Arabic Diacritization Using Weighted Finite-State Transducers

Rani Nelken and **Stuart M. Shieber**
Division of Engineering and Applied Sciences
Harvard University
33 Oxford St.
Cambridge, MA 02138
{nelken,shieber}@deas.harvard.edu

Abstract

Arabic is usually written without short vowels and additional diacritics, which are nevertheless important for several applications. We present a novel algorithm for restoring these symbols, using a cascade of probabilistic finite-state transducers trained on the Arabic treebank, integrating a word-based language model, a letter-based language model, and an extremely simple morphological model. This combination of probabilistic methods and simple linguistic information yields high levels of accuracy.

Introduction

Most semitic languages in both ancient and contemporary times are usually written without short vowels and other diacritic marks, often leading to potential ambiguity. While such ambiguity only rarely impedes proficient speakers, it can certainly be a source of confusion for beginning readers and people with learning disabilities (Abu-Rabia, 1999). The problem becomes even more acute when people are required to actively generate diacritized script, as used for example in poetry or children's books. Diacritization is even more problematic for computational systems, adding another level of ambiguity to both analysis and generation of text. For example, full vocalization is required for text-to-speech applications, and has been shown to improve speech-recognition perplexity and error rate (Kirchoff et al., 2002).

We present a system for Arabic diacritization[1] in Modern Standard Arabic (MSA) using weighted finite-state transducers. The system is constructed using standardly available finite-state tools, and encodes only minimal morphological knowledge, yet achieves very high levels of accuracy. While the methods described in this paper are applicable to additional semitic languages, the choice of MSA was motivated by the availability of the Arabic Treebank, distributed by the Linguistic Data Consortium (LDC), a sizable electronic corpus of diacritized text, which we could use for training and testing. Such resources are rare for other semitic languages.

This paper is structured as follows. In Section 1, we describe the task, including a brief introduction to Arabic diacritization, and the corpus we used. In Section 2, we describe the design of our system, which consists of a trigram word-based language model, augmented by two extensions: an extremely simple morphological analyzer and a letter-based language model, which are used to address the data sparsity problem. In Section 3, we report the system's experimental evaluation. We review related work in Section 4, and close with conclusions and directions for future research in Section 5.

1 The task

The Arabic vowel system consists of 3 short vowels and 3 long vowels, as summarized in Ta-

[1] We distinguish between vocalization—the restoration of vowel symbols—and diacritization, which includes additionally the restoration of a richer system of diacritic marks, as explained below.

Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages, pages 79–86,
Ann Arbor, June 2005.

Short vowels				
Vocalized		Unvoc.		Pronounc.
Arab.	*Tran.*	*Arab.*	*Tran.*	
ـُ	u	-	-	/u/
ـَ	a	-	-	/a/
ـِ	i	-	-	/i/
Long vowels				
ـُو	uw	و	w	/u:/
ـَا	aA(1),(2)	ا	A	/a:/
ـَى	aY	ى	Y(3)	/a:/
ـِي	iy	ي	y	/i:/
Doubled case endings				
ـٌ	N	-	-	/un/
ـً	F	-	-	/an/
ـًا	AF	ا	A	/an/
ـٍ	K	-	-	/in/

(1) aA may also appear as A even in vocalized text.
(2) In some lexical items, aA is written as ‘; in which case it is dropped in undiacritized text.
(3) Y and y can appear interchangeably in the corpus (Buckwalter, 2004).

Table 1: Arabic vowels

ble 1.[2] Short vowels are written as symbols either above or below the letter in diacritized text, and dropped altogether in undiacritized text. Long vowels are written as a combination of a short vowel symbol, followed by a vowel letter; in undiacritized text, the short vowel symbol is dropped. Arabic also uses vowels at the end of words to mark case distinctions, which include both the short vowels and an additional doubled form ("tanween").

Diacritized text also contains two syllabification marks: بّ (trans.: ~) denoting doubling of the preceding consonant (in this case, ب), and بْ (trans.: o), denoting the lack of a vowel.

The Arabic glottal stop ("hamza") deserves special mention, as it can appear in several different forms in diacritized text, enumerated in

[2]Throughout the paper we follow Buckwalter's transliteration of Arabic into 7-bit ASCII characters (2002a).

