statistically-derived probabilistic information is acquired for a given a data set. The inclusion or exclusion of certain graphemes from the affix or stem patterns requires only a simple change to the lists defining those alphabets. Addition or subtraction of stem pattern possibilities is similarly simple. Prior probabilities on the frequency of stem patterns, acquired from a data set annotated with such information or linguistic knowledge of the language, can be included as weights on the FSMs. New FSMs can be quickly compiled; no extra training on annotated data is needed to implement the use of new constraints.

- The MADA method of morpheme derivation provides specific information about each word's decomposition, including semantic, syntactic, and phonological (pronunciation) data. This information may be useful in building class-based n-gram models, factored language models, or acoustic models. However, the hypotheses are likely to be less reliable when the data being analyzed is not Modern Standard Arabic. Changing the parameters of the MADA model to accommodate a change in the language form of the data is non-trivial, especially given that a significant amount of annotated data is needed to train the SVM classifier.

Faced with the task of building an Arabic ASR system, the results of this study and those shown in Table 5.8 suggest that one should examine the use of morphemes in the language models using a simple method, such as Morfessor, which is extremely resource-light. If it seems for the given system and data set that morphemes are advantageous, then the use of a more complex derivation method might be warranted. The choice of morpheme derivation method might then be driven by the dialect of Arabic in use, as well as the tools and resources available for training the models. For instance, for Factored Language Models to be used to their full potential, a wealth of grammatical annotations

should be available. If they are not, a less resource-heavy method, such as MADA, affix definitions, or stem definitions will likely provide the sought-after effect.

Despite the lack of positive results from incorporating morphemes into word models for MSA, we have still to discover which of the morpheme derivation methods will prove to be useful cross-dialectally, if any. The next several chapters focus on this modeling challenge.

## **CHAPTER 6: HOW REGIONAL DIALECTS DIFFER FROM MSA**

This chapter deals with linguistic variation between Modern Standard Arabic (MSA) and Levantine Conversational Arabic (LCA), and variation between different colloquial dialects, usually defined by their geographic regions. Levantine Conversational Arabic is a regional spoken dialect of the family of languages called Arabic, whereas Modern Standard Arabic is a written, modernized form of Classical Arabic, used by literate people in formal settings such as news broadcasts and university lectures. This delineation is a major simplification; stating for certain what is a colloquial form and what is standard is very difficult. A speaker may use linguistic features - sounds, words, or structures - from both informal and formal language within a single utterance. In addition to addressing differences at the phoneme, word, and sentence level, sociolinguistic variables are discussed in this chapter to help the reader understand the diglossic situation that exists in the Arabic-speaking world.

In designing natural language processing (NLP) tools meant to be useful in multiple dialects, such as the language modeling framework described in this dissertation, it is crucial to understand the differences between the chosen dialects. The design of NLP tools benefits from knowledge of specific linguistic differences between language forms, as well as knowledge of the contexts in which those differences arise. Some of the difficulties in transferring NLP technology from one form of Arabic to another may be resolved by understanding the linguistic and sociolinguistic issues described below.

## 6.1 Understanding Diglossia

Diglossia, as defined by Ferguson [1959], is

“a relatively stable language situation in which, in addition to the primary dialects of the language (which may include a standard or regional standards), there is a very divergent, highly codified (often grammatically more complex) superposed variety, the vehicle of a large and respected body of written literature, either of an earlier period or in another speech community, which is learned largely by formal education and is used for most written or formal spoken purposes, but is not used by any sector of the community for ordinary conversation.”

In other words, in a diglossic community, the choice of language form is inherently both linguistic and social; speakers choose to use more formal or more casual forms based on the content they wish to convey, as well as on the basis of their particular social context.

As stated above, grammar is a main difference between standard and colloquial language forms. In fact, linguistic features in the domains of phonology, morphology, lexicon, and syntax are all relevant to describing how LCA and MSA are different. Each will be discussed in turn below.

## 6.2 Phonology

MSA and LCA differ phonologically in their vowel inventories and use of some consonants. The set of phonemes for each language is described in Tables 6.1, 6.2, and 6.3 below.

Tables 6.1 and 6.2 show that LCA exhibits a greater variety of short and long vowels, adding /e/ and /o/ to MSA's /a/, /i/, and /u/. As a dialect that is mainly spoken but not written, LCA has no standard alphabet. Those who wish to write in the colloquial form

often use the MSA script. However, since symbols for /e/ and /o/ (long or short) do not exist in the MSA alphabet, this information is lost in an MSA orthographic representation of LCA. Linguistic resources (such as spoken language corpora and their transcriptions) will often include a Romanized transcription, which may or may not represent the /e/ and /o/ vowels as separate from /a/, /i/, and /u/, depending on the transcriber’s preferences or instructions. When working with a transcribed corpus of LCA, NLP tool designers must take care to note the alphabet used in the transcription.

The consonants represented in Table 6.3 are all phonemic in both MSA and LCA. However, some lexical items exhibit phonological variation across the two forms. The phoneme /θ/ is a good example of this variation. As described in Hussein [1991], the MSA words ثنائي [θa:ni] “two” and ثلاث [θa:liθ] “three” are pronounced [ta:ni] and [ta:liθ] in Levantine Conversational Arabic<sup>7</sup>. The MSA words مثلا [maθala:n] “for example” and ثابت [θa:bIt] “firm” are respectively pronounced [masalan] and [sa:bIt] in Levantine. However, the /θ/ is pronounced the same in both MSA and LCA in the word أمثال [ʔamθa:l] “similar”. The following phonemes have a regular, one-to-one relationship with their associated grapheme in MSA, but the same grapheme or segment of the Levantine cognate may have a different pronunciation: /ʔ/, /q/, /ð/, /ḏ/, and /d/.

Another example of articulatory diversity similar to that of /θ/ is the phoneme /q/. This phoneme has many articulatory representations in the colloquial dialects. It may be pronounced as /q/ in MSA (an unvoiced uvular stop), as an unvoiced palatal stop /k/, as the glottal stop /ʔ/, or as /g/, a voiced palatal stop. In more than one dialect, phonological innovations involve the /q/ variable. According to Benrabah [1994], in a rural Algerian Arabic dialect some speakers use /q/ in informal speech where one would expect to hear /k/ in such

<sup>7</sup>Throughout this chapter and the dissertation, Arabic words are spelled in the Arabic script, followed by the IPA representation of the word’s pronunciation, followed by an English gloss in quotes.

Vowel	Description
i:	closed, front, lip-spread, long
i	closed, front, lip-spread, short
u:	closed, back, lip-rounded, long
u	closed, back, lip-rounded, short
a:	open, long
a	open, short

Table 6.1: Vowel Inventory of Modern Standard Arabic (Holes [1995])

Vowel	Description
i:	close, front, lip-spread, long
i	close, front, lip-spread, short
u:	close, back, lip-rounded, long
u	close, back, lip-rounded, short
a:	open, long
a	open, short
e	close-mid, front, lip-spread, short
ee	close-mid, front, lip-spread, long
oo	close-mid, back, lip-rounded, long

Table 6.2: Vowel Inventory of Levantine Conversational Arabic (Hussein [1991])

Place of articulation	Manner of articulation				
	Plosive	Fricative	Affricate	Liquid	Nasal
Labial	b	w			m
Labiodental		f			
Dental plain	t d	s z		l	
Dental emphatic	ṭ ḍ	ṣ (ẓ)			
Interdental plain		θ ð			
Interdental emphatic		ḏ			
Alveolar		ʃ ʒ	j	r	n
Palatal		y			
Velar	k (g)	x ɣ			
Uvular	q				
Pharyngeal		ħ ʕ			
Glottal	ʔ	h fi			

Table 6.3: Consonant Inventory of Modern Standard Arabic and Levantine Conversational Arabic, as presented in [Holes \[1995\]](#)

interactions. Similarly, in Tunis, /g/ is historically pronounced in the Bedouin dialects in words where /q/ is found in urban dialects or MSA. Those Bedouin dialects now exhibit instances of /q/ where they were not found in the past ([Gibson \[2002\]](#)). [Gibson \[2002\]](#) proposes two hypotheses to account for these changes: either the rural dialects (both Algerian and Tunisian) are influenced by the Modern Standard Arabic that is widely disseminated in today's world, or they are influenced by the urban dialects with which they have more contact than ever before. Other phonological variables in the same dialects point to the latter hypothesis as more likely. The /æʕ/ vs /a/ variable in Algerian Arabic exhibits the urbanization trend: according to the study performed in [Benrabah \[1994\]](#), the preferred variant is /a/, which is associated with Oran, an urban center of Algeria, and with characteristics like monetary wealth, higher education, a likeable personality, and speech that is pleasant

to the ear<sup>8</sup>. Gibson [2002] identifies three Tunisian variables besides /q/ vs /g/; one is a morphological variable, the second is a lexical variable<sup>9</sup>, the third phonological. This last variable is the alternation in rural areas between /aj/ and /i/ and between /oʊ/ and /u/. The rural dialects use /aj/ and /oʊ/, also found in an older form of Arabic and MSA. The urban dialects, especially in Tunis, are more inclined toward /i/ and /u/ or other monophthong versions. All three variables indicate the trend of rural dialects changing to be more like urban dialects, not MSA. Therefore, the spoken dialects used in urban areas are the current prestige forms toward which rural dialects are likely to change. The innovations in rural dialects may change in either of two directions, towards the conservative MSA forms, or towards the forms used in urban spoken dialects. In some cases, MSA and urban dialects constitute the same direction of change. In cases where the urban dialectal form and the MSA form differ, researchers are increasingly finding that the innovations trend towards the urban spoken dialects.

### 6.3 Morphology

Arabic morphology has different properties as concern the inflectional, mainly concatenative processes, or the derivational, templatic processes. This dissertation assumes that there are more significant differences between MSA and colloquial dialects in relation to the inflectional morphology than the derivational. Table 6.4 displays an example of inflectional variation. The verb “to study” in both MSA and Levantine is derived from the

<sup>8</sup>in Benrabah [1994], these are characterized as “richness, education, likeability, and prettiness of speech”

<sup>9</sup>These two variables are: the treatment of the final vowel in defective verbs, that is, does the final vowel of the stem remain or is it replaced by the perfective morpheme, and the use of the word انت: [inti] “you” as the generic form of the pronoun, rather than as the feminine form.

person	number	gender	Levantine	MSA
1	1	m/f	darast	darastu
1	2	m/f	darasna	darasna:
1	3+	m/f	darasna	darasna:
2	1	m	darast	darasta
2	1	f	darasti	darasti
2	2	m/f	darastu	darastuma:
2	3+	m	darastu	darastu:
2	3+	f	darastu	darasna
3	1	m	daras	darasa
3	1	f	darasat or darsat	darasat
3	2	m/f	darasu	darasa:
3	3+	m	darasu	darasu:
3	3+	f	darasu	darasna

Table 6.4: Conjugation of the root /drs/ with Pattern 1 template, giving the perfect tense form for all person/number/gender combinations in both Levantine (Hussein [1991]) and Modern Standard Arabic (Haywood and Nahmad [1965]).

root letters د ر س (d, r, s). In both language varieties, the pattern that combines with the root to create the basic past tense is simple: an open vowel occurs between the first and second and between the second and third root slots. Subtle variation between the dialects exists in the inflectional endings used to define number, person, and gender on these perfect tense verbs. MSA has more unique forms, reflecting more specifications of number and gender. For each form, the suffixes do not always match between the two dialects, such as the first person singular and second person singular masculine endings.

MSA also differs from most colloquial dialects in its use of short vowel case endings on nouns. Three case endings are possible: /a/ signals accusative nouns, /i/ on genitive nouns, and /u/ on nouns in the nominative case. When a noun occurs at the end of a phrase (pause position), the case ending remains unspoken. In most colloquial dialects, these case distinctions are not made.

Parkinson [1994] exploits this difference in exploring Egyptian speakers' ability to speak correct MSA spontaneously. Using case endings as a single representative variable, the study shows that the subjects' proficiencies in MSA range from quite poor to surprisingly good. Those subjects with an elementary education use case endings as a kind of MSA 'marker': a single type of vowel ending is used sporadically throughout the dialogue, often incorrectly, but deliberately. Other subjects with more education or literary interests are more likely to use correct forms, and to use them consistently. Parkinson [1994] concludes that very few people can speak MSA spontaneously with great accuracy for sustained periods. More importantly, the variation in ability proves a point made by Ferguson [1991] and others: diglossia in the Arab world is not a simple situation of a "high" and a "low" language that are clearly delineated and used in complementary situations. Rather, there is a continuum of language, from the most informal to the most formal. A speaker may choose features along this continuum from the phonological, morphological, lexical, and syntactic domains, to the best of his or her ability and to best suit the purpose. Not every speaker has access to the same range of the continuum, nor does every speaker wish to use the entire range. Because the continuum varies along so many dimensions, and because the forms may change subtly or drastically within a conversation, sentence, or even within a single word, classifying a segment of speech as "standard" or "colloquial" is nearly always an approximation or generalization.

## 6.4 Lexicon

Insofar as MSA and LCA are distinct dialects, the differences in their lexica arise in large part from the differences in their contexts of use. Colloquial dialects like LCA are spoken while chatting with family and friends or pursuing day-to-day business. MSA in

its most conservative form is mainly associated with broadcast news and university lectures. The topics that are discussed in MSA include science, technology, literature, and medicine, but also daily topics like weather and shopping, sports and politics. Naturally, these domains overlap; the TV news must report on the weather, and like anywhere else, people chat casually about the weather as well (Al Wer [2002], *inter alia*). Table 6.5 shows a list of simple, everyday terms in MSA and LCA. While some terms are shared, lexical differences exist even for common words. As mentioned earlier, it is not always possible to clearly define an utterance as “standard” or “colloquial”. Lexical choice is one way that speakers can vary the formality of their speech. If a speaker using mainly colloquial forms chooses a word normally associated with MSA, does it mean that the term has been borrowed into the local dialect and may become a part of that dialect, or is it an instance of code-switching? Although answering questions like this is beyond the scope of this dissertation, it is important to realize that the relationship between the standard and regional spoken forms of Arabic is complex and fluid, both at the level of the lexicon and in relation to the other variations discussed in this chapter.

