

Language modeling for local and Modern Standard Arabic

Ilana Heintz, Chris Brew

Department of Linguistics
222 Oxley Hall
1712 Neil Avenue
Columbus, OH 43210
heintz.38@osu.edu, cbrew@acm.org

Abstract

We propose a Finite State Machine framework for Arabic Language Modeling. The framework provides several decompositions per word based on the forms of possible stems. The statistical modeling is responsible for ranking the most plausible (prefix)-stem-(suffix) sequences higher than the less plausible decompositions. In addition to being useful for Modern Standard Arabic, we show that the framework is easily applied to colloquial Arabic, which suffers from low amounts of text data for use in Natural Language Processing.

1. Introduction

The language modeling task in Arabic faces these three challenges, among others: Lack of textual resources for the dialectal forms of Arabic, a lack of compatibility of tools and resources between Modern Standard Arabic and the locally spoken forms, and the lack of short vowel representations in written forms of either colloquial or Modern Standard Arabic.¹

It has been shown that one can compensate for the small amount of data available for some Arabic dialects by modeling over morphemes rather than words (Kirchhoff et al., 2006; Choueiter et al., 2006). We propose a finite state machine framework for hypothesizing the morphemes that make up each word. We will show that the FSM framework, which requires minimal definition of the specific word forms within the language, can be used cross-dialectally without loss of functionality. Furthermore, the framework is designed to allow short vowel predictions simultaneously with the decomposition hypotheses.

A number of tools have been developed to determine the morphemes that make up an Arabic word, many of which use finite state tools to do so. For instance, (Habash et al., 2005) present a tool designed to determine the morphological make-up of Arabic words, and include both dialectal and phonological information in their analyses. In (Beesley, 1998), a finite-state framework is proposed that produces highly specific morphological analyses of Arabic words, based on the regularities of how concepts like articles, prepositions, parts of speech, number, gender, and case are expressed. Similarly, (Kiraz, 2000) presents a multi-tape automata approach to morphological analyses of Semitic languages, focusing on fruitfully combining knowledge about roots, patterns, phonological rules, and concatenative morphology.

In these studies and others, the goal is to use as input, or receive as output, a detailed morphological analysis of a term. If the morphological information is given to the FST, the appropriate surface form including all phonologically

and orthographically correct segments is expected as output. If a whole word form is given as input, a single analysis of the word's parts is expected. In order to achieve high accuracy, these studies use pre-determined vocabularies of whole words, stems, or roots.

In this study, we are not concerned in particular with the accuracy of the morphological analyses; rather, we are looking for more prolific output from our system. We eliminate the need for pre-determined lists of words, stems, or roots by being highly permissive in the possible analyses. In doing so, we hope to reduce the out-of-vocabulary problem found in automatic speech recognition by giving the language model more options in morpheme types.

In the balance of the article, we describe the form of the finite state machines, the procedure for evaluating our FSM-based language models (LMs), two baseline experiments, and we analyze our results in the context of a potential speech recognition system.

2. Finite State Framework

Our finite state transducers are designed to decompose each Arabic word into all possible (prefix)-stem-(suffix) combinations. We design one finite state transducer for each of 30 possible patterns. One such transducer is shown in Figure 1. These describe the root letter placement and placement of other pattern consonants, such as alif, nun, taa', siin, and miim, that occur in the augmented patterns. In this study, we do not include the short vowel descriptions that are associated with each pattern; however, the design of this framework is such that vowels can be easily inserted in future work. In addition, the transducers allow an affix to occur on either or both sides of the stem.

In the current study, we assume only this pattern information, and we do not restrict the decompositions in any other way. That is, the affixes may be of any length, and may consist of any letters of the alphabet. The characters that fill the root slots are also unrestricted. We could take a more informed view by restricting the alphabet for either the root or affix letters, or restricting the affixes to a certain size. Instead, we allow the frequencies of the resulting morpheme predictions to determine within the LM which are the most likely decompositions.

¹This work was supported by a student-faculty fellowship from the AFRL/Dayton Area Graduate Studies Institute, and worked on in partnership with Ray Slyh and Tim Anderson of the Air Force Research Labs

3. Procedure

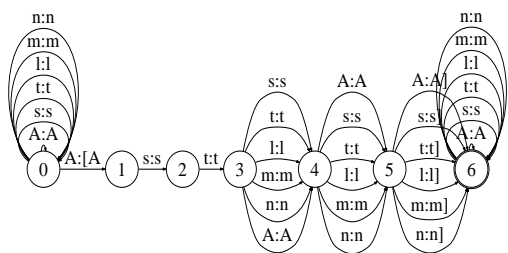


Figure 1: Finite State Transducer for the pattern AstCCC, the pattern for the perfect form of the form X verb, with the example alphabet A,s,t,l,m,n (A represents alif). Affixes may be any length, and can include the same characters as the root and pattern slots. The output will include brackets around the stem portion of the word.