Arab.	*Tran.*	*Arab.*	*Tran.*
أ	>	آ	|
إ	<	ء	'
ؤ	&	آ	{
ئ	}		

Table 2: Arabic glottal stop

Table 2. In undiacritized text, it appears either as A or as one of the forms in the table.

We used the LDC's Arabic Treebank of diacritized news stories (Part 2). The corpus consists of 501 news stories collected from Al-Hayat for a total of 144,199 words. In addition to diacritization, the corpus contains several types of annotations. After being transliterated, it was morphologically analyzed using the Buckwalter morphological analyzer (2002b). Buckwalter's system provides a list of all possible analyses of each word, including a full diacritization, and a part-of-speech (POS) tag. From this candidate list, annotators chose a single analysis. Afterwards, clitics, which are prevalent in Arabic were separated and marked, and a full parse-tree was manually generated. For our purposes, we have stripped all the POS tagging, to retain two versions of each file—diacritized and undiacritized, which we use for both training and evaluation as will be explained below.

An example sentence fragment in Arabic is given in Figure 1 in three forms: undiacritized, diacritized without case endings, and with case endings.

رد الأمين العام لجامعة الدول العربية عمرو موسى...
رَدَّ الأَمِين العَامّ لِجَامِعَة الدُوَل العَرَبِيَّة عَمْرو مُوسَى...
رَدَّ الأَمِينُ العَامُّ لِجَامِعَةِ الدُوَلِ العَرَبِيَّةِ عَمْرو مُوسَى...

Figure 1: Example sentence fragment

The transliteration and translation are given in Figure 2. We follow precisely the form that appears in the Arabic treebank.[3]

[3]The diacritization of this example is (strictly speaking) incomplete with respect to the diacritization of the

```
rd     Al>myn    AlEAm   ljAmEp     Aldwl    AlErbyp        Emrw   mwsY
rad~a Al>amiyn  AlEAm~  lijAmiEap  Alduwal  AlEarabiy~ap  Eamorw muwsaY
rad~a Al>amiynu AlEAm~u lijAmiEapu Alduwali AlEarabiy~api Eamorw muwsaY
```

"Arab League Secretary General Amr Mussa replied..."

Figure 2: Transliteration and translation of the sentence fragment

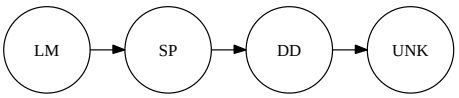

Figure 3: Basic model

2 System design

To restore diacritization, we have created a generative probabilistic model of the process of losing diacritics, expressed as a finite-state transducer. The model transduces fully diacritized Arabic text, weighted according to a language model, into undiacritized text. To restore diacritics, we use Viterbi decoding, a standard algorithm for efficiently computing the best path through an automaton, to reconstruct the maximum likelihood diacritized word sequence that would generate a given undiacritized word sequence.

The model is constructed as a composition of several weighted finite-state transducers (Pereira and Riley, 1997). Transducers are extremely well-suited for this task as their closure under composition allows complex models to be efficiently and elegantly constructed from modular implementations of simpler models.

The system is implemented using the AT&T FSM and GRM libraries (Mohri et al., 2000; Allauzen et al., 2003), which provide a collection of useful tools for constructing weighted finite-state transducers implementing language models.

2.1 Basic model

Our cascade consists of the following transducers, illustrated in Figure 3.

Language model (LM) A standard trigram language model of Arabic diacritized words. For smoothing purposes, we use Katz backoff (Katz, 1987). We learn weights for the model from a training set of diacritized words. This is the only component of the basic model for which we learn such weights. During decoding, these weights are utilized to choose the most probable word sequence that could have generated the undiacritized text. The model also includes special symbols for unknown words, ⟨UNK⟩, and for numbers, ⟨NUM⟩, as explained below.

Spelling (SP) A spelling transducer that transduces a word into its component letters. This is a technical necessity since the language model operates on word tokens and the following components operate on letter tokens. For instance the single token `Al>amiyn` is transduced to the sequence of tokens `A,l,>,a,m,i,y,n`.

Diacritic drop (DD) A transducer for dropping vowels and other diacritics. The transducer simply replaces all short vowel symbols and syllabification marks with the empty string, ϵ. In addition, this transducer also handles the multiple forms of the glottal stop (see Section 1). Rather than encoding any morphological rules on when the glottal stop receives each form, we merely encode the generic availability of these various alternatives, as transductions. For instance, DD includes the option of transducing { to `A`, without any information on when such transduction should take place.