While the lexicon of MSA is more or less consistent across geographic boundaries, the lexica of the varying regional dialects vary greatly. Examples of this variation are the focus of the discussion between four teachers of Arabic in 1960, transcribed and analyzed in Blanc [1960]. In their conversation, the young men agreed vocabulary differences made communicating with Arabs outside their home region both difficult and intellectually stimulating. In particular, they discussed terms for food items at length. They chatted about a restaurant menu printed in two regional dialects as an example of the extent of the differences. The conversation was similar to one that might take place between British and

English	MSA	Levantine
balcony	شرفة ، بلكون [balku:n],[ʃarfa]	برندة [bərəndə]
yard	حديقة ، فناء ، ساحة [hazɪ:ra], [fnæʔ], [sa:ha]	ساحة [sa:ha]
inside	داخل [da:ħil]	جوا [ʒu:wa]
room	غرفة [ʕurfa]	غرفة [ʕurfa]
kitchen	مطبخ [maṭbəx]	مطبخ [maṭbəx]
pencil	قلم [qalam]	آلم [ʕalam]
chicken	دجاجة [daʒa:ʒa]	جاج [ʒa:ʒ]

Table 6.5: Vocabulary differences and similarities for some common terms in Modern Standard Arabic and Levantine Conversational Arabic, as defined in [Hussein \[1991\]](#), [McCarus \[1979\]](#), and [Alosh \[2000\]](#), spelled with MSA orthography and in IPA symbols.

American English speakers, comparing how their dialects differ. While regional accents are clearly in play, such that pronunciations of the same words differ, of greater interest to the non-linguists in this conversation are the different labels speakers give to the items in their world.

## 6.5 Syntax

Among the syntactic features that may differ between standard and colloquial forms of Arabic, word order and agreement are cited in recent literature. The following discussion on differences in syntactic forms between MSA and LCA is informed by [Holes \[1995\]](#), a detailed description of the structure of MSA and colloquial varieties of Arabic.

The meaning of the sentence usually defines the choice of word order in MSA and LCA. In general, speakers describe events using the verb + subject + complement order, whereas the subject + verb + complement order occurs more often when the predicate describes a state or circumstance. One may expect to hear the former construction more often in MSA, especially given its use in news broadcasts, where speakers are often describing events.

Speakers of colloquial forms are more likely to use the subject + verb + complement order. In either case, the speaker may choose to manipulate the order given the definiteness of the subject, the part of the sentence that is to receive the most emphasis, and the information that is intended to follow the current statement. It is more useful, therefore, to analyze word order given its context rather than to compare types across language forms. Having built a language model, it may become clear that one word order is more common than another in our data set. In an  $n$ -gram language model, the reason for the order does not matter, but the syntax will be accounted for given large enough  $n$ . A language model that takes into account context-free grammar rules may need to incorporate this knowledge.

In MSA, a verb need only agree with the subject's gender, not number, in verb + subject + complement sentences. In such a sentence, the verb may have masculine conjugation even if the subject is feminine. However, if the subject is topicalized and in front of the verb, then agreement must be true in number and gender. The urban dialects in the Levant tend to use a "stricter" form, always requiring the verb to agree, regardless of order. MSA distinguishes between human and non-human nouns: the non-human nouns are always paired with feminine singular adjectives, regardless of the noun's gender and number. In the Levantine dialect, agreement between noun and adjective is likely even in the case that the subject is non-human. These kinds of differences between dialects should be taken into account when extending a language model from one dialect to another.

Blanc [1964] points out some syntactic differences that existed between the Iraqi dialects spoken in Baghdad at that time. Among the three religious communities living in Baghdad, Muslim, Christian, and Jewish, each spoke a slightly different form of Arabic. The clearest syntactic difference was the occasional use of a copula in the Christian Arabic dialect in the region. Neither the Muslim Arabic nor Jewish Arabic dialects used a copula.

This is yet another instance of diglossia; in public venues, speakers mostly used the majority Muslim variant, leaving the minority Jewish and Christian dialects mostly reserved for use at home.

Differences between dialects of certain syntactic structures occasionally serve to mark a speaker's usage as formal or casual, however, these structures are less often cited in the sociolinguistic literature. [Al Batal \[2002\]](#), in a study of news anchors' dialect choices on Lebanese broadcast news programs, reaches some general conclusions about dialect-mixing in this setting. Whereas most news programs use a very formal register of MSA with minimal amounts of straying into colloquial forms, the on-location reporters in the study mix in a fair amount of LCA. Al Batal describes the dialect in this way: "This high concentration of *fusha* (formal) lexicon and structures demonstrates that this register is mainly comprised of a *fusha* core that is passed through Levantine Colloquial phonological and morphological filters." The author also cites the reporters' use of some colloquial negation terms as instances of Levantine syntactic structures interpolated in MSA speech. Alternatively, the lapses into the informal register may indicate a lack of linguistic sophistication in the reporters' grasp of MSA syntax. Overall, this case study is another example of the very close and ever-changing interaction between the forms of language: formal vs. casual, conservative vs. innovative, literate vs. everyman.

## **6.6 Relationship to Natural Language Processing**

Although it may seem that LCA and MSA have more linguistic features in common than differences, NLP tools should be designed with the differences in mind. For instance, pronunciation dictionaries, morphological disambiguators, lexica, and parsers should take into account the linguistic variations described above. Also, the divide between the contexts

of use of Modern Standard Arabic and colloquial Arabic dialects is a gradient one, and multiple levels of the language hierarchy will influence language models or NLP tools built on any single training text. The resulting errors in our models, and the expectations of individual models generalizing to other language forms, will be associated with the social as well as linguistic forces that drive the variation.

## CHAPTER 7: PREVIOUS APPROACHES TO MODELING REGIONAL ARABIC DIALECTS

As discussed in Chapter 6, regional dialects of Arabic differ from each other in ways that affect the way we design NLP tools for each dialect. In addition, because many regional dialects are spoken and rarely written, getting text and annotated data is very difficult. Taken together, these two properties make the task of addressing the OOV (out-of-vocabulary) problem more difficult. In the studies discussed in this chapter, the OOV problem in regional dialects of Arabic is addressed by including morphological information in the language model in place of, or in addition to, word information. Alternatively, some of the efforts focus on extending Modern Standard Arabic (MSA) data, for which there are many resources, to dialect applications.

As described in Chapter 3, the goal of Afify et al. [2006] is to decrease word error rate for Iraqi Arabic by morphologically decomposing the words, thereby decreasing the OOV rate. Segmentation is performed by defining Iraqi affixes and applying blind segmentation, then using length of stems and other heuristics to limit segmentation possibilities and inclusion in the segment dictionary. Segmentation decreases the OOV rate, as expected, and also decreases the word error rate in a speech recognition task. Interpolating word- and morpheme-based language models is also useful, as it allows for greater contextual information while maintaining a low OOV rate. It may not be the case that the stems and affixes of Iraqi Arabic are very similar to those of MSA; however, decomposition and modeling over morphemes is useful for both language forms. In the following chapter we will

compare the use of morphemes in MSA and Levantine Arabic, and continue to compare different methods for acquiring morphemes from words.

Using sub-word units in place of words is not found to be as useful in [Creutz et al. \[2007\]](#), in which the authors use *Morfessor* to compare morpheme models to word models across several languages, including Egyptian Arabic. For all languages except Arabic, the morpheme models improve word error rate due to their ability to model OOV words. The authors show that for the Egyptian Arabic corpus, the rate of vocabulary growth is not much different from vocabulary growth in English, especially as compared to the growth rate of Finnish and Turkish words. In other words, [Creutz et al. \[2007\]](#) find that the explosion of word forms due to concatenative morphology is not as extreme in Egyptian Arabic as it can be in MSA or other languages. This is a different result than was found in [Afify et al. \[2006\]](#), where morphological decomposition did help for a different dialectal form, Iraqi Arabic. It is therefore crucial to take into account as much knowledge as possible for the dialect with which one is working. The *Morfessor* algorithm will be used to predict morphemes for the Levantine Arabic data set, allowing us to test whether the use of morphemes generated in this way are more useful for Levantine than Egyptian Arabic.

[Kirchhoff et al. \[2006\]](#) address the out-of-vocabulary problem by testing three morphology-enhanced language models plus a baseline: a stream model, a particle model, a word model, and factored language models. Morphological information to populate these models is derived from the annotated lexica provided by the LDC for the Egyptian CallHome Arabic corpus ([Gadalla et al. \[1997\]](#)), and from a MSA root finder ([Darwish \[2002\]](#)). Including morphology-enhanced language models in the rescoring of decode lattices improves the word error rate a small but significant amount. The decode lattices are initially created using word-based bigram language models; therefore, as in the ASR experiments with MSA

discussed in Chapter 5, even if the morpheme models do not improve the OOV rate, the information arrived at by studying sequences of morphemes can be useful in choosing the correct word sequence.

Factored language models are used with Levantine Conversational Arabic data in [Stolcke \[2006\]](#). While attempts to introduce short vowels to the Levantine data were unsuccessful, introducing factored language models did decrease the word error rate slightly. The factored language models were built with minimal morphological knowledge of LCA. For instance, the root information used in the models was derived with a tool designed for MSA data. Including the morphological information improved results from 42.7% and 47.0% word error rate to 42.1% and 46.5% on two different evaluation sets. The statistical significance of these improvements is not stated. Perhaps using morphological information tailored for LCA would have improved results even further. We will use a combination of constraints chosen to work specifically with LCA and constraints designed to work globally.

[Sarikaya et al. \[2009\]](#), using Iraqi Arabic as the data set, takes a different approach to the OOV problem by framing the language modeling task in the way that acoustic modeling is normally framed. Here, the language model is represented as a series of numeric vectors, where the parameters are estimated as Gaussian distributions and low-frequency parameters are tied to higher-frequency parameters via state-tying. HMMs are used to model the word or morpheme sequences. A state may represent a word with morpheme observations, or a morpheme with varying morpheme observations. The features of each HMM are derived from a word or morpheme co-occurrence matrix that is reduced to a lower dimensionality using singular value decomposition. The labels emitted by the HMM are hypotheses for the next word. If the states represent morphemes, then context-dependent tri-morphemes

can be modeled. Decision trees are used to determine the tying parameters. In ASR experiments on Iraqi Arabic, where the experimental LM is used to rescore a word-based n-best list, the results do not improve greatly. However, when interpolated with a 3-gram word model, the tied-mixture language model does decrease the word error rate. Crucially, morphological information helped the speech recognition task for Iraqi Arabic, as it did in [Afify et al. \[2006\]](#).

[Rambow et al. \[2005\]](#) take a more general approach in addressing the issue of Arabic dialects. Their goal is to produce NLP tools that relate Levantine Conversational Arabic to MSA in a way that foreshadows building similar tools for other dialects. They assume that, for a given dialect, very little data is available, but some knowledge of the lexicon or grammar is available. For instance, a lexical mapping between dialects is described, a resource that could help resolve language-internal ambiguities. Since no parallel corpus is available for the two dialects in question (LCA and MSA), a seed dictionary and co-occurrence statistics are used to find candidate word pairs. They find that searching the larger corpus for examples or sections that best match the content or genre of the smaller corpus is a useful processing step. By answering the question, “how exactly are these two language forms related?”, they hope to make MSA tools more accurate on dialectal data, by enhancing them with this linguistic knowledge. In using the stem-based morphological decomposition method, we take a different approach, in allowing multiple dialects to share some basic structure (the stem patterns), rather than focusing on specific aspects of shared lexica.

The same kind of inter-dialectal linguistic knowledge discussed in [Rambow et al. \[2005\]](#) is also codified in the tool described in [Habash et al. \[2005\]](#). The goal of this study is to build a multi-dialectal morphological analyzer that includes lexemes, real morphemes,

roots, patterns, orthography, and phonology, to be used for both analysis and generation. A resource with this kind of information can be very valuable to NLP research when it is properly populated and maintained; however, the trade-off between great detail and ability to generalize comes to the forefront here. It is unclear whether such a tool would be useful to new dialects, or how one might know which parts of the resource were applicable to new data. The initial effort to build such a tool is also very great, especially where few textual resources are available to aid in the work. For these reasons, many researchers prefer data-driven solutions, estimated models built on annotated data, rather than looking to highly detailed, knowledge-infused tools such as that described in [Habash et al. \[2005\]](#).

A similar application of cross-language resource sharing is described in [Hana et al. \[2004\]](#), wherein a Czech morphological tagger is combined with unannotated Russian corpora to obtain a resource-light Russian morphological tagger. The relationship of Czech to Russian is similar to the relationship between some Arabic dialects; as described in [Hana et al. \[2004\]](#), the languages share many linguistic properties, including having similar morphological processes. The lexica, however, are different. The authors apply a tagging method used with one language variety to the data of a similar language variety, resulting in a usable tool for a language that is otherwise bereft of NLP tools. The language modeling framework proposed in this dissertation is very much in the spirit of this kind of resource sharing.

## CHAPTER 8: BUILDING AND TESTING THE LEVANTINE LANGUAGE MODELS FOR ASR

In this chapter we test the assumption that the stem-finder transducer framework is more effective at finding useful morphemes for language modeling in colloquial Arabic than the other morpheme modeling methods discussed in Chapter 4. This should be the case because, whereas the affix-finder and MADA methods use MSA-specific information to derive morphemes from words, the stem-finder method relies on cross-dialectally valid information to derive the possible morphemes. We build word- and morpheme-based language models using the same or similar methods as for MSA, this time using a data set comprised of Levantine Conversational Arabic (LCA) speech. Both text-based and ASR evaluations of the LCA language models are included in this chapter.

In addition to being a different language variety, the LCA data set is different from the MSA data in its size and linguistic makeup: it is much smaller, and it is conversational telephone data, as opposed to the MSA broadcast news data. All of these characteristics make the LCA data more difficult to process automatically than the MSA set; we therefore expect much lower accuracy for the LCA speech recognition than for the MSA. Nevertheless, we can learn whether morpheme modeling is more or less beneficial in this domain than for the MSA broadcast news, and whether the differences in morpheme-based language models are more or less pronounced for this data set.

For acoustic modeling in LCA, we use the Fisher Levantine data set from the LDC (Maamouri et al. [2007a,b]). A random 10% of the files are held out as development data,

and another 10% is held out as test data. The acoustic training data is approximately 32 hours and 286K words. The development set is about 1 hour long, and contains 9851 words. The test set is also approximately 1 hour long, and contains 10419 words.

Three other corpora are additionally used for building the language models. They are the Babylon, levcts07, and levqt sets, all acquired from LDC ([Makhoul et al. \[2005\]](#), [Appen Pty Ltd \[2007\]](#), [Maamouri et al. \[2005b\]](#)). Each includes a mix of speakers from Lebanon, Jordan, Israel, and Palestine. The levqt and levcts07 data, like the Fisher data, is spontaneous speech collected via casual telephone conversations. The Babylon data is spontaneous but prompted speech: the participants were asked to play the role of refugees, and answered questions such as “How old are you,” and “Where is your pain?” Together, the four corpora used for language modeling constitute 1.6 million tokens. We will find that the differences between these corpora, even if they seem minor, contribute to a decrease in accuracy when the corpora are combined.