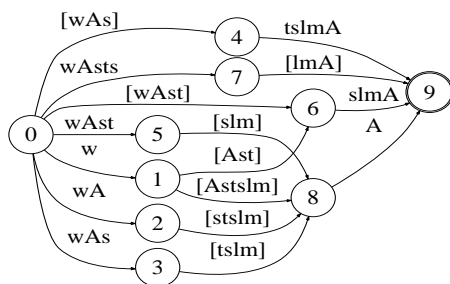


Figure 2: The union of some of the decompositions of the word ‘wAstslmA’, with the possible stems between brackets, and one transition per morpheme. The path labeled 0-1-8-9 is the output of the transducer in Figure 1; the other paths are the output of other patterns’ transducers.

We calculate the union of the 30 pattern FSTs. This set was collected 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 verb paradigms. For this study, the short vowel specifications were removed from the pattern templates, and 30 unique patterns remained. In future studies, we will retain the short vowel information, and apply it to the output of the template transducers.

For each word in the vocabulary, we create a finite state acceptor representing that word, one transition per character, and compose the word acceptor with the pattern transducer. The result is (having performed some additional scripting to augment the AT&T FSM tools), for each word, the union of all possible (prefix)-stem-(suffix) decompositions. This is depicted in Figure 2 for the word *wAstslmA*, “and they (dual) submitted”. In this final representation, each affix and stem are traversed over a single transition, rather than one transition per character. In addition, the stems are identified by surrounding brackets. (In the future, the stems will also include short vowel possibilities, provided by the definitions of the patterns.) Each word is thus represented by a single unweighted morpheme lattice.

We use two sets of text-only data in our experiments. We will show that the FSM framework that we apply to Modern Standard Arabic (MSA) data can also be successfully applied, with no alteration, to Levantine data, a local form of Arabic spoken in Lebanon, Israel, Jordan, and other places in the Middle East. The MSA data is extracted from the TDT4 Broadcast news corpus (Kong and Graff, 2005), a set of radio and TV news show transcriptions. We use 100 files spanning the duration of the data set and each of the five Arabic sources included in the distribution (Agence France Presse, An-Nahar, Al-Hayat, Voice of America, and Nile TV). These files include 104,757 unique word forms and over 1.8 million total word tokens. We convert the original UTF-8 files into the Buckwalter transcription scheme so as to be more straightforwardly compatible with the AT&T Finite State Toolkit (Mohri et al., 1997), and the SRI language modeling toolkit.

The Levantine data is also distributed through the LDC. We use the QT Levantine Training data sets 3 and 4, as well as the BBN/AUB DARPA Babylon transcripts (Maamouri et al., 2005a; Maamouri et al., 2005b; Makhoul et al., 2005). These consist of transcribed spontaneous speech of people living in the US or Lebanon who speak Levantine Arabic. In this case we use all of the data, which has 61905 unique word forms over a total of 1.75 million words.

The composition of the two corpora bely their differences. Whereas the MSA data contains many very long sentences, the dialogue turns in the Levantine data are quite short, often only one or two words. Only 24% of the Levantine vocabulary is covered in the MSA vocabulary, and only 14% of the MSA words are found in the Levantine word list. Nevertheless, we apply the same language modeling procedure to both data sets.

The procedure outlined below is performed separately for the MSA and Levantine data. That is, we do not include the FSM decompositions of one set in the language modeling or evaluation of the other. Rather than share data between language types, we share the language modeling tool. We perform our experiments using a 10-fold cross-validation scheme for both data sets.

We begin by extracting the vocabulary of each corpus, and individually decomposing each word according to the FSM procedure outlined above. The union of all possible decompositions is stored for each word. If a word does not compose successfully with any of the pattern transducers, then the word is stored as its original acceptor: one transition per character. Next, we transform the training sentences into lattices by replacing each word in the sentence with its decomposed finite state representation, and concatenating the FSMs in the order the words occur in the sentence. We use the SRILM toolkit (Stolcke, 2002) to calculate the n-gram statistics over all of the training sentences. All of the paths and morphemes are equally weighted, so we count the frequency of an n-gram as follows:

$$freq(n\text{-gram}) = \frac{\text{occurrences of } n\text{-gram in lattice}}{\text{number of paths in lattice}} \quad (1)$$

Good-Turing smoothing is implemented to allow for unseen words in the test set. We build a 4-gram model for the

data. The LM is limited to the 64000 most common morphemes. (This is the maximum size vocabulary for many speech recognizers.)