Unknowns (UNK) Due to data sparsity, a test input may include words that did not appear in the training data, and will thus be unknown to the language model. To handle such words, we add a transducer, UNK, that transduces ⟨UNK⟩, into a stochastic sequence of arbitrary letters. During decoding, the letter se-

determiner `Al` and the letter immediately following it. For instance, in `Alduwal`, the `d` should actually have been doubled, yielding `Ald~uwal`. The treebank consistently does not diacritize `Al`, and we adhere to its conventions in both training and testing.

quence is fixed, and since it has no possible diacritization in the model, Viterbi decoding would choose ⟨UNK⟩ as its most likely generator.

UNK plays a similar purpose in handling numbers. UNK transduces ⟨NUM⟩ to a stochastically generated sequence of digits. In the training data, we replace all numbers with ⟨NUM⟩. On encountering a number in a test input, the decoding algorithm would replace the number with ⟨NUM⟩. As a post-processing step, we replace all occurrences of ⟨UNK⟩ and ⟨NUM⟩ with the original input word/number.

2.2 Handling clitics

Arabic contains numerous clitics, which are appended to words, either as prefixes or as suffixes, including the determiner, conjunctions, some prepositions, and pronouns. Clitics pose an important challenge for an n-gram model, since the same word with a different clitic combination would appear to the model as a separate token. We thus augment our basic model with a transducer for handling clitics.

Handling clitics using a rule-based approach is a non-trivial undertaking (Buckwalter, 2002b). In addition to cases of potential ambiguity between letters belonging to a clitic and letters belonging to a word, clitics might be iteratively appended, but only in some combinations and some orderings. Buckwalter maintains a dictionary not only of all prefixes, stems, and suffixes, but also keeps a separate dictionary entry for each allowed combination of diacritized clitics. Since, unlike Buckwalter's system, we are interested just in the most probable clitic separation rather than the full set of analyses, we implement only a very simple transducer, and rely on the probabilistic model to handle such ambiguities and complexities.

From a generative perspective, we assume that the hypothetical original text from which the model starts is not only diacritized, but also has clitics separated. We augment the model with a transducer, Clitic Concatenation (CC), which non-deterministically concatenates clitics to words. CC scans the letter stream; on encountering a potential prefix, CC can non-deterministically append it to the following

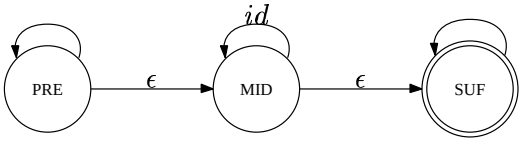

Figure 4: Clitic concatenation

word, merely by transducing the space following it to ϵ. This is done iteratively for each prefix. After concatenating prefixes, CC can non-deterministically decide that it has reached the main word, which it copies. Finally, it concatenates suffixes symmetrically to prefixes. For instance, on encountering the letter string `w Al >myn`, CC might drop the spaces to generate `wAl>myn` ("and the secretary").

The transducer implementation of CC consists of three components, depicted schematically in Figure 4. The first component iteratively appends prefixes. For each of a fixed set of prefixes, it has a set of states for identifying the prefix and dropping the trailing space. A non-deterministic jump moves the transducer to the middle component, which implements the identity function on letters, copying the putative main word to the output. Finally, CC can non-deterministically jump to the final component, which appends suffixes by dropping the preceding space.

By design, CC provides a very simple model of Arabic clitics. It maintains just a list of possible prefixes and suffixes, but encodes no information about stems or possible clitic orderings, potentially allowing many ungrammatical combinations. We rely on the probabilistic language model to assign such combinations very low probabilities.[4]

We use the following list of (undiacritized) clitics: (cf. Diab et al. (2004) who use the same set with the omission of `s`, and `ny`):

prefixes: `b` (by/with), `l` (to), `k` (as), `w` (and), `f` (and), `Al` (the), `s` (future);

suffixes: `y` (my/mine), `ny` (me), `nA` (our/ours), `k` (your/yours), `kmA` (your/yours

[4]The only special case of multiple prefix combinations that we explicitly encode is the combination of l+Al (to + the) which becomes ll, by dropping the A.

masc. dual), km (your/yours masc. pl.), knA (your/yours fem. dual), kn (your/yours fem. pl.), h (him/his), hA (her/hers), hmA (their/theirs masc. dual), hnA (their/theirs fem. dual), hm (their/theirs masc. pl.), hn (their/theirs fem. pl).