The corpora are modified from their original format in UTF-8 encoding to the same encoding used in the MSA experiments, described in Table 4.4. The word model is collected in a straightforward manner from the transcripts. The morpheme models are collected in a manner similar to or the same as was done for the MSA corpora. We use the same alphabets for determining affix and root letters and sequences for the stem-defined and affix-defined finite state models. The MADA-based morphemes are collected using the same MSA-informed morphological and syntactic models as described in Section 4.5. The Morfessor-based morphemes are collected using the same algorithm with the vocabulary derived from the LCA transcripts.

We use SRILM ([Stolcke \[2002\]](#)) to estimate language models over the word and morpheme n-gram counts. All language models are built with Witten-Bell discounting and

Model	Unigrams	Bigrams	Trigrams	4-Grams	5-Grams	6-Grams
Word	76831	585329	111633	50789	18159	6246
Stem	67107	772577	1881558	3079600	3970679	4670470
Affix*	143869	780606	1742289	1865186	1166782	460720
Morfessor	18724	289918	255129	306290	298566	270291
MADA	48740	351253	213887	201130	142761	90173

Table 8.1: Size of each of the language models built on four LCA corpora. The data and therefore the language models are much smaller than for the MSA data. Morpheme models generally encode the data more effectively than words. \*The affix model was pruned in order to encode it as an FSM.

include an open-class unknown word. They are subsequently encoded as finite state machines in order to calculate the average negative log probability of the development and test sets.

## 8.1 Text-based Language Model Analysis

Table 8.1 shows the size of each of the language models discussed in this chapter. While the stem, Morfessor, and MADA models all have fewer unigrams than the word model, again the affix model predicts many more morpheme types than word types. The affix model is also quite large and must be pruned to be accommodated as an FSM. It is normal to find that higher-order n-grams repeat less often than lower-order n-grams, thus many 6-grams are pruned due to low frequencies in all models. In general, across all n-gram orders, the MADA and Morfessor models are near to each other in size, and the stem and affix models have similar sizes. For n-grams larger than unigrams, all morpheme models have more n-grams than the word model.

We can calculate coverage and log probability scores for each of the models. Please refer to Section 4.7 for the description of how each of these measures is calculated. Briefly,

Model	Unigrams	Bigrams	Trigrams	4-Grams	5-Grams	6-Grams
Word	96.76	60.24	10.28	1.64	0.24	0.0
Stem	91.78	71.63	41.38	16.52	5.25	1.81
Affix	97.57	71.18	34.34	12.84	4.05	1.02
Morfessor	<b>98.23</b>	<b>83.46</b>	<b>53.88</b>	<b>28.99</b>	<b>12.98</b>	<b>5.26</b>
MADA	85.15	64.68	39.06	19.99	9.01	3.28

Table 8.2: Percentage of n-gram tokens in the test set that are included in the language model. The Morfessor model has the best coverage for all n-gram orders.

we wish to know how many of the n-grams in the test set are included in the language model (Table 8.2), how many of the out-of-vocabulary unigrams can be formed by combining the morphemes in the model (Table 8.3), what percentage of the higher-order out-of-vocabulary n-grams have backoff probabilities for their lower-order alternatives in the model (Table 8.4), and the average negative log probability of the test set as defined by each model (Table 8.5). For simplicity, we have included only the all-paths ANLP calculations here for the stem-, affix-, and MADA-based morpheme models. That is, for these models, we calculate the ANLP by summing the probabilities given to all paths in the sentence-long lattice by the language model FST.

The Morfessor model shows the best coverage of morphemes in the development set for all n-gram orders. The stem- and affix-based models also have good coverage for the higher n-gram orders, better than the word model. The MADA model does not provide good coverage of the development set, although for n-gram orders higher than 1, its coverage is better than that of the word model.

Table 8.3, describes what percentage of OOV words can be recovered by concatenating morphemes in each model. In the LCA set, the morphemes derived by the affix-defined

Model	Percent word OOVs covered by morphemes
Stem	84.29
Affix	95.29
Morfessor	62.83
MADA	64.40

Table 8.3: Percentage of OOV words that are made up of morphemes in the given model. The affix model can potentially recover the most OOVs.

Model	Bigram	Trigram
Word	61.23	4.09
Stem	72.53	50.06
Affix	86.66	44.88
Morfessor	<b>87.22</b>	<b>51.00</b>
MADA	44.15	5.22

Table 8.4: Percentage of OOV n-grams for which a backoff n-gram (one order lower) and backoff weight exist in the model.

transducers are best equipped to recover the OOV terms by concatenation. The stem-derived morphemes are better designed for this than the Morfessor and MADA morphemes.

As for availability of backoff weights for lower-order n-grams, Table 8.4 shows that the affix- and Morfessor-derived morpheme models fare the best for bigrams, and also do well for trigrams. The stem-based model also includes backoff weights for about half of the trigrams. These three models will therefore be able to make the strongest, most accurate predictions even when backoff is necessary.

The values for average negative log probability over all paths for each model are relatively close (Table 8.5). However, the word model has the lowest scores. Adding context to the morpheme models does not decrease the ANLP scores, in fact it often increases this

Model	Trigrams	4-Grams	5-Grams	6-Grams
Word	<b>3.056</b>	<b>3.056</b>	<b>3.057</b>	<b>3.057</b>
Stem	5.472	5.472	5.724	5.727
Affix	5.139	5.630	5.789	5.813
Morfessor	4.480	4.471	4.471	4.471
MADA	4.076	4.073	4.073	4.073

Table 8.5: Average negative log probability of a development set as determined by summing over all paths. The Morfessor and MADA models encode the test data more effectively than the Stem and Affix models. The word model has the lowest ANLP for all n-gram orders.

measure. This may indicate that the probabilities of the higher-level n-grams are not well-estimated, or simply that they are not often used because the coverage is so low. Because of these scores and the experience working with the MSA data, we will again find it necessary to interpolate morpheme models with word models to derive the most useful language models for decoding.

No clear best model is indicated by these coverage and ANLP scores. While the MADA model does achieve the lowest ANLP of the morpheme models, it scores low on the other measurements. Interestingly, the ANLP scores exhibit the same split among the four models as we saw in ASR results in Chapter 5: the MADA and Morfessor models get similar scores that are better than the similar scores achieved by the stem and affix models. This trend continues in the ASR task with Levantine data.

## 8.2 Levantine Data Training Regime

Training of the LCA acoustic models follows the same procedure as discussed in Chapter 5; only the acoustic data, transcripts, and pronunciation models are different. For word-based decoding, our best score on the development set is 19.82%, much lower than those of

[Abdou et al. \[2004\]](#), [Stolcke \[2006\]](#) for LCA data. These previous studies use much more sophisticated training techniques that we are not able to replicate on our system, so we settle for lower scores that nevertheless allow us to compare the utility of different language models.

For the Levantine data, we use grapheme models: the pronunciation of each word is defined as the sound represented by each of the characters in the word. This means that we expect each grapheme to represent a unique phoneme plus any following short vowel. This approach is used in, for instance, [Nguyen et al. \[2009\]](#), [Sarikaya et al. \[2007\]](#). There is a significant detriment in training with a grapheme model as opposed to a vocalized model. For instance, when we change the MSA pronunciation dictionary to graphemes (37 monophones) and retrain the acoustic model for the broadcast news data, we achieve an accuracy of only 37.42% on the MSA development set using a word-only language model. Several attempts to add vowels to the LCA data using BAMA ([Buckwalter \[2004b\]](#)) were unsuccessful, so our experiments for LCA are all with a grapheme-based acoustic model. In future work, the romanized form given as alternative transcripts should be used for acoustic modeling, as they may produce better results.

Another detrimental aspect of the LCA data is that its sampling rate is only 8kHz, as compared to the 16kHz data used for MSA broadcast news. For comparison purposes, we rebuilt the MSA grapheme-based acoustic model using down-sampled data, and saw the accuracy of the MSA development set decrease even further, to 31.33%. Decoding the Levantine data using the MSA acoustic models did not produce useful results.

As described in Chapter 7, many studies focus on the use of combined data sets for recognizing dialectal Arabic. That is, information from MSA data or models is used to enhance the dialect-based models. We also made an attempt to use the MSA data to enhance

the acoustic models built to decode dialectal data. Rather than begin acoustic model training with a “flat start”, calculating the average of each feature over the Levantine acoustic data, we seeded the models with monophone estimates based on the MSA data. This procedure did not result in a working model. Combining data sets in this manner or in other clever ways for acoustic or language modeling is deserving of further research, but outside the scope of this thesis.

A number of non-speech events, such as laughter or coughing, are included in the transcripts of the Fisher data, all marked with ‘%’. All of these instances are modeled with a single garbage label. Because we do not differentiate between the non-speech events in training, we map them all to the same label (“%Djp”) in the output and the reference texts.

To summarize, while we attempted to make more robust acoustic models through the use of vowelization and adaptation from MSA data, such adaptation was unsuccessful. A more thorough investigation into how the data types could best be combined may be fruitful despite these initial results. For the remainder of this study, we use grapheme models of the Levantine data only.

In the experiments that follow, all of the morpheme language models are in fact mixtures of words and morphemes. As in the MSA data, we collect n-grams over sequences of both words and morphemes, using the properties of FSM morpheme derivation for the stem- and affix-based models, and by not decomposing frequent words for the MADA and Morfessor models. For the stem- and affix-based models, the probability mass of each word is distributed among the word and morpheme paths such that the word arc receives half the weight, and the morpheme paths share the remaining weight of that word (see Figure 5.3, page 80). The mixed model is then interpolated with the word-only model for increased reliability and predictive context.

Model	LM-transcript	LM-alltext
Word	<b>21.53</b>	<b>19.03</b>
Stem	17.89	16.84
Affix	18.15	17.48
Morfessor	19.21	18.59

Table 8.6: Word Accuracy scores on the development set using transcript-only language models and combined-corpus language models. The word model is most effective.

### 8.3 Speech Recognition Results

We were surprised to find that for the Levantine data, combining multiple data sources for language modeling does not produce better results than using the ASR transcripts alone, Table 8.6. The scores are those of the initial decode with that model, without removing illegal morpheme sequences. In fact, the additional resources supply unreliable data, bringing the accuracy score down for all models.<sup>10</sup> Given more time, a more extensive evaluation might show that some subset of the four corpora produce better results than the ASR transcripts alone, even if the concatenation of all four is not a useful corpus. For instance, because the Babylon data set is play-acting rather than conversational speech, its contents may be inconsistent with the rest of the corpora in a way that detracts from the predictiveness of the model. Also, although all four corpora were collected by the LDC and therefore should have been transcribed under the same procedures, some aspects of the transcriptions may be different. For instance, the non-speech events are marked explicitly in the Fisher corpus (used for acoustic modeling), but are not marked or are differently annotated in the other corpora. It might be useful to extract some subset of utterances from the three other corpora to add to the transcript data. Or, a language model could be built for each corpus

<sup>10</sup>The MADA model was slow to collect n-grams. Given the results on the other four models, we decided to not collect the MADA-based n-grams over the entire corpus.

Model	Development Set		Evaluation Set	
	3-gram	4-gram	3-gram	4-gram
Word	<b>21.50</b>	<b>20.28</b>	<b>16.32</b>	<b>17.25</b>
Stem	18.68	18.19	15.55	15.64
Affix	18.69	17.63	15.36	15.72
Morfessor	19.35	19.26	15.67	15.30
MADA	19.59	19.80	<b>16.39</b>	15.94

Table 8.7: Word Accuracy for LCA data, using only the acoustic modeling transcript data for language modeling. The word model is still most effective. The Morfessor and MADA models are slightly better than the Stem and Affix models in some cases.

separately and the probabilities interpolated with a log-linear model. This experimentation is left for future research. In the remainder of the experiments in this chapter, only the transcripts used for acoustic modeling are employed for language modeling.

Learning from our experience with the MSA data, we decode using language models built with interpolated word and morpheme n-gram sequences. The decoding results are shown in Table 8.7. For each system, we used the development set to tune the beam width pruning parameter to 150.0 or 200.0, the insertion penalty and language model weight, and whether to repeat decoding with zero probabilities on illegal morpheme sequences (the Morfessor+word model achieved better scores on the first iteration of decoding). We also expanded the decode lattices with 4-gram models using SRILM lattice-tool command.

As shown in Table 8.7, the word model achieves significantly better results than any of the morpheme+word models, except in the case of the MADA+word model on the evaluation set, where the word-only accuracy rate is matched. Similar to the results in Chapter 5, the Morfessor- and MADA-based morpheme+word models perform better than the FSM-built stem and affix morpheme+word models, although not always significantly. That the

word accuracy of the Morfessor model is among the higher scores of the morpheme models is not wholly surprising, as it is language independent, and therefore should generate plausible morphemes in Levantine Arabic as well as it does in MSA. The accuracy of the MADA model is more surprising, because it is trained on MSA-specific data. The difference between 16.39% accuracy with the MADA model and 15.55% accuracy with the stem model is significant at  $p \leq .05$ . The difference between the Morfessor- and MADA-based models as opposed to the stem- and affix-based models is likely again due to the simple difference that the former two methods allow only one derivation per word, and the latter two methods allow several derivations per word, with the probability mass of each word split among the paths. The shared probability mass, with prior weights set to distribute the probability evenly between word and morpheme paths, causes the FSM-built models to be less predictive. Different weighting methods intended to correct this imbalance are shown for the stem model in Section 8.5.

Expanding the decode lattices with 4-gram FSMs is helpful in four cases: for the word, stem+word, and affix+word models on the evaluation set, and for the MADA+word model on the development set. In all of these cases the improvement is only slight, and non-significant. In all other cases, expanding to 4-grams is harmful, or at best does not help. This indicates that the probabilities associated with the 4-grams are unreliable.

The Morfessor and MADA morpheme models are very different in their implementation, yet very similar in their result. The Morfessor model requires no linguistic knowledge, while the MADA algorithm makes use of grammatical information and a corpus of annotated data to create a model for morpheme decomposition. Nevertheless, the only significant difference between the two when incorporated into this domain appears for the trigram

Model	Development Set		Evaluation Set	
	Best Decode	Rescoring, 3-grams	Best Decode	Rescoring, 3-grams
Word	19.82	–	16.32	–
Stem	18.44	20.04	15.55	16.69
Affix	20.92	19.77	15.36	16.84

Table 8.8: Rescoring the word-based decode lattices with morpheme-based language models. Rescoring is effective to make the morpheme models achieve scores equal to the word model, but not better.

models on the evaluation set. Otherwise, the models perform equivalently. This result is indicative that unless grammatical information is going to be used in downstream processing, the extra work that goes into finding grammatically precise morphemes may not be worthwhile; morphemes derived using a simple, language-independent function may suffice.