Having built the LM, we transform the test sentences into morpheme decomposition lattices, as we did with the training sentences. The weights of the LM are applied to the test lattices, and the best path is extracted. We use the score of the best path in our calculation of average negative log probability, which is described fully in Section 5..

4. Comparisons

We compare the FSM-based LMs to two baseline models: a word-based LM, and a morpheme model built using pre-defined affixes.

We build word-based LMs for each of the ten-fold cross-validation training sets for both Levantine and MSA. Again, we use Good-Turing smoothing and 4-grams.

For the affix model, we begin with the pre-defined set of MSA affixes given in (Xiang et al., 2006). For each word in the vocabulary, we determine one or no (prefix)-stem-(suffix) decompositions, based on the presence of any of the affixes, or sequences of affixes, and the length of the remaining stem. Our heuristics are designed to separate the largest possible affix while assuring that the stem is at least 3 characters long. We transform the training data by replacing each word with its morpheme representation, where morphemes are separated by whitespace. We build the LM over the decomposed training data. Because the decompositions are determined *a priori*, we can transform the test data in the same way, and evaluate the LMs on morpheme data. This baseline is a simplified approximation of the methods used in (Xiang et al., 2006) and (Choueiter et al., 2006).

For the Levantine data, we have two affix models. In the first, we apply the same MSA affixes to the Levantine data. In the second, we define a new set of affixes based on information in a textbook on spoken Levantine (Hussein, 1993). The same procedure as above is applied.

5. Evaluation

In order to evaluate the FSM-based LM against the word-based and affix-based models, we use an adapted measure of perplexity introduced in (Kirchhoff et al., 2006), called average negative log probability. Perplexity measures the average branching factor of the model, and is calculated as:

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

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 FSM and affix models, as n , the number of morphemes, will invariably be larger for those models. Instead, we calculate the average negative log probability as follows:

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

where n is the number of *words* in the test set, as determined by the word model. This allows for a more fair comparison among the models.

	Word	Affix	FSM
Avg Neg Log Prob	4.61	5.17	4.82
Unigram	96.96	99.37	99.10
Bigram	15.75	50.72	64.36
Trigram	0.67	9.61	24.22
4-gram	0.12	2.14	8.62

Table 1: Average negative log probability and n-gram coverage (%) results for the MSA test set.

	Word	Affix-Lev	Affix-MSA	FSM
Avg Neg Log Prob	4.35	5.71	5.68	3.45
Unigram	91.30	77.26	78.77	98.98
Bigram	11.19	10.18	10.76	68.24
Trigram	0.26	0.29	0.29	31.01
4-gram	0.00	0.0	0.0	10.53

Table 2: Average negative log probability and n-gram coverage (%) results for the Levantine test set.

In addition, we calculate the coverage of each size n-gram for each model. For the word and affix models, we simply count how many of the n-gram tokens that appear in the test set are included in the language model. For the FSM model, we calculate the coverage of the language model over the morphemes that appear in the best path of the test set lattices. In addition, we compare the word and FSM-based models by calculating the percentage of word n-grams in the word model exactly accounted for by morpheme n-grams in the FSM-based LMs.

6. Results

In Tables 1² and 2, the results are shown for one fold that represents the pattern found across all folds of the cross-validation scheme. The results in the first row show that the FSM model is a good model of the language. That is, in both cases the average negative log probability of the FSM model is lower than that the affix model. The score of the FSM-based model is lower than the word model in the Levantine case, and very close to the word model in the MSA data set. It is difficult to judge significance for these measures, so it is not appropriate to claim that the FSM model outperforms the other two. The best way to test that claim is to implement the models in an ASR system. These results do show, however, that the FSM model is a promising method for Arabic language modeling. This is especially true in that the same method, with the same background knowledge of standard Arabic morphology, worked well for both Modern Standard Arabic and Levantine Arabic. The coverage statistics in Tables 1 and 2 confirm that the FSM model is a useful method. Coverage of all n-grams is greatest for the FSM model for both languages. This may be because the FSM model results in many one- or two-

²These values differ from those in (Heintz, 2008). In this study, we do not use a single unknown symbol for all OOV terms. Also, we do not include character-wise FSMs in the decomposed representation of each word.

	MSA	Levantine
Unigram coverage (%)	35.84	30.35
Bigram coverage (%)	12.58	43.94

Table 3: Percentage of word unigrams and bigrams covered by the FSM model for each of the two data sets.