We integrate CC into the cascade by composing it after DD, and before UNK. Thus, clitics appear in their undiacritized form. Our model now assumes that the diacritized input text has clitics separated. This requires two changes to our method. First, training must now be performed on text in which clitics are separated. This is straightforward since clitics are tagged in the corpus. Second, in the undiacritized test data, we keep clitics intact. Running Viterbi decoding on the augmented model would not only diacritize it, but also separate clitics. To generate grammatical Arabic, we reconnect the clitics as a post-processing step. We use a greedy strategy of connecting each prefix to the following word, and each suffix to the preceding word.[5]

While our handling of clitics helps overcome data sparsity, there is also a potential cost for decoding. Clitics, which are, intuitively speaking, less informative than regular words, are now treated as lexical items of "equal stature". For instance, a bigram model may include the collocation Al>amiyn AlEAm~ (the secretary general). Once clitics are separated this becomes Al >amiyn Al EAm~. A bigram model would no longer retain the connection between each of the main words, >amiyn and EAm~, but only between them and the determiner Al, which is potentially less informative.

Figure 5 shows an example transduction through the word-based model, where for illustration purposes, we assume that Aldwl is an unknown word.

2.3 Letter model for unknown words

To diacritize unknown words, we trained a letter-based 4-gram language model of Arabic words, LLM, on the letter sequences of words in the training set. Composing LLM with the vowel-drop transducer, DD, yields a probabilistic generative model of Arabic letter and diacritization patterns, including for words that were never encountered in training.

In principle, we could use the letter model as an alternative model of the full text, but we found it more effective to use it selectively, only on unknown words. Thus, after running the word-based language model, we extract all the words tagged as ⟨UNK⟩ and run the letter-based model on them. Here is an example transduction:

Diacritized	Alduwal
LLM	(Weighted)
DD	(Diacritics dropped) Aldwl

We chose not to apply any special clitic handling for the letter-based model. To see why, consider the alternative model that would include CC. Since LLM is unaware of word tokens, there is no pressure on the decoding algorithm to split the clitics from the word, and clitics may therefore be incorrectly vocalized.

3 Experiments

We randomly split the set of news articles in each of the two parts of the Arabic treebank into a training and held-out testing set of sizes 90% and 10% respectively. We trained both the word-based and the letter-based language models on the diacritized version of the training set. We then ran Viterbi decoding on the undiacritized version of the testing set, which consists of a total of over 14,000 words. As a baseline, we used a unigram word model without clitic handling, constructed using the same transducer technology. We ran two batches of experiments—one in which case endings were stripped throughout the training and testing data, and we did not attempt to restore them, and one in which case markings were included.

Results are reported in Table 3. For each model, we report two measures: the word error rate (WER), and the diacritization error rate (DER), i.e., the proportion of incorrectly restored diacritics.

Surprisingly, a trigram word-based language

[5] The only case that requires special attention is ka which can be either a prefix (meaning "as") or a suffix (meaning "your/yours" masc.). The greedy strategy always chooses the suffix meaning. We correct it by comparison with the input text.

Diacritized	`rad~a Al >amiyn Al EAm~ li jAmiEap` ⟨UNK⟩
LM	(Weighted, but otherwise unchanged)
SP	(Change in token resolution from words to letters)
DD	(Diacritics dropped) `rd Al >myn Al EAm l jAmEp`
CC	(Clitics concatenated) `rd Al>myn AlEAm ljAmEp`
UNK	(⟨UNK⟩ becomes `Aldwl`) `rd Al>myn AlEAm ljAmEp Aldwl`

Figure 5: Example transduction

Model	*without case*		*with case*	
	WER	*DER*	*WER*	*DER*
Baseline	15.48%	17.33%	30.39%	24.03%
3-gram word	14.64%	16.9%	28.42%	23.34%
3-gram word + CC	8.49%	9.32%	24.22%	15.36%
3-gram word + CC + 4-gram letter	7.33%	6.35%	23.61%	12.79%

Table 3: Results on the Al-Hayat corpus

model yields only a modest improvement over the baseline unigram model. The addition of a clitic connection model and a letter-based language model leads to a marked improvement in both WER and DER. This trend is repeated for both variants of the task—either with or without case endings. Including case information naturally yields proportionally worse accuracy. Since case markings encode higher-order grammatical information, they would require a more powerful grammatical model than offered by finite-state methods.