The morpheme models are not adding useful information to the decoding process in Levantine Arabic, and as we saw in the MSA experiments, the morphemes in fact detract from the word accuracy rate of the word-only decoding experiments.

## 8.4 Rescoring Word Lattices with Morpheme Models

Table 8.8 shows the results of converting the word-only decode lattices into morpheme+word lattices and composing with the morpheme+word language models. This technique removes the mismatch between the word-based acoustic models and the morpheme-based language models, but also eliminates the chance of improving on the OOV rate. The procedure is the same as that used in Section 5.6. We see that rescoring with the morpheme models achieves a word accuracy just above, but not significantly above, that of the word-only model. The results of rescoring with the morpheme models are better than the results of

regular morpheme decoding, but rescoring does not give a significant increase over decoding with the word-only model. This is true for both the stem-based and affix-based models on the evaluation set. Again, the morpheme models, even when mixed with the word-only models, do not include enough useful information about the language to improve on the predictions made by the words alone.

## 8.5 Implementing the EM algorithm with LCA data

It appears from the development set scores in Table 8.7 that the two morpheme derivation methods involving a single morpheme path through each word, the Morfessor and MADA methods, again outrank the stem and affix methods, both of which propose multiple hypotheses per path. As discussed in Section 5.4, the weakness of the two FSM-based models may be due to the spread of probability mass among the hypotheses, with half the probability mass for each word allotted to the whole-word arc, and the rest split among the morpheme paths. The fact that the probability mass is shared, that it is shared in a way that does not harmonize with linguistic intuition, and that some of the probability mass is attributed to spurious morpheme sequences all contribute to these models' lack of predictive ability as compared to the Morfessor and MADA models. As in Section 5.4.1, we will attempt to counteract this problem by implementing an Expectation-Maximization (EM) algorithm with the sentence-wide FSMs.

We begin by calculating the sentence-wide FSM for each utterance in the corpus. Each word is decomposed according to the stem-finder transducer, with *no* weights attached to the paths, and the words concatenated to make the sentence FSM. The n-gram counts are collected and an initial language model is estimated. This iteration will be labeled EM0. We calculate ANLP over the development text and perform ASR using this language

Model	ANLP	Word Accuracy
Even Weight (as above)	3.13	18.68
No Weight, EM0	3.07	19.40
Weighted, EM1	2.87	19.83

Table 8.9: ANLP and Word Accuracy on LCA data using stem-derived morphemes with different weighting schemes. As with MSA data, no weighting or EM-based weighting is more effective than the evenly-applied weights.

model interpolated with the word-only language model. Next, the EM0 language model is encoded as an FSM, and this is composed with each of the sentence-wide FSMs in the corpus. New n-gram counts are estimated along the weighted paths, and a new language model, EM1, is estimated from those counts. We calculate ANLP and perform ASR again with the EM1 language model. The results of all stem-based morpheme models on the development set are shown in Table 8.9.

Both ANLP and word accuracy improve as we change from the even-weighted priors to non-weighted paths. Therefore, even without removing the potential bias towards morphemes, the number of morpheme paths is not overpowering the number of word arcs to a bad effect (Table 8.9, row 2). Using EM to refine the weights produces an even more predictive model in terms of both ANLP and word accuracy (Table 8.9, row 3). Nevertheless, the best score shown in Table 8.9 does not equal the score achieved by the word-only model on the same data set.

## 8.6 Discussion

The low scores achieved on the Levantine data set prevent us from coming to any strong conclusions about the utility of the morphemes in this context, or the utility of the different morpheme derivation methods. But, it seems that the results exhibit the same trends we

saw in the ASR data in terms of the interaction of words and morphemes. The problem of mismatch between acoustic and language models experienced in the MSA experiments was replicated here, resulting in no improvement with the addition of morphemes. In the case of rescoring, where the acoustic and pronunciation models are used only in conjunction with the word models, the morphemes models achieve slightly, but non-significantly, higher scores than the word model alone. There does not appear to be a difference in using the stem-based or affix-based models for this task. It is unclear whether the Morfessor- or MADA-derived morphemes would have been equally or more useful for rescoring; we are not able to feasibly implement these decompositions within the FSM classes that perform the lattice rescoring.

Despite the surprising result that the MADA-based model exceeds the word accuracy rate of the stem-based model, it remains the case that stem patterns are considered more cross-dialectally consistent than affix definitions or lexica, so that the derived morphemes may be found to be more useful in language modeling for some tasks. We were able to improve the stem-based morpheme model, first by removing the prior on the FSM paths through the corpus sentences, and again by applying language model weights to the arcs of the decomposition, refining the n-gram counts and subsequent LM estimates. This indicates that the stem-based modeling itself is not necessarily at fault, rather the method of choosing from among multiple decompositions is faulty. FSMs are flexible in a way that allows us to easily correct this problem with statistically-derived information, such as through the EM method of applying LM weights to the original sentence FSMs. The discussion in Section 5.7 also highlights the flexibility of the FSMs on which the stem-based model is built to accommodate new information or constraints. This should be particularly useful for dialectal data about which we have some linguistic knowledge for affixes, stem patterns, or

frequency of stem patterns. Chapter 10 will explore the potential of this flexibility again in attempting to predict short vowels for the MSA stem morphemes.

## CHAPTER 9: PREVIOUS RESEARCH ON SHORT VOWELS

Previous chapters have addressed two of three issues posed by working with Arabic natural language processing: the out-of-vocabulary problem and the problem of differing regional dialects. In the next two chapters, we discuss the the third challenge: missing short vowels in Arabic text.

Modern Standard Arabic has six vowels that differ phonemically: long and short /a/, long and short /i/, and long and short /u/. While the long versions appear as characters in text, the short vowels are rarely written. Occasionally, diacritics are used to mark short vowels; these are seen in religious texts and children’s books, as well as in some cases when they are needed for syntactic disambiguation. The short vowels perform mainly grammatical functions. They mark part-of-speech, tense, aspect, and case (Haywood and Nahmad [1965]). Two other phonological events are normally unmarked in the orthography: gemination of root consonants, and the addition of the nasal /n/ after a short vowel at the end of the word to mark indefiniteness, called nunation. These two processes are usually, but not always, included in the task of automatically diacritizing, or ‘vowelizing’ a text.

For training pronunciation models, acoustic models, and language models, the following questions arise: Is it better to use transcripts that are under-specified as to vowel quality, or to add in symbols representing short vowels in the training data? If the latter is better, what process should be used to generate the short vowel symbols? The studies described below take various approaches to answering the questions of whether and how to insert diacritics into the training data. The approaches include:

- using no diacritics in training,
- using only hand-labeled diacritics,
- automatically generating diacritics,
- or some combination of the above.

In some cases, many diacritic possibilities are generated, and downstream processing makes use of the set of hypotheses, for instance in training an acoustic model. Other studies choose a single path from among the multiple hypotheses to generate voweled language modeling data. In other cases, the goal of the processing is to find the correct sequence of diacritics for a given sentence, with the subsequent use of the voweled data left unspecified.

Grapheme models, which include no indication of short vowel quality, gemination or nunation, are used for acoustic modeling when no hand-diacritized data is available and automatic diacritization is not considered. This method is used in [Choueiter et al. \[2006\]](#), [Affy et al. \[2006\]](#), and [Billa et al. \[2002\]](#). The Arabic graphemes are used as the phonetic inventory. Each grapheme models four possibilities: the represented phoneme alone, or followed by any of the three short vowels /i/, /a/, or /u/. The first two studies achieve word error rates of around 30%, while the [Billa et al. \[2002\]](#) study achieves error rates less than 20%. This last result compares quite favorably to the studies described below that include short vowels in the pronunciation models, most of which also achieve WERs around 20%. Direct comparisons between studies are unreliable, as vastly different ASR systems and data sets are under consideration. Nevertheless, from this brief comparison we might learn that the underspecification of vowel quality has a negative effect on word error rate, even when the vowel markings are not included in the output. Other kinds of data or system implementations might make up for the shortcomings under some circumstances.

Manually adding diacritics to Arabic text is a slow, tedious, and expensive task (Kirchhoff and Vergyri [2005]). For that reason, few corpora large enough for use in natural language processing tasks exist with such annotations. The CallHome Egyptian Colloquial Arabic (ECA) data set (Gadalla et al. [1997]) is one such corpus, as it provides more granular information than most other Arabic corpora. In addition to the traditional Arabic-script transcript, a romanized form is given, which includes short vowel information. The data was created by trained transcribers. Kirchhoff et al. [2006] use this diacritized corpus to train their acoustic models; no comparison is given to a non-vocalized training regime, but we can assume that the vowel specification aided in the acoustic modeling. No study in the literature to date has used the romanization available in the Fisher Levantine Conversational Arabic corpus (Maamouri et al. [2007b]) to produce voweled transcripts or pronunciations models for Levantine Arabic.

Use of the Buckwalter Arabic Morphological Analyzer (BAMA) to produce pronunciation variations (as shown in Figure 5.1) is a common method for improving acoustic modeling. In Xiang et al. [2006], the generation of voweled pronunciations with BAMA was performed before various “decompounding” methods were implemented. The authors show that working with automatically diacritized data in both the language model and the acoustic model is beneficial. In particular, pronunciation models are built with the aid of the BAMA tool as well as Arabic Treebank data, and subsequent constraints are enforced to prevent spurious pronunciations of affixes. The constraints are found to be useful in lowering word error rate by 2.0% absolute.

The BAMA tool is again used to enhance acoustic and language modeling data in Mes-saudi et al. [2004]. Here, the MSA acoustic model is trained on diacritized data, where voweled pronunciations are provided either by manual annotation or with the aid of the

BAMA tool. For the transcripts that were diacritized using BAMA, a best path through the possible vowels is chosen by recognizing the training data with the trained acoustic model. The authors then measure the effect of using vowelized or non-vowelized data in the language model. If only transcript data is used for the language model, then the short vowel transcriptions are helpful: the word error rate is decreased when the language model includes short vowels. This result is somewhat surprising, given that the diacritics add to the number of unique word forms in the language modeling data, making the n-gram estimates less reliable, and increasing the out-of-vocabulary rate. Word error rates continue to decrease as more data is added to the language model, including non-voweled additional data. In [Messaoudi et al. \[2006\]](#), the study is extended to find that if manually diacritized transcripts are added to the large non-vocalized language modeling data, the word error rate decreases even further. There is no improvement, however, if automatically diacritized data is then added to the language modeling data. In addition to adding short vowel hypotheses to language modeling data, [Messaoudi et al. \[2006\]](#) also explore the use of vowelized data for acoustic modeling. The acoustic modeling experiments in this study agree with those in [Kirchhoff and Vergyri \[2005\]](#) (described further below); adding the automatically generated short vowel hypotheses to the pronunciation model is beneficial to the ASR system.

The BAMA technique for populating the pronunciation model, which is also used in the experiments described in Chapter 5, does not directly address the questions asked by the remainder of studies discussed below - how can the *correct* diacritization of a given text be chosen and can such diacritizations be used to improve acoustic modeling, especially when added to hand-vowelized data?

[Kirchhoff et al. \[2002\]](#) uses a very simple method for automatic vowelization of the Egyptian Arabic text: maximum-likelihood estimation, given a small training corpus of

hand-vowelized data. Each word in a non-vowelized test set was assigned the vowelized form found most frequently in the training set (assuming multiple vowelized forms, true for only a subset of the examples). For the evaluation set, this produced a word accuracy of 80.1%, which was raised to 83.5% by adding a simple edit-distance algorithm to find the best vowelization for unseen forms. The character-level accuracies increased from 58.6% to 76.9% with the addition of the edit-distance method. This rate of diacritization accuracy is likely high enough to build useful transcripts for acoustic modeling. Enhancing the accuracy of these vowelizations using trained acoustic models is also an option, and was explored in the studies described next.

Vergyri and Kirchhoff [2004] and Kirchhoff and Vergyri [2005] focus on ways in which data might be shared between dialects, specifically MSA and Egyptian Arabic, and especially as concerns short vowel prediction. The two studies compare several ways of adding vowels to MSA text, in order to then use that vowelized text to enhance the smaller corpus of dialectal data. The procedure is to first examine the ways in which acoustic, morphological, and contextual data contribute to the process of automatic diacritization on the MSA text. For the comparison of vowelization methods, a set of 500 MSA words in context are automatically diacritized using four different techniques, and the results are compared to a hand-annotation of the same data. In each method, a series of diacritized possibilities are produced using the Buckwalter Arabic Morphological Analyzer. When a random path is chosen through the vowelized lattices, a character-level error rate of 12.7% is achieved (Kirchhoff and Vergyri [2005]). If acoustic information from trained Egyptian-only acoustic models is aligned with the MSA morphological possibilities and the best path chosen from the combined morphological and acoustic information, the error rate drops to 3.4%. Contextual data gathered from trigram statistics over the morphological tags do not further

reduce the diacritization error. For the shared-data task between MSA and Egyptian Arabic, the morphological and acoustic information method is used to diacritize a larger MSA corpus, and those transcripts are combined with hand-annotated Egyptian data to improve recognition over the Egyptian evaluation sets. The addition of automatically-vowelized MSA data helps to improve word accuracy only modestly, but with statistical significance. Importantly, it did not matter which of the automatic diacritization methods was used to create the vowelized MSA data. Three important conclusions that can be drawn from these studies are that A) using morphological information to influence vowel choice is necessary, B) using acoustic information to guide the vowel choice is useful, and C) when applying automatically-derived vowels to a transcript for acoustic model training, it is okay for the automatically transcribed vowels to contain errors, so long as at least a small set of correctly transcribed data is available to seed the models.

Finite state transducers are successfully employed for diacritization in [Nelken and Shieber \[2005\]](#). The authors compose a series of FSTs designed to account for short vowel placement, gemination, case endings, as well as clitics and hamza alternation to generate diacritized word forms. The FSTs are trained on the diacritized Arabic Treebank, then simply reversed to generate possibilities for undiacritized data. The most likely diacritization is chosen using a trigram language model, also built on the diacritized Treebank data. Unknown words are accounted for using a 4-gram letter model. The generated diacritic sequences are compared to the correct sequences in the Treebank, and the results reach as low as 6.35% diacritic error rate, when case endings are not included. This is a clever way of bootstrapping from a small amount of diacritized data, and learning the rules of diacritization directly from the data.