	MSA	Lev
word-25%, fsm-75%	5.06	3.33
word-50%, fsm-50%	5.29	3.47
word-75%, fsm-25%	5.70	3.69

Table 4: Average negative log probability results of interpolating the word and fsm models for both Levantine Arabic and MSA, with three interpolation weighting schemes. Results represent a single fold of the cross-validation.

character morphemes. These come from both the affixes and the words that do not compose with the patterns, and are stored as one-character-per-transition acceptors. These morphemes are likely to repeat in both training and test sets; however, they are harder to distinguish in an ASR task.

In addition, we explore how well the morpheme n-grams stored in the FSM-LM cover the word unigrams and bigrams stored in the word-based LM. These results are shown in Table 3. Surprisingly, the coverage is not very good. It seems that the n-grams stored in the FSM-based model cover more inter-word sequences than whole words. Again, this may have to do with the tendency of the FSM model towards short morphemes, which recur often, as opposed to the stems, which are less likely to repeat. One way to amend this would be to store a whole-word morpheme representation for words that do not decompose, rather than storing a character-based morpheme representation.

To test whether interpolation of the word- and FSM-based models is beneficial, we must adjust the FSM representation slightly. We combine the word and morpheme representations by creating a single-transition FSA for each word, and calculating the union of that FSA with the morpheme FST defined earlier for that word. We use these word + morpheme lattices to build the test sentence lattices for the interpolated models. The results for three interpolation weighting schemes are shown in Table 4.

We see that for both MSA and Levantine data, using a heavier weight for the FSM part of the interpolation is best. For the MSA data, the scores for the FSM model and word model shown in Table 1 are lower than any of the interpolated models. For the Levantine data, comparing to Table 2, interpolating the word model into the FSM model, giving more weight to the FSM model, achieves a better score than the FSM model alone. It seems that the extra context gained by interpolating the word model is beneficial for the type of data found in the Levantine corpus, but the same is not true for the more varied data found in the MSA corpus.

7. Conclusion & Future Work

Thus we find that the FSM framework, originally built to work with MSA data, extends to Levantine Arabic. We find that proposing affixes based on probable stem forms works as well as defining the affixes in advance to search

out stems. This method works in large part because the language model statistics are able to rule out the least likely decompositions. Implausible affixes are bound to be less frequent in this model than verifiable affixes.

In future work, we will interpolate this morpheme model with a stem-only model, to see if the increased scope of the stem n-grams can improve the model. Applying the language model weights to the individual word lattices, then rebuilding the model based on weighted decompositions in an E-M process may prove to be beneficial. In addition, we plan to implement these language models in a speech recognition system.

8. References

- Kenneth R Beesley. 1998. 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.
- Ghinwa Choueiter, Daniel Povey, Stanley F. Chen, and Geoffrey Zweig. 2006. Morpheme-based language modeling for Arabic LVCSR. In *ICASSP 06*, pages 1053–1056.
- Nizar Habash, Owen Rambow, and George Kiraz. 2005. Morphological analysis and generation for Arabic dialects. In *Proc. ACL Wrkshp on Comp'l Approaches to Semitic Languages*, pages 17–24.
- J.A. Haywood and H.M. Nahmad. 1965. *A New Arabic Grammar of the Written Language*. Lund Humphries, Burlington, VT.
- Ilana Heintz. 2008. Arabic language modeling with finite state transducers. In *ACL 2008*, Columbus, OH.
- Lufti Hussein. 1993. *Levantine Arabic for Non-Natives*. Yale University Press, New Haven, CT.
- G.A. Kiraz. 2000. Multi-tiered nonlinear morphology using multi-tape finite automata: A case study on Syriac and Arabic. *Computational Linguistics*, 26(1):77–105.
- Katrin Kirchhoff, Dimitra Vergyri, Kevin Duh, Jeff Bilmes, and Andreas Stolcke. 2006. Morphology-based language modeling for conversational Arabic speech recognition. *Computer Speech and Language*, 20(4):589–608.
- Junbo Kong and David Graff. 2005. TDT4 multilingual broadcast news speech corpus.
- Mohamed Maamouri, Tim Buckwalter, and Hubert Jin. 2005a. Arabic cts levantine fisher training data set 3, transcripts.
- Mohamed Maamouri, Tim Buckwalter, and Hubert Jin. 2005b. Levantine arabic qt training data set 4 (speech + transcripts).
- John Makhoul, Bushra Zawaydeh, Frederick Choi, and David Stallard. 2005. Bbn/aub darpa babylon levantine arabic speech and transcripts.
- Mehryar Mohri, Fernando Pereira, and Michael Riley. 1997. At&t FSM Library.
- Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proc. Intl. Conf. Spoken Language Processing*, Denver, Colorado.
- Bing Xiang, Kham Nguyen, Long Nguyen, Richard Schwartz, and John Makhoul. 2006. Morphological decomposition for Arabic broadcast news transcription. In *Proc. ICASSP 2006*, pages 1089–1092.