To illustrate the system's performance, here are some decodings made by the different versions of the model.

Basic model

- `An`, `<in~a`, and `>an~a`, three versions of the word "that", which may all appear as `An` in undiacritized text, are often confused. As Buckwalter (2004) notes, the corpus itself is sometimes inconsistent about the use of `<in~a` and `>an~a`.
- Several of the third-person possessive pronoun clitics can appear either with a `u` or an `i`, for instance, the third person singular masculine possessive can appear as either `hu` or `hi`. The correct form depends on the preceding letter and vowel (including the case vowels). Part of the tradeoff of treating clitics as independent lexical items is that the word-based model is ignorant of the letter preceding a suffix clitic.

Clitic model

- `wstkwn` was correctly decoded to `wa sa takuwnu`, which after post-processing becomes `wasatakuwnu` ("and [it] shall be").

Letter model

- `AstfzAz` correctly decoded to {`isotifozAz` ("instigation"). This example is interesting, since a morphological analysis would deterministically predict this diacritization. The probabilistic letter model was able to correctly decode it even though it has no explicit encoding of such knowledge.
- Non-Arabic names are obviously problematic for the model. For instance `bwrtlAnd` was incorrectly decoded to `buwrotilAnoda` rather than `buwrotlAnod` (Portland), but note that some of the diacritics were correctly restored. Al-Onaizan and Knight (2002) proposed a transducer for modeling the Arabic spelling of such names for the purpose of translating from Arabic. Such a model could be seamlessly integrated into our architecture, for improved accuracy.

4 Related work

Gal (2002) constructed an HMM-based bigram model for restoring vowels (but not additional diacritics) in Arabic and Hebrew. For Arabic, the model was applied to the Qur'an, a corpus of about 90,000 words, achieving 14% WER. The word-based language model component of our system is very similar to Gal's HMM. The very flexible framework of transducers allows us to easily enhance the model with our simple but effective morphology handler and letter-based language model.

Several commercial tools are available for Arabic diacritization, which unfortunately we did not have access to. Vergyri and Kirchhoff (2004) evaluated one (unspecified) system on two MSA texts, reporting a 9% DER without case information, and 28% DER with case endings.

Kirchoff et al. (2002) focuses on vocalizing transcripts of oral conversational Arabic. Since conversational Arabic is much more free-flowing, and prone to dialect and speaker differences, diacritization of such transcripts proves much more difficult. Kirchoff et al. started from a unigram model, and augmented it with the following heuristic. For each unknown word, they search for the closest known unvocalized word to it according to Levenshtein distance, and apply whatever transformation that word undergoes, yielding 16.5% WER. Our letter-based model provides an alternative method of generalizing the diacritization from known words to unknown ones.

Vergyri and Kirchhoff (2004) also handled conversational Arabic, and showed that some of the complexity inherent in vocalizing such text can be offset by combining information on the acoustic signal with morphological and contextual information. They treat the latter problem as an *unsupervised* tagging problem, where each word is assigned a tag representing one of its possible diacritizations according to Buckwalter's morphological analyzer (2002b). They use Expectation Maximization (EM) to train a trigram model of tag sequences. The evaluation shows that the combined model yields a significant improvement over just the acoustic model.

5 Conclusions and future directions

We have presented an effective probabilistic finite-state architecture for Arabic diacritization. The modular design of the system, based on a composition of simple and compact transducers allows us to achieve high levels of accuracy while encoding extremely limited morphological knowledge. In particular, while our system is aware of the existence of Arabic clitics, it has no explicit knowledge of how they can be combined. Such patterns are automatically learned from the training data. Likewise, while the system is aware of different orthographic variants of the glottal stop, it encodes no explicit rules to predict their distribution.