In summary, the following types of data can be used in building acoustic and language models: completely non-vowelized data, hand-vowelized data (usually from the Egyptian Colloquial Arabic corpus), automatically vowelized data, or some combination of the above. Training acoustic models on hand-vowelized data is the best route; adding other types of data to these models is sometimes helpful. Language models are usually built on non-diacritized data, as that is what is most widely available, but adding diacritics to the language modeling data can be useful in recognition. The most common way to generate pronunciation possibilities for acoustic modeling is to use the Buckwalter Arabic Morphological Analyzer. Adding constraints or performing additional modeling over its output can be beneficial. Alternatively, bootstrapping vowelization rules from a small, hand-vowelized corpus can produce a useful model for automatic vowelization.

In the following chapter, the finite state transducer algorithm described in Chapter 4 is expanded to generate short vowel and gemination hypotheses according to the same grammatical rules used to predict morphemes. The candidate pronunciations, which could be used to train an acoustic model, are compared to those output by the BAMA tool.

## CHAPTER 10: PREDICTION OF SHORT VOWELS WITH STEM-BASED FSMS

In Chapter 2, we discussed the functioning of stem patterns in Arabic morphology. The use of short vowels was briefly mentioned in that discussion and also in the literature review in Chapter 9. To review, some grammatical details such as case, tense, and aspect are defined by the use of particular vowel sounds within and around the stem. A change in vowel quality from /a/ to /u/ between the second and third root letters, for instance, indicates a grammatical change. There are three short vowels, plus the phonological processes of gemination and nunation (the adding of an /n/ sound at the end of the word) that are crucial for proper grammatical knowledge of a word, and to model the word acoustically. However, these phonological events are not normally represented in the orthography of Arabic. Because they are not represented in text, but are important for proper modeling, one challenge of Arabic natural processing tasks is to predict the presence and quality of these phenomena.

In this chapter, we describe how the stem-based morpheme derivation technique with finite state machines described in Chapter 4 allows for the prediction of short vowels and gemination. The implementation is feasible so long as knowledge is available of how stem patterns and short vowels interact. This information is available for Modern Standard Arabic.

Stem Pattern	With Short Vowels
Paradigm II, Imperfect	
RRR	RaR:iRa
RRR	RaR:iRi
RRR	RaR:iRu
RRR	RaR:iR
Paradigm X, Imperfect	
stRRR	staRRiRa
stRRR	staRRiRi
stRRR	staRRiRu
stRRR	staRRiR
Paradigm X, Active Participle	
mstRRR	mustaRRiRun
Paradigm VIII, perfect	
ARtRR	AiRtaRaRa

Table 10.1: Several examples of how short vowel sequences can be paired with specific stem patterns. The colon ‘:’ represents a geminate consonant. The short vowels are /a/, /i/, and /u/. The /n/ represents nunation.

## 10.1 Implementation of Short Vowel Predictions

For each of the thirty stem patterns listed in Table 4.1, we enumerate the possible short vowel sequences, and where in the pattern the short vowels occur. Table 10.1 shows four examples of how short vowel sequences are paired with the non-voweled stem patterns. These grammatical rules have been studied by Arabic linguists for many centuries, and are codified in textbooks such as Haywood and Nahmad [1965]. For the thirty unique non-voweled patterns used to derive morphemes for the language modeling experiments, we have identified 104 variations with vowels. These patterns do not include the short vowel variations that appear on the affix portions of the words, only the vowels that are possible within the stem.

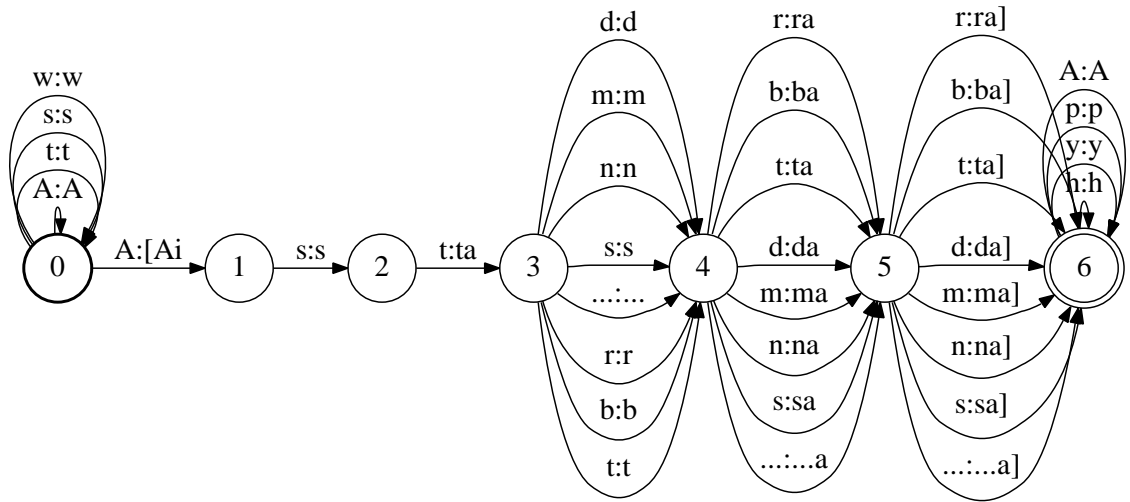


Figure 10.1: An example stem-finder transducer with vowels included on the output tape.

To predict short vowels in tandem with morpheme derivations, we use both the non-voweled and voweled version of each pattern to create new stem-finder transducers. On the input tape of the stem-finder transducer (described in Section 4.1), we have the stem pattern characters only, which compose with the characters in a word. On the output tape, both the characters in the stem pattern and the short vowels that are permitted with that pattern appear. The example stem-finder transducer shown in Figure 4.1 (page 38), now enhanced with vowels is shown in Figure 10.1 above. In this example, the ‘A’ (alif) that begins the stem is followed by the short vowel /i/, the ‘t’ before the first root letter is followed by /a/, and the second and third root letters are followed by the short vowel /a/. The first root letter is not followed by a short vowel. For this particular stem pattern (Paradigm X, perfect tense), this is the only short vowel sequence permissible. However, other stem patterns allow multiple short vowel sequences, in particular in the imperfect. In these cases,

multiple transducers are defined. On each transducer, the input tape is the same - the non-voweled pattern, one letter per arc. On the output tape, the characters include the different vowel patterns possible with the given input pattern.

Successful prediction of short vowels is dependent on whether the word will compose with any of the (non-voweled) patterns. If the word does not compose successfully with any stem-finder transducers, then the word remains whole for further processing, and does not receive any vowel predictions. In the cases where the word does compose with the stem-finder, only the stem portion of the word will include vowel predictions. Affix vowel predictions are possible, but only in a general sense: we have the option of allowing the affix characters to be followed by any of the three short vowels or no short vowel. This was not implemented for processing reasons described below.

## 10.2 Counting N-grams over Voweled Morpheme Sequences

The number of possible morpheme derivations per word increases greatly with the vowel-enhanced transducers. In the following descriptions, we do not add the whole-word arc to morpheme decompositions; each word is either represented as an FSM of morpheme sequences or a whole word, not both. For a particular 10 word sentence taken from the MSA corpus, there are 180 non-voweled paths, and  $2.48 \times 10^8$  voweled paths. This explosion of paths is not only difficult to process and store, but makes n-gram counting (using the efficient method described in Section 4.2 and Appendix A) infeasible. For just the two words “AvnAQ syrh”, there are 10 non-voweled paths and 86 non-voweled 1- through 4-grams. Using the voweled transducers for these same two words, there are 2484 paths and 5945 1- through 4-grams. The slowdown in n-gram collection time is considerable; on a single processor, it took 14 minutes to collect the voweled n-grams of these two words, and

negligible time to collect the non-voweled n-grams. For the first file in the corpus, we find that the sentences have an average of 7.3 words and  $5.5 \times 10^8$  voweled paths,  $1.918 \times 10^{12}$  voweled paths in total. If we assume that for n-gram collection we can process approximately 425 paths per minute on a single processor as indicated by the two-word example above, then it would take 8,585 years to collect 4-grams over a single corpus file. For our training corpus consisting of 1080 files, it would take 9.3 million years! The increase would be even more profound if we included short vowel predictions on the affixes. It will be imperative to use pruning techniques to make this approach feasible for research.

### 10.3 Pruning the Short Vowel Hypotheses

To reduce the size of the output, we must find a way to constrain the possible short vowel sequences. We might design the constraints using linguistic knowledge about the most common sequences, or using statistical methods to derive such knowledge from vowel-annotated corpora. Probabilistic information about vowel sequences can be encoded in FSM format and composed with the FSM-derived hypotheses to apply weights to the paths through the sentence.

Alternatively or additionally, we can apply the same EM method discussed in Section 4.4 to the voweled morpheme FSMs. For instance, starting with a vowel-enhanced sentence FSM, we can calculate its composition with a non-voweled LM to apply weights to the paths through the sentence. Pruning would then be possible over the sentence FSM, reducing the number of paths and n-gram estimates. Using the non-voweled LM for weight application means that all vowel sequences corresponding to a specific stem pattern would be given the same probability. Assuming the paths were pruned to the extent that n-gram collection and LM estimation over the voweled morpheme sequences was possible, the

next step would be to perform another iteration of LM weighting. In this next iteration, the vowel-informed LM would be used for weight application, leading to more pruning, frequency-informed n-gram counts, and refined LM estimation.

As previously discussed, one of the stem-based morpheme derivation method is that it is cross-dialectally useful, because the stem patterns are relatively stable across regional dialects. The same cannot necessarily be said of short vowel patterns across dialects. It would therefore be necessary to collect as much information as possible about the interaction of stem patterns and short vowel sequences within a particular dialect before applying this method to a given dialectal data set. Such knowledge could be given by a language expert, in which case the stem-finder transducers would be defined with this information. Alternatively, the relationships could be learned from a data set annotated with short vowel information, over which a statistical language model including vowels would be built. Given annotated data in a regional dialect and MSA, we could combine language model statistics over vowelized data from both the colloquial and standard forms to make predictions for the non-vowelized dialect data. Formally:

$$P(v_1 \dots v_n | w) = \lambda f(v_1 \dots v_n | w) + (1 - \lambda)g(v_1 \dots v_n | w) \quad (10.1)$$

where  $v_1 \dots v_n$  represents a hypothesized sequence of short vowels for a word  $w$ ,  $f(x)$  is the probability given to that sequence by the dialect-trained language model, and  $g(x)$  is the probability given to that sequence by the MSA-trained language model. Such a method would be similar to the technique described in [Kirchhoff and Vergyri \[2005\]](#) for combining predictions of different language types. In this way, pronunciations with attested vowel sequences would receive higher probability than those with unseen vowel hypotheses.

## 10.4 Building a Vowel-Enhanced Pronunciation Model

Even without the benefit of language model composition, we can use the short vowel predictions to aid in acoustic model development. Rather than use the BAMA tool to derive short vowel pronunciations for each word in the training transcripts, we can use the finite state method to produce hypotheses for each word. In this case, the number of hypotheses is still quite large, but because we are only considering one word at a time, the storage and processing is feasible. Therefore, we created a pronunciation model for the MSA transcripts using this method. In comparing to the BAMA-derived pronunciation model we used in the experiments in Chapter 5, the new model is much larger, with nearly 4 times the number of pronunciations. The BAMA-derived model has 5.03 pronunciations per word on average, whereas the FSM-derived model has 19.41 pronunciations per word on average. The BAMA tool includes a number of lexeme-specific rules that allow only attested pronunciations to appear in the lexicon. Because we do not have such a restriction, and because the stem-finder transducers may produce spurious hypotheses even before vowel prediction is incorporated, the number of pronunciations using the stem-based derivation is much larger than that of the BAMA-derived model. Also, the resulting graphemes are slightly different than those in the BAMA output, which are used in the acoustic models in Chapter 5. For this reason, and because of the size of the new pronunciation model and lack of affix vowel predictions, building acoustic models with the new pronunciations is not within the scope of this work. Such models remain a potentially fruitful area for future research, especially if previously described methods to reduce the number of hypotheses are implemented.

## 10.5 Discussion

The method of using stem-finder transducers to predict short vowels, given a restricted set of short vowel patterns, shows potential. Currently, pronunciation models are often built with information from the MADA or BAMA tools, which provide multiple pronunciation possibilities for each word in their lexicon. The stem-based morpheme derivations also provide such pronunciations, however, there is currently a problem of over-generation of hypotheses that should be addressed using statistical or linguistic knowledge of common short vowel sequences. The predictions of all three models are based on the morphological processes of MSA, and must be adapted to different phonological processes when presented with data from other Arabic dialects. Adding such knowledge to the MADA and BAMA tools will require new lexica, rule definitions, and in the case of MADA, substantial annotated training data. On the other hand, if even a small amount of specific information about a dialect's short vowels is known, it will be relatively easy to change the stem pattern and vowel definitions for the FSM predictions, simply by listing the new stem pattern and short vowel associations. Transducers built on the new definitions can be quickly compiled and employed with dialect data.

In summary, it is a straightforward task to add short vowel predictions to the stem-based finite state machines we use for morphological decomposition. However, it is crucial to use either knowledge- or statistically-based methods to narrow the vowel sequence possibilities before applying the procedure to real data, either for voweled LM estimation or pronunciation modeling.

## CHAPTER 11: CONCLUSION

We have attempted in this work to determine the comparative usefulness of four morpheme derivation techniques for Arabic by building language models for speech recognition with the resulting morphemes. Among four models and two dialects, we found that the utility of building morpheme-based language models was limited on our system. This conclusion does not negate previous literature showing that word accuracy increases through the use of morpheme language models. However, the benefit of morphemes is only realized for some data sets and ASR systems of a given complexity. There is no guarantee that a morpheme-based model will result in higher accuracy than a word model. The word accuracy depends equally on the reliability of the acoustic and pronunciation models and their relationship to the language model terms, and possibly the characteristics of the data, including its size and out-of-vocabulary rate. Other characteristics of morpheme language modeling were discovered in the course of the ASR experiments. Some unexpected differences between the morpheme models were found.