The main resource that our method relies on is the existence of sufficient quantities of diacritized text. Since semitic languages are typically written without vowels, it is rare to find sizable collections of diacritized text in digital form. The alternative is to diacritize text using a combination of manual annotation and computational tools. This is precisely the process that was followed in the compilation of the Arabic treebank, and similar efforts are now underway for Hebrew (Wintner and Yona, 2003).

In contrast to morphological analyzers, which usually provide only an unranked list of all possible analyses, our method provides the most probable analysis, and with a trivial extension, could provide a ranked n-best list. Reducing and ranking the possible analyses may help simplify the annotator's job. The burden of requiring large quantities of diacritized text could be assuaged by iterative bootstrapping—training the system and manually correcting it on corpora of increasing size.

As another future direction, we note that occasionally one may find a vowel or two, even in otherwise undiacritized text fragments. This is especially true for extremely short text fragments, where ambiguity is undesirable, as in banners or advertisements. This raises an interesting optimization problem—what is the least number of vowel symbols that are required in

order to ensure an unambiguous reading, and where should they be placed? Assuming that the errors of the probabilistic model are indicative of the types of errors that a human might make, we can use this model to predict where disambiguating vowels would be most informative. A simple change to the model described in this paper would make vowel drop optional rather than obligatory. Such a model would then be able to generate not only fully unvocalized text, but also partially vocalized variants of it. The optimization problem would then become one of finding the partially diacritized text with the minimal number of vowels that would be least ambiguous.

Acknowledgments

We thank Ya'akov Gal for his comments on a previous version of this paper. This work was supported in part by grant IIS-0329089 from the National Science Foundation.

References

Salim Abu-Rabia. 1999. The effect of Arabic vowels on the reading comprehension of second- and sixth-grade native Arab children. *Journal of Psycholinguist Research*, 28(1):93–101, January.

Yaser Al-Onaizan and Kevin Knight. 2002. Machine transliteration of names in Arabic texts. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, pages 34–46, Philadelphia, July. Association for Computational Linguistics.

Cyril Allauzen, Mehryar Mohri, and Brian Roark. 2003. Generalized algorithms for constructing statistical language models. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'2003)*, pages 40–47.

Tim Buckwalter. 2002a. Arabic transliteration table. http://www.qamus.org/transliteration.htm.

Tim Buckwalter. 2002b. Buckwalter Arabic morphological analyzer version 1.0. Linguistic Data Consortium, catalog number LDC2002L49 and ISBN 1-58563-257-0.

Tim Buckwalter. 2004. Issues in Arabic orthography and morphology analysis. In Ali Farghaly and Karine Megerdoomian, editors, *COLING 2004 Computational Approaches to Arabic Script-based Languages*, pages 31–34, Geneva, Switzerland, August 28th. COLING.

Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2004. Automatic tagging of Arabic text: From raw text to base phrase chunks. In Susan Dumais, Daniel Marcu, and Salim Roukos, editors, *HLT-NAACL 2004: Short Papers*, pages 149–152, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Ya'akov Gal. 2002. An HMM approach to vowel restoration in Arabic and Hebrew. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, pages 27–33, Philadelphia, July. Association for Computational Linguistics.

Slava M. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–4001, March.

Katrin Kirchoff, Jeff Bilmes, Sourin Das, Nicolae Duta, Melissa Egan, Gang Ji, Feng He, John Henderson, Daben Liu, Mohamed Noamany, Pat Schone, Richard Schwarta, and Dimitra Vergyri. 2002. Novel approaches to Arabic speech recognition: report from the 2002 Johns-Hopkins summer workshop. Technical report, Johns Hopkins University.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231(1):17–32.

Fernando C. N. Pereira and Michael Riley. 1997. Speech recognition by composition of weighted finite automata. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Devices for Natural Language Processing*. MIT Press, Cambridge, MA.

Dimitra Vergyri and Katrin Kirchhoff. 2004. Automatic diacritization of Arabic for acoustic modeling in speech recognition. In Ali Farghaly and Karine Megerdoomian, editors, *COLING 2004 Computational Approaches to Arabic Script-based Languages*, pages 66–73, Geneva, Switzerland, August 28th. COLING.

Shuly Wintner and Shlomo Yona. 2003. Resources for processing Hebrew. In *Proceedings of the MT Summit IX Workshop on Machine Translation for Semitic Languages*, New Orleans, September.