### **11.1 Morpheme Modeling for ASR, Arabic Dialects and Short Vowel Predictions**

The text evaluations of the language models in Chapter 4 showed an improvement in coverage and encoding of the test sets in moving to morpheme-based models, as the theory predicts. While this improvement merited the exploration of these LMs in an ASR system, the predictions about improved word accuracy were not borne out. This may have

been due to a number of factors: a mis-match between the pronunciation dictionary used in training and the one used in testing, lack of context in the predictions of the morpheme-based models, and ambiguity stemming from the acoustic model's predictions of the shorter morphemes as opposed to the longer words. We changed the recognition process to attempt to counteract these influences, by removing illegal sequences from the morpheme language model, and by rescored the word-based decode lattices with morpheme LMs. Despite these changes, adding morpheme information to the word-based language models resulted in our ability to achieve the same results as the word models alone, but no better, as shown in Tables 5.3 and 5.7 on pages 84 and 94, respectively. Creating more sophisticated pronunciation models that account for phonological processes at morpheme boundaries is a worthwhile area for further research with these morpheme models.

We do find some differences between the morpheme-based language models. These are shown in differences in coverage and perplexity as described in Chapter 4. For the ASR task, the two FSM-based morpheme models (stem and affix) work at about the same level of accuracy, as do the Morfessor and MADA models. As discussed in Section 5.4, we believe the main difference between the two pairs of models is the spread of probability mass across multiple hypotheses in the FSM models, weakening the probability of possibly correct morpheme sequences. With the Morfessor and MADA models, no such spread of probability mass occurs, resulting in stronger, more predictive models. An interesting *lack* of difference is seen between the Morfessor and MADA models. As discussed in Section 5.7, the morphemes derived from these two models are equally useful in the speech recognition task, despite the differing amounts of linguistic knowledge required to implement the two morphological decomposition algorithms.

It may be the case that with more data and a better-trained acoustic model, the less linguistically-informed morpheme models will show more improvement on Levantine Conversational Arabic data than those informed by MSA grammar. The cross-dialectal nature of the stem-based method should avoid errors made by a model informed by a different language form. However, in the model built for this work, none of the morpheme models lent useful information to the task, and in fact worsened the word accuracy rate from using a word-only model on LCA data. Given the importance of the task of processing dialectal data, this problem merits further exploration; perhaps it is the case that the out-of-vocabulary rate in colloquial dialects is less severe than that of the standard form, in a way that makes morpheme modeling ill-advised. This was one conclusion of [Creutz et al. \[2007\]](#), and is supported here, although given the overall low accuracy rate of the system, this may not be a reliable conclusion. The use of the stem-based and other morpheme derivation methods across a variety of dialectal data would shed more light on whether the use of morphemes for non-MSA forms of Arabic is in general beneficial, and if so, whether it is important to use models that are designed for broad use across multiple dialects or if techniques and models designed for one language form will be adequate for others.

The results and discussion in Chapter [10](#) highlight the potential of the stem-based derivation system to work as well as the MADA or BAMA tools for deriving vowel pronunciation hypotheses. The accuracy of these stem-based predictions will be improved with a procedure for narrowing the search space of vowel sequence possibilities in MSA, either through statistical or linguistic knowledge. The potential for the proposed method to be ported to other dialects depends on the availability of information about the relationships between stem patterns and short vowels in those dialects. As described in Section [5.7](#), the incorporation of such knowledge into the FSM-implemented stem-based derivations

would likely be much more straightforward than making the same changes in the BAMA or MADA tools. The EM method of refining language models will also be useful in this task.

One aspect of morpheme modeling that became clear in this study is that one morpheme hypothesis per word is preferable to multiple hypotheses. Alternatively, when multiple morpheme hypotheses per word are available, they must be ranked according to statistical knowledge. To this end, the use of the EM method discussed in Section 4.4 is useful. The effect of the method is positively shown for a small data set in Table 4.10 on page 67, as well as in initial ASR results for both MSA (page 87) and LCA (page 135). Using EM on larger datasets is also recommended in instances when parallel processing is possible. On larger data sets, it may take longer for the process to converge to ideal weights. The resulting refined language models may produce higher word accuracy scores on the ASR task. The ability to use EM to refine the stem-derived morpheme language model makes it attractive even though its results were not as good as other models in this study.

What this study provides that is not otherwise available in the literature is a head-to-head comparison of four morpheme derivation techniques, from knowledge-light to knowledge-rich, on a single ASR system. The results show that rather than the amount of linguistic knowledge being the key parameter to success of the morpheme model, the amount of confidence in the morpheme hypotheses is the key. Given that all models were produced and tested on the same data and system, this result should be seen as generalizable. Where only the effect of morpheme modeling on ASR is taken into account, a resource-light morpheme derivation system can be as effective as a more complex, resource-heavy system. However, if other properties are taken into account, and the morphological or short vowel information generated by the Stem or MADA systems is to be

used for more than simply generating morpheme hypotheses, then the research might favor the knowledge-rich method. In this case, the conclusion stated above regarding multiple morpheme hypotheses and use of EM remains important.

## **11.2 Suggestions for Future Work in Arabic ASR**

Further research might describe which of the four methods produces the most accurate morphemes according to linguistic principles; for this, we would expect the MADA method to lead the pack, as it was designed using the most amount of grammatical information. The stem-based and affix-based methods might also fare well as they are based on grammatical insights, however, it is probable that both methods over-generate the possible morpheme decompositions.

Given the discussion above and in Sections 5.7 and 8.6, one suggestion for creating useful Arabic language models for speech recognition is as follows. The acoustic model should be built with word-based transcripts, using a vocalized pronunciation model if possible. To improve the base scores, one might look to improving the acoustic model by adjusting the short vowel predictions, triphone clustering, or adding more data. Given a working word-based language model, it is possible, though dependent on the data set and the system implementation, that adding morpheme sequences to the language model will result in a slight increase in accuracy. If this route is to be taken, the Morfessor algorithm is a good choice for a first implementation, because it is straight-forward and fast, and requires no resources other than a vocabulary and optional frequency counts. It is then simple to use the one-to-one word-to-morphs mapping to transform the corpus into word and morph sequences before counting n-grams to build the language model (as described in Section 4.5).

This is a low-cost, language-independent way of determining whether morpheme modeling will be helpful. If so, more time might be spent adjusting the pronunciation model to account for phonemic changes that occur with morpheme modeling, given the difficulties encountered in incorporating morphemes into the system described in Chapters 5 and 8. If grammatical information is to be incorporated in other ways into the speech recognition system or output, then the work necessary to use more complex tools such as MADA or the stem-based morpheme derivation method might be merited. If, on the other hand, a less-informed model will suffice, the affix-definition method can be employed without great effort, and may produce better results than the Morfessor-derived morphs. It is advisable, as in [Nguyen et al. \[2009\]](#) and [Choueiter et al. \[2006\]](#), to use heuristics to limit the possible morpheme decompositions when using the affix method, in order to limit the number of spurious morpheme hypotheses. It is also highly advisable, with any of these methods, to combine the morpheme hypotheses with the word hypotheses in some way. This can be done through interpolation of word-only and morpheme-only models, and through limiting decompositions to only the less-frequent words. Frequent words will be better modeled in the acoustic and language models if they remain whole.

### **11.3 Other Applications of Morpheme-Based Language Models**

Because one of the main problems encountered in applying the morpheme language models to the speech recognition application is a mis-match with the acoustic models, it may still be the case that these models will be useful for applications not discussed in this dissertation, such as machine translation. In the case of translation, the grammatical rather than the acoustic aspects of the morphemes will be of concern, and will raise new challenges. For instance, Modern Standard Arabic has a specific suffix for marking the dual

number on nouns and verbs. If we are translating into Arabic from a language that does not make that distinction, such as English, it will be difficult to predict when to use this suffix. The morpheme-derivation techniques that employ more grammatical information, such as the BAMA and MADA tools, may provide needed information to downstream processing regarding morphological and syntactic aspects of the words. The Morfessor and affix derivations may be of less use here, as they cannot provide necessary annotation. An aspect discussed briefly in Section 5.7 is the ability of the stem-derivation method to provide such syntactic information, by linking each stem pattern with its associated word class, and providing this information on the output tape. This idea is similar to that discussed in Beesley [1998] and Kiraz [2000]. The utility of adding grammatical information to the stem-finder transducers is left for future work.

The issue of predictive context dealt with in this thesis will apply to other tasks; it is necessary to use a higher-order n-gram model when working with morphemes in order to have the same predictive context that one would have when working with words. However, lack of training data may prevent the higher-order n-grams from acquiring reliable statistics. Interpolation with word-only models is one solution to this problem.

In conclusion, while the experiments described in Chapters 5 and 8 did not produce the expected outcomes, we nonetheless gain knowledge from them. Hesitation in implementing a morpheme-based language model on a new Arabic ASR system is merited, and simpler methods for doing so may work as well as more resource-intensive methods. If morphological information is intended for use beyond morpheme derivation, then cross-dialectal use or ability to predict short vowels should be accounted for in choosing

a derivation method; the stem-based derivation method is the only one of the four methods described in this study for which both are feasible. The problem of creating tools that work cross-dialectally remains a serious, important challenge, and the use of stem patterns to bridge the gap between dialects has not been ruled out as a possible fruitful path. Furthermore, the challenge of predicting short vowels in Modern Standard Arabic as well as colloquial Arabic remains an open question, and one that the stem-finder transducers discussed in this work may help address.

## APPENDIX A: COUNTING N-GRAMS

To build an n-gram language model, we must begin by counting the occurrences of each n-gram of length  $n$  or less. Counting over a typical corpus is very simple; we simply enumerate the occurrences of each n-gram as seen in the text. However, if the corpus is arranged as a series of finite state machines, one FSM per sentence, then the counting process is more complex. In each FSM, there are potentially many paths. At the outset we assume all paths to be equiprobable as we have no evidence by which to assign priors. The count of an n-gram in an FSM is defined, as in [Allauzen et al. \[2005\]](#), as the weighted sum of the number of occurrences of the n-gram in all paths of the FSM. Formally,

$$c(x) = \sum_{w \in \Sigma} |w|_x \llbracket A \rrbracket(w) \quad (\text{A.1})$$

where  $\Sigma$  is the alphabet of our language,  $w$  is a string in that language, and  $x$  is the n-gram (a string) that we are counting. The notation  $|w|_x$  represents the number of times the string  $x$  appears in the string  $w$ .  $A$  is the FSM in which  $x$  occurs, and  $\llbracket A \rrbracket(w)$  is the weight attributed to  $w$  by  $A$ . Therefore, the count of the string  $x$  in FSM  $A$  is estimated as the number of times  $x$  occurs in each string  $w$  appearing in  $A$ , times the weight of the path accepting  $w$  in  $A$ . The remainder of this section will explain in detail the derivation of this equation and its implementation for counting n-grams.

To solve Equation [A.1](#), it seems necessary to enumerate all of the possible strings accepted by  $A$ . Even with an FSM of only moderate size, this task is computationally expensive, and if there are cycles in the graph, a full enumeration is impossible. However,

Allauzen et al. [2003] and Allauzen et al. [2005] describe an alternative counting method that is far less expensive. Briefly, we construct a simple transducer modeled on the n-gram being counted, then compose this transducer with the FSM representing the sentence. After epsilon removal and projection onto the output alphabet, we are left with an FSM  $R$ . The estimated count of the n-gram is equal to the sum of the weights of all of the paths of  $R$ .

To further explain the counting method, we first reproduce the proof given in Allauzen et al. [2003] with an expanded explanation. Then, we describe in detail the counting algorithm, which we use to calculate the estimated count of each n-gram in the FSM without enumerating all paths.

## A.1 Proof of the Efficient N-gram Counting Method

It is first necessary to understand the properties of a formal power series. We start by describing the concept of a semiring, which is laid out in detail in Allauzen et al. [2003]. A semiring is a collection of mathematical operations and a set of numbers over which those operations are defined. The operations are  $\oplus$  and  $\otimes$ , which play the role of addition and multiplication. The operations may be defined in various ways depending on the nature of the set of numbers. The set of numbers is always closed under the two operations, and there always exists an identity element for each operation.

For example, the real semiring can be defined as the collection:

$$\mathbb{K} = \{\mathbb{R}, +, *, 0, 1\}$$

Here we define the  $\oplus$  operation as addition, and the  $\otimes$  operation as multiplication. Zero is the identity element for addition, one is the identity element for multiplication, and the set of real numbers is closed under addition and multiplication.

Another example is the boolean semiring:

$$\mathbb{K} = (\{0, 1\}, \vee, \cdot, 0, 1)$$

We “add” two elements of the set  $\{0,1\}$  with the OR operator ( $\vee$ ); we multiply with the AND operator ( $\cdot$ ). Zero and one are the identity elements of those two operations. The set  $\{0,1\}$  is closed under those operations.

A simple formal power series is

$$S : \Sigma^* \rightarrow \mathbb{K} \tag{A.2}$$

meaning that  $S$  is a power series in which each string in the alphabet  $\Sigma^*$  picks out a coefficient that exists within the semiring  $\mathbb{K}$ .  $S$  is therefore a function. If  $\mathbb{K}$  is our boolean semiring as defined above, then we can redefine  $S$  as

$$S : \Sigma^* \rightarrow \{0, 1\} \tag{A.3}$$

If  $w$  is a string in our alphabet,  $w \in \Sigma^*$ , then the notation  $(S, w)$  can be read as: “the coefficient picked out by the string  $w$  as defined by  $S$ .”

As defined in [Kuich and Salomaa \[1986\]](#), if  $S$  is a formal power series of  $\Sigma^*$  into  $\mathbb{K}$ , the values of  $S$  are denoted by  $(S, w)$ , and  $S$  itself is written as a formal sum:

$$S = \sum_{w \in \Sigma^*} (S, w)w \tag{A.4}$$

To make this more clear, we will now define an example alphabet  $\Sigma$  and one of the functions that can describe how  $S$  and  $\mathbb{K}$  interact.

$$\Sigma = \{a, \varepsilon\} \tag{A.5}$$

$$(S, w) = \begin{cases} 0 & \text{if } |w| \text{ is even} \\ 1 & \text{if } |w| \text{ is odd} \end{cases} \tag{A.6}$$

We choose this definition because it uses only coefficients that exist in boolean semiring  $\mathbb{K}$ . Later, in the proof of the efficient counting algorithm, we will define a different function for  $S$  that picks out coefficients in the probability semiring.

Using the definition of  $S$  in Equation A.6, the sum defined in Equation A.4 can be instantiated as follows:

$$S = (s, \varepsilon)\varepsilon + (S, a)a + (S, aa)aa + (S, aaa)aaa + \dots + (S, a^*)a^* \quad (\text{A.7})$$

and, according to the definition in Equation A.6 :

$$S = 0\varepsilon + 1a + 0a^2 + 1a^3 + \dots \quad (\text{A.9})$$

Again,  $(S, w)$  picks out a coefficient for  $w$ , whereas  $S$  is a list (technically, sum) of coefficients and  $w$  types.

Now we can turn to the proof in Allauzen et al. [2003], and fill out some of the missing steps. In this proof,  $L(X)$  is a regular language:

Let  $S$  be the formal power series (Kuich and Salomaa [1986])  $S$  over the probability semiring defined by  $S = \Omega^* \times x \times \Omega^*$ , where  $x \in L(X)$ .

**Lemma:** For all  $w \in \Sigma^*$ ,  $(S, w) = |w|_x$ . (Allauzen et al. [2003])

Proof of this lemma will allow us to use the efficient method for counting n-grams.

**Proof** as in Allauzen et al. [2003], expanded:

- Let  $\Omega$  be the formal power series  $\Omega : w \rightarrow \mathbb{R}_{\{0,1\}}$ , where  $w$  picks out the coefficient that is the probability of that string in  $\Sigma^*$ .
- $\Omega^*$  is the sum over all  $w$  of the products of the probabilities of all segmentations of  $w$ :

$$\Omega^* = \sum_{0 \leq n < \infty} \Omega^n \quad (\text{A.10})$$

where  $n$  is the number of possible segments of  $w$ . For instance:

$$(\Omega \cdot \Omega, w) = \sum_{w_1 w_2 = w} (\Omega, w_1) \cdot (\Omega, w_2) \quad (\text{A.11})$$

More concretely, if  $w = ab$ , then we calculate  $(\Omega \cdot \Omega, w)$  by summing over all possible two-part segmentations of  $w$ , expanding the last equation as follows:

$$(\Omega \cdot \Omega, w) = (\Omega, \varepsilon) \cdot (\Omega, ab) + (\Omega, a) \cdot (\Omega, b) + (\Omega, ab) \cdot (\Omega, \varepsilon) \quad (\text{A.12})$$

Since we are summing over all probabilities, we should expect the sum to be equal to one. This property is important later in the proof.

We can now define  $(S, w)$  for a particular string  $w$ :

$$(S, w) = (\Omega^* \times x \times \Omega^*, w) \quad (\text{A.13})$$

The associative property holds for the formal power series (Kuich and Salomaa [1986]). This means that we can group the multiplied terms in all possible ways. Therefore:

$$(S, w) = ((\Omega^* \times x) \times \Omega^*, w) \quad (\text{A.14})$$

Splitting the string  $w$  as in Equation A.11:

$$(S, w) = \sum_{w_1 w_2 = w} (\Omega^* \times x, w_1) \times (\Omega^*, w_2) \quad (\text{A.15})$$

where the first term may be split a second time:

$$(S, w) = \sum_{w_1 w_2 = w} \left( \sum_{w_3 w_4 = w_1} (\Omega^*, w_3) \times (x, w_4) \right) \times (\Omega^*, w_2) \quad (\text{A.16})$$

Here,  $x$  is a rational power series that, when the string  $w$  is equal to  $x$ , picks out the coefficient 1. In all other cases, the coefficient is zero. Since we know that  $(x, w_4) \neq 0$  only when  $w_4 = x$ , we can remove the inner sum:

$$(S, w) = \sum_{w_1 w_2 = w} (\Omega^*, w_3) \times (x, w_4) \times (\Omega^*, w_2) \quad (\text{A.17})$$

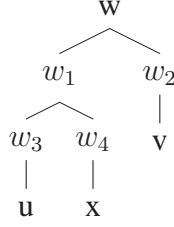


Figure A.1: One way to split up the string  $w \in \Sigma^*$

If we redefine the variables according to the assignments shown in Figure A.1, we can rewrite the equation as:

$$(S, w) = \sum_{w_1 w_2 = w} (\Omega^*, u) \times (x, x) \times (\Omega^*, v) \quad (\text{A.18})$$

We see here that  $(S, w)$  only has a non-zero value when  $x$  is in  $w$ . Also  $(\Omega^*, u)$  and  $(\Omega^*, v)$  should sum to one, as they are sums of all probabilities of  $u$  and  $v$ . Therefore, when  $x$  appears in  $w$ ,  $(S, w) = 1$ . This is the meaning of  $1_{uxv=w}$  in Allauzen et al. [2003]. It also means that we look for  $x$  at every index in  $w$ . Therefore,  $(S, w) = |w|_x$ , because we add one every time we see  $x$  in  $w$ , regardless of what the other elements of  $w$  are. This is the proof of Lemma 1 above.  $\square$

We wish to be able to count  $|w|_x$  without enumerating all possible segmentations of  $|w|$ . To do this, we create a transducer  $T$  such that  $\llbracket T \rrbracket(x, y) = \llbracket T \rrbracket(w, x)$ . Such a transducer has self-loops at the start and end states for each symbol in the alphabet, and transitions between the start and end states that echo the symbols in  $x$  exactly. We can then state:

$$\llbracket T \rrbracket(w, x) = (S, w) = |w|_x \quad (\text{A.19})$$

We are looking for instances of the string  $x$  in a weighted finite state automaton  $A$ , which we will define to be in the probability semiring. The count of  $x$  must take into

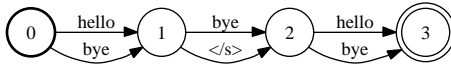


Figure A.2: The WFSA representing our corpus, all transitions have weight 1.

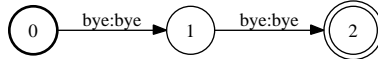


Figure A.3: FSA representing the n-gram  $x$  we wish to count occurrences of in  $A$ .

account the weights on the paths of  $A$ . Therefore, as defined in [Allauzen et al. \[2005\]](#):

$$c(x) = \sum_{w \in \Sigma^*} |w|_x \llbracket A \rrbracket(w) \quad (\text{A.20})$$

Again, we will use  $\llbracket T \rrbracket(w, x)$  to represent  $|w|_x$ . The equivalent of multiplication for automata is composition, so we find the count of  $x$  in  $A$  by composing  $A$  with the transducer  $T$  defined above, removing epsilon transitions, and projecting the result onto the output alphabet. We use  $\oplus$  to sum the weights of the resulting paths to calculate the count of the string  $x$  in  $A$ .

## A.2 Implementing the Efficient Counting Algorithm

The weighted FSA in Figure A.2 is a representation of our example corpus,  $A$ . In it, each transition has a weight of 1. The weights are in the probability semiring. The strings accepted by that WFSA are listed in Table A.1. The total weight attributed to a path accepting any of these strings is 1.

hello bye hello	bye bye hello
hello </s> hello	bye bye bye
hello bye bye	bye </s> hello
hello </s> bye	bye </s> bye

Table A.1: All paths of  $A$ , all have weight 1.

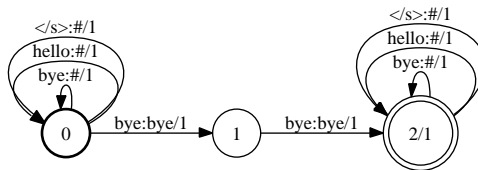


Figure A.4: The weighted transducer  $T$ , as defined in [Allauzen et al. \[2003\]](#), that allows us to count the occurrences of  $x$  of in  $A$ .

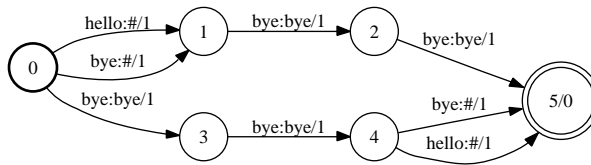


Figure A.5: The composition of  $A$  and  $T$ .

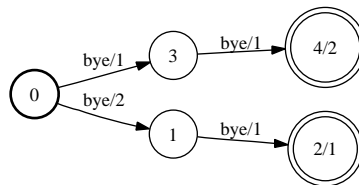


Figure A.6: The WFSM in Figure [A.5](#) having performed epsilon removal and projection onto the output symbols.

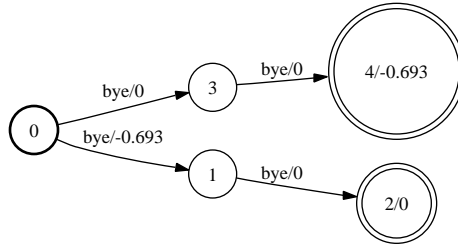


Figure A.7: Composition of  $A$  and  $T$  in the log semiring after projection onto output labels and epsilon removal. We use log addition to sum in the log semiring; the exponent of the result equals 4.

The example n-gram  $x$  that we will count is “bye bye”. It is represented as a simple FSA in Figure A.3.

The transducer  $T$  that we build in order to derive the count of  $x$  in  $A$  is shown in Figure A.4.

In Step 1, we compose WFSAs  $A$  and  $T$ . The result is shown in Figure A.5.

In Step 2, we remove the epsilons from the previous WFSAs, and project the result onto the output symbols, as shown in Figure A.6.

To calculate the count of  $x$  in  $A$ , Step 3, we sum the weights of the paths in this last WFSAs, which is equal to four. We can verify in Table A.1 that the n-gram “bye bye” is found in the WFSAs  $A$  a total of four times.

Note that in the probability semiring,  $\oplus$  is addition and  $\otimes$  is multiplication. This contrasts with the operations in the log and tropical semirings more commonly used in weighted automata, where  $\oplus$  is *logadd* and *min* respectively, and  $\otimes$  is addition. The final automaton in the log semiring is shown in Figure A.7.

Therefore, having built a sentence-long morpheme FSM  $A$ , we first determine the list of n-grams to be counted within that sentence, then create a transducer  $T$  describing each of

the n-grams (in fact this can be done with a single transducer). We compose this transducer  $T$  with FSM  $A$ . We further compose the result with an FSM representing each n-gram individually. These separate compositions result in FSMs  $R$  from which we can calculate the count of each n-gram in that sentence (such as the FSM in Figure A.7). These counts are stored and combined with counts from all other sentences in the training corpora. This process can be done efficiently using the MapReduce algorithm in a Hadoop cluster (Dean and Ghemawat [2004]). Many training files can be read at once over a series of computers, each with several processors. The output of each Map function is a set of n-grams and their counts. The counts are combined using the Reduce function. This is possible because morpheme decomposition and n-gram counting over any portion of the corpus does not depend in any way on the morphemes or n-gram counts derived from any other portion.

C++ code to perform this n-gram counting algorithm over any FST will be published on the author's website.

## Bibliography

- Sherif Abdou, Tresi Arvizo, Amin Atrash, Thomas Colthurst, Chia-Lin Kao, Owen Kimball, Jeff Ma, John Makhoul, Spyros Matsoukas, Rohit Prasad, Dongxin Xu, and Bing Zhang. The 2004 BBN Levantine Arabic and Mandaring CTS transcription systems. In *RT-04 Workshop*, Palisades, NY, 2004. 128
- Mohamed Afify, Ruhi Sarikaya, Hong-Kwang Jeff Kuo, Laurent Besacier, and Yuqing Gao. On the use of morphological analysis for dialectal Arabic speech recognition. In *Proceedings of ICSLP 2006*, 2006. 3, 10, 20, 27, 39, 47, 97, 98, 117, 118, 120, 139
- Mahmoud Al Batal. Identity and language tension in Lebanon: The Arabic of local news at LBCI. In Aleya Rouchdy, editor, *Language Contact and Language Conflict in Arabic: Variations on a Sociolinguistic Theme*, pages 91–115. RoutledgeCurzon, 2002. 115
- Imad A. Al-Sughaiyer and Ibrahim A. Al-Kharashi. Arabic morphological analysis techniques: A comprehensive survey. *Journal of the American Society for Information Science and Technology*, 55(3):189–213, 2004. 9
- Enam Al Wer. Education as a speaker variable. In Aleya Rouchdy, editor, *Language Contact and Language Conflict in Arabic: Variations on a Sociolinguistic Theme*, pages 41–53. RoutledgeCurzon, 2002. 112

- Cyril Allauzen, Mehryar Mohri, and Brian Roark. Generalized algorithms for constructing statistical language models. In *Proceedings of ACL 2003*, pages 40–47, 2003. [xix](#), [9](#), [43](#), [162](#), [164](#), [166](#), [168](#)
- Cyril Allauzen, Mehryar Mohri, and Brian Roark. The design principles and algorithms of a weighted grammar library. *International Journal of Foundations of Computer Science*, 16:403–421, 2005. [41](#), [44](#), [81](#), [161](#), [162](#), [167](#)
- Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. OpenFst: A general and efficient weighted finite-state transducer library. *Lecture Notes in Computer Science*, 4783:11–23, 2007. [44](#), [51](#)
- Mahdi Alosh. *Ahlan wa Sahlan: Functional Modern Standard Arabic for Beginners*. Yale University Press, 2000. [xvi](#), [113](#)
- Australia Appen Pty Ltd, Sydney. Levantine Arabic Conversational telephone speech, transcripts. <http://www ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2007T01>, 2007. [123](#)
- Mohamed Attia, Mohsen A. A. Rashwan, and Mohamed A. S. A. A. Al-Badrashiny. Fassieh<sup>®</sup>, a semi-automatic visual interactive tool for morphological, PoS-tags, phonetic, and semantic annotation of Arabic text corpora. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(5), 2009. [26](#)
- Kenneth R Beesley. Arabic morphology using only finite-state operations. In Michael Rosner, editor, *Computational Approaches to Semitic Languages: Proceedings of the Workshop*, pages 50–57. Montréal, Québec, 1998. [9](#), [24](#), [33](#), [159](#)

- Mohamed Benrabah. Attitudinal reactions to language change in an urban setting. In Yasir Suleiman, editor, *Arabic Sociolinguistics: Issues & Perspectives*, pages 213–226. Curzon Press Ltd, 1994. 106, 108, 109
- Fadi Biadisy, Nizar Habash, and Julia Hirschberg. Improving the Arabic pronunciation dictionary for phone and word recognition with linguistically-based pronunciation rules. In *HLT:NAACL 2009*, pages 397–405, Boulder, CO, 2009. 73
- J. M. Noamany Billa, A. Srivastava, D. Liu, R. Stone, J. Zu, J. Makhoul, and F. Kubala. Audio indexing of broadcast news. In *ICASSP 2002*, pages 5–8, 2002. 73, 139
- Haim Blanc. Style variations in spoken Arabic: A sample of interdialectal educated conversation. In Charles A. Ferguson, editor, *Contributions to Arabic Linguistics*, pages 81–156. Harvard University Press, 1960. 112
- Haim Blanc. *Communal Dialects in Baghdad*. Harvard University Press, Cambridge, Massachusetts, 1964. 114
- Tim Buckwalter. Issues in Arabic orthography and morphology analysis. In *COLING 2004*, pages 31–34, 2004a. 8, 52
- Tim Buckwalter. Buckwalter Arabic morphological analyzer version 2.0. <http://www ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2004L02>, 2004b. 25, 49, 71, 128
- Stanley Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the ACL*, pages 310–318, Santa Cruz, CA, 1996. 4

- Ghinwa Choueiter, Daniel Povey, Stanley F. Chen, and Geoffrey Zweig. Morpheme-based language modeling for Arabic LVCSR. In *ICASSP 06*, pages 1053–1056, 2006. [29](#), [73](#), [83](#), [95](#), [97](#), [139](#), [158](#)
- Mathias Creutz and Krista Lagus. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor. Report A81, Helsinki University of Technology, March 2005. [48](#), [49](#), [82](#)
- Mathias Creutz, Teemu Hirsimäki, Mikko Kurimo, Antti Puurula, Janne Pytkönen, Vesa Siivola, Matti Varjokallio, Ebru Arisoy, Murat Saraçlar, and Andreas Stolcke. Analysis of morph-based speech recognition and the modeling of out-of-vocabulary words across languages. In *Proceedings of NAACL HLT 2007*, pages 380–387, Rochester, NY, 2007. [6](#), [22](#), [30](#), [31](#), [97](#), [118](#), [155](#)
- Martine Cuvalay-Haak. *The Verb in Literary and Colloquial Arabic*. Mouton de Gruyter, Berlin, 1997. [7](#), [35](#)
- K. Darwish. Building a shallow Arabic morphological analyser in one day. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 1–8, Philadelphia, PA, 2002. [28](#), [118](#)
- Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, pages 137–150, San Francisco, 2004. [170](#)
- Mona Diab and Nizar Habash. Arabic dialect processing. In *Proceedings of NAACL 2007*, Rochester, NY, 2007. [6](#)

- F. Diehl, M.J.F. Gales, M. Tomalin, and P.C. Woodland. Morphological analysis and decomposition for Arabic speech-to-text systems. In *INTERSPEECH 2009*, pages 2675–2678, 2009. [26](#), [49](#), [73](#), [97](#)
- Amr El-Desoky, Christian Gollan, David Rybach, Ralf Schlüter, and Hermann Ney. Investigating the use of morphological decomposition and diacritization for improving Arabic LVCSR. In *INTERSPEECH 2009*, pages 2679–2682, 2009. [23](#), [26](#), [49](#), [73](#), [81](#), [97](#)
- Ahmad Emami, Imed Zitouni, and Lidia Mangu. Rich morphology based n-gram language models for Arabic. In *Interspeech 2008*, pages 829–832, Brisbane, Australia, 2008. [28](#), [97](#)
- Charles A. Ferguson. Diglossia. *Word*, 15:325–340, 1959. [105](#)
- Charles A. Ferguson. Diglossia revisited. *Southwest Journal of Linguistics*, 10(1):214–34, 1991. [111](#)
- Hassan Gadalla, Hanaa Kilany, Howaida Arram, Ashraf Yacoub, Alaa El-Habashi, Amr Shalaby, Krisjanis Karins, Everett Rowson, Robert MacIntyre, Paul Kingsbury, David Graff, and Cynthia McLemore. CALLHOME Egyptian Arabic transcripts. <http://www ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC97T19>, 1997. [118](#), [140](#)
- P. Geutner. Using morphology towards better large-vocabulary speech recognition systems. In *Proceedings of ICASSP-95*, volume 1, pages 445–448, 1995. [22](#)
- P. Geutner, M. Finke, and P. Scheytt. Adaptive vocabularies for transcribing multilingual broadcast news. In *ICASSP 1998*, pages 925–928, 1998. [23](#)

- Maik Gibson. Dialect levelling in Tunisian Arabic: Towards a new spoken standard. In Aleya Rouchdy, editor, *Language Contact and Language Conflict in Arabic: Variations on a Sociolinguistic Theme*, pages 24–40. RoutledgeCurzon, 2002. 108, 109
- Umit Guz, Benoit Favre, Dilek Hakkani-Tür, and Gokhan Tur. Generative and discriminative methods using morphological information for sentence segmentation of Turkish. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(5):895–903, 2009. 22
- Nizar Habash and Owen Rambow. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 573–580, Ann Arbor, 2005. 25, 49
- Nizar Habash, Owen Rambow, and George Kiraz. Morphological analysis and generation for Arabic dialects. In *Proc. ACL Workshop on Computational Approaches to Semitic Languages*, pages 17–24, 2005. 23, 33, 120, 121
- Jiri Hana, Anna Feldman, and Chris Brew. A resource-light approach to Russian morphology: Tagging Russian using Czech resources. In *Proceedings of EMNLP 2004*, Barcelona, Spain, 2004. 121
- J.A. Haywood and H.M. Nahmad. *A New Arabic Grammar of the Written Language*. Lund Humphries, Burlington, VT, 1965. xvi, 6, 12, 36, 110, 138, 146
- Teemu Hirsimäki, Mathias Creutz, Vesa Siivola, Mikko Kurimo, Sami Virpioja, and Janne Pylkkönen. Unlimited vocabulary speech recognition with morph language models applied to Finnish. *Computer Speech and Language*, 20:515–541, 2006. 22, 30

- Clive Holes. *Modern Arabic: structures, functions, and varieties*. Longman Group Limited, Harlow, Essex, England, 1995. [xvi](#), [107](#), [108](#), [113](#)
- John Holmes and Wendy Holmes. *Speech Synthesis and Recognition, 2nd Edition*. Taylor & Francis, New York, 2001. [69](#)
- Lufti Hussein. *Levantine Arabic for Non-Natives: Proficiency-Oriented Approach; Teacher's Manual Parts I & II*. The Ohio State University, 1991. [xvi](#), [106](#), [107](#), [110](#), [113](#)
- D. Kiecza, T. Schultz, and A. Waibel. Data-driven determination of appropriate dictionary units for Korean LVCSR. In *Proceedings of the Int'l Conference on Speech Processing*, pages 323–327, 1999. [31](#)
- G.A. Kiraz. Multi-tiered nonlinear morphology using multi-tape finite automata: A case study on Syriac and Arabic. *Computational Linguistics*, 26(1):77–105, 2000. [9](#), [24](#), [25](#), [33](#), [159](#)
- K. Kirchhoff, J. Bilmes, J. Henderson, R. Schwartz, M. Noamany, P. Schone, G. Ji, S. Das, M. Egan, F. He, D. Vergyri, D. Liu, and N. Duta. Novel approaches to Arabic speech recognition: Report from the 2002 Johns-Hopkins summer workshop. Technical report, Johns Hopkins University, 2002. [7](#), [97](#), [141](#)
- Katrin Kirchhoff and Dimitra Vergyri. Cross-dialectal data sharing for acoustic modeling in Arabic speech recognition. *Speech Communication*, 46:37–51, 2005. [28](#), [31](#), [140](#), [141](#), [142](#), [150](#)
- Katrin Kirchhoff and Mei Yang. Improved language modeling for statistical machine translation. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, 2005. [5](#)

- Katrin Kirchhoff, Dimitra Vergyri, Kevin Duh, Jeff Bilmes, and Andreas Stolcke. Morphology-based language modeling for conversational Arabic speech recognition. *Computer Speech and Language*, 20(4):589–608, 2006. **xviii, 3, 20, 28, 61, 73, 97, 99, 118, 140**
- Junbo Kong and David Graff. TDT4 multilingual broadcast news speech corpus. <http://www ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2005S11>, 2005. **51**
- W. Kuich and A Salomaa. *Semirings, Automata, Languages*. Springer-Verlag, New York, 1986. **163, 164, 165**
- Young-Suk Lee, Kishore Papineni, Salim Roukos, Ossama Emam, and Hany Hassan. Language model based Arabic word segmentation. In *Proceedings of the 41st Annual Meeting for ACL*, volume 1, pages 399–406, 2003. **29, 47**
- Linguistic Data Consortium. Arabic transliteration/encoding chart. <http://www ldc.upenn.edu/myl/morph/buckwalter.html>, 2008. Accessed October 5, 2008. **xiv, 53**
- Mohamed Maamouri, Ann Bies, Tim Buckwalter, Hubert Jin, and Widad Mekki. Arabic Treebank: Part 4 v 1.0 (mpg annotation). <http://www ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2005T30>, 2005a. **25**
- Mohamed Maamouri, Tim Buckwalter, and Hubert Jin. Levantine Arabic QT training data set 4 (speech + transcripts). <http://www ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2005S14>, 2005b. **123**

- Mohamed Maamouri, Tim Buckwalter, Dave Graff, and Hubert Jin. Fisher Levantine Arabic Conversational Telephone Speech. <http://www ldc upenn edu/Catalog/CatalogEntry.jsp?catalogId=LDC2007S02>, 2007a. 122
- Mohamed Maamouri, Tim Buckwalter, Dave Graff, and Hubert Jin. Fisher Levantine Arabic Conversational Telephone Speech Transcripts. <http://www ldc upenn edu/Catalog/CatalogEntry.jsp?catalogId=LDC2007T04>, 2007b. 122, 140
- John Makhoul, Bushra Zawaydeh, Frederick Choi, and David Stallard. BBN/AUB DARPA Babylon Levantine Arabic speech and transcripts. <http://www ldc upenn edu/Catalog/CatalogEntry.jsp?catalogId=LDC2005S08>, 2005. 123
- Joseph McCarthy. A prosodic theory of non-concatenative morphology. *Linguistic Inquiry*, 12(3):373–418, 1981. 12, 25
- Ernest N. McCarus. *A Course in Levantine Arabic*. The University of Michigan, 1979. xvi, 113
- A. Messaoudi, J. Gauvain, and L. Lamel. Arabic broadcast news transcription using a one million word vocalized vocabulary. In *Proceedings of ICASSP 2006*, pages 1093–1096, Toulouse, 2006. 141
- Abdel Messaoudi, Lori Lamel, and Jean-Luc Gauvain. Transcription of Arabic broadcast news. In *INTERSPEECH 2004 - ICSLP*, pages 1701–1704, 2004. 73, 140

- Rani Nelken and Stuart Shieber. Arabic diacritization using weighted finite-state transducers. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 79–86, Ann Arbor, 2005. 143
- Long Nguyen, Tim Ng, Kham Nguyen, Rabih Zbib, and John Makhoul. Lexical and phonetic modeling for Arabic automatic speech recognition. In *INTERSPEECH 2009*, pages 712–715, 2009. 30, 73, 97, 98, 128, 158
- Dilworth P. Parkinson. Speaking Fuṣḥā in Cairo: The role of the ending vowels. In Yasir Suleiman, editor, *Arabic Sociolinguistics: Issues & Perspectives*, pages 179–212. Curzon Press Ltd, 1994. 110, 111
- Thomas Pellegrini and Lori Lamel. Automatic word decompounding for ASR in a morphologically rich language: Application to Amharic. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(5):895–903, 2009. 23
- L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 37:257–286, 1989. 74
- Owen Rambow, David Chiang, Mona Diab, Nizar Habash, Rebecca Hwa, Khalil Sima’an, Vincent Lacey, Roger Levy, Carol Nichols, and Safiullah Shareef. Parsing Arabic dialects. Technical report, Johns Hopkins University, 2005. 120
- Ruhi Sarikaya, Mohamed Afify, and Yuqing Gao. Joint morphological-lexical language modeling (JMLLM) for Arabic. In *ICASSP 2007*, pages 181–184, 2007. 47, 97, 128
- Ruhi Sarikaya, Mohamed Afify, and Brian Kingsbury. Tied-mixture language modeling in continuous space. In *Human Language Technologies: NAACL 2009*, pages 459–467, Boulder, Colorado, 2009. 119

- Ksenia Shalnova, Bruno Golénia, and Peter Flach. Towards learning morphology for under-resourced fusional and agglutinating languages. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(5), 2009. 32
- Ray Slyh, Brian Ore, and Tim Anderson. Decision tree questions for MSA triphones. Air Force Research Laboratories, 2007. 75
- Andreas Stolcke. SRILM - an extensible language modeling toolkit. In *Proc. Intl. Conf. Spoken Language Processing*, Denver, Colorado, 2002. 42, 51, 76, 123
- Andreas Stolcke. Re: lattice-tool question/reference. <http://www.speech.sri.com/projects/srilm/mail-archive/srilm-user/2006-September/4.html>, 2006. 3, 97, 119, 128
- Andreas Stolcke. n-gram discount. <http://www.speech.sri.com/projects/srilm/manpages/ngram-discount.7.html>, 2008. 43
- D. Vergyri, A. Mandal, W. Wang, A. Stolcke, J. Zheng, M. Graciarena, D. Rybach, C. Gollan, R. Schlüter, K. Kirchhoff, A. Fari, and N. Morgan. Development of the SRI/Nightingale Arabic ASR system. In *INTERSPEECH 2008*, pages 1437–1440, 2008. 26, 98
- Dimitra Vergyri and Katrin Kirchhoff. Automatic diacritization of Arabic for acoustic modeling in speech recognition. In *COLING 2004 Computational Approaches to Arabic Script-based Languages*, pages 66–73, 2004. 142
- Dimitra Vergyri, Katrin Kirchhoff, Kevin Duh, and Andreas Stolcke. Morphology-based language modeling for Arabic speech recognition. In *Eighth International Conference on Spoken Language Processing*, 2004. 6, 97

- Hans Wehr. *The Hans Wehr Dictionary of Modern Written Arabic*. Spoken Language Services, Inc., Urbana, IL, 1994. [xiv](#), [12](#), [16](#)
- Michael White. Reining in CCG chart realization. In *Proceedings of the 3rd Int'l Conference on Natural Language Generation (INLG-04)*, pages 182–191, 2004. [5](#)
- Bing Xiang, Kham Nguyen, Long Nguyen, Richard Schwartz, and John Makhoul. Morphological decomposition for Arabic broadcast news transcription. In *Proc. ICASSP 2006*, pages 1089–1092, 2006. [3](#), [10](#), [20](#), [47](#), [73](#), [97](#), [140](#)
- Jim Yaghi and Sane M Yagi. Systematic verb stem generation for Arabic. In *COLING 2004*, pages 23–30, 2004. [26](#)
- S.J. Young, J.J. Odell, and P.C. Woodland. Tree-based state tying for high accuracy acoustic modeling. In *Human Language Technology Conference*, pages 307–312, Plainsboro, NJ, 1994. [74](#)
- Steve Young, Gunnar Evermann, Thomas Hain, Dan Kershaw, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev, and Phil Woodland. *The HTK Book (for HTK Version 3.2.1)*. Cambridge University Engineering Department, 2002. [73](#)
- Imed Zitouni, Xiaoqiang Luo, and Radu Florian. A cascaded approach to mention detection and chaining in Arabic. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(5), 2009. [39](#)