

Diacritization as a Machine Translation Problem and as a Sequence Labeling Problem

Tim Schlippe
Cognitive Systems Lab
Universität Karlsruhe (TH)
Karlsruhe, Germany
schlippe@ira.uka.de

ThuyLinh Nguyen **Stephan Vogel**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{thuylinh, vogel}@cs.cmu.edu

Abstract

In this paper we describe and compare two techniques for the automatic diacritization of Arabic text: First, we treat diacritization as a **monotone machine translation** problem, proposing and evaluating several translation and language models, including **word and character-based models** separately and combined as well as a model which uses statistical machine translation (SMT) to post-edit a **rule-based diacritization system**. Then we explore a more traditional view of diacritization as a **sequence labeling problem**, and propose a solution using conditional random fields (Lafferty et al., 2001). All these techniques are compared through **word error rate and diacritization error rate** both in terms of full diacritization and ignoring vowel endings. The empirical experiments showed that the **machine translation approaches perform better than the sequence labeling** approaches concerning the error rates.

1 Introduction

Modern Arabic texts are normally composed of scripts without diacritic marks. The problem is that many words have different meanings depending on their diacritization. This leads to ambiguity when processing data for text-to-speech and speech-to-text applications. A reduction of this ambiguity with the help of diacritization in a text document may benefit other language processing tasks as well. Diab et al. (2007) report an improvement from 0.4389 to 0.4416 BLEU scores (Papineni et al., 2002) in Arabic-English SMT after inserting the

passivization diacritic “damma”. Since diacritics at the word endings mark the cases in Arabic, a resulting diacritized translation in Arabic language is easier to understand for native speakers even if the word order is wrong. The techniques used for Arabic diacritization are applicable to other languages such as **Romanian, French (Tufis and Chitu, 1999) and Hebrew (Gal, 2002)**.

We study two solutions for the diacritization problem. First, we regard the diacritization problem as a phrase-based translation task. In this case a SMT system is used as a tool for our experiments. We also combine a **rule-based approach** with our SMT methods by post-editing the output of a rule-based diacritizer. Then we solve the problem as a **sequence labeling problem with the help of conditional random fields (CRFs)**. That means we integrate global features to determine our diacritized output sequence and create dependencies between a non-diacritized input sequence, global features such as **part-of-speech tags** and the output sequence by feature functions.

In the next section we describe aspects of the Arabic language and the diacritics and present the related work in Section 3. Section 4 outlines our approaches, while Section 5 explains which data sources we used to establish, tune and test our systems. Our experiments and results are outlined in Section 6. We conclude our work in Section 7 and suggest further approaches based on our results.

2 Arabic Language and Diacritics

The Arabic script is written, read and encoded from right to left. Many Arabic letters change their ap-

pearance depending on their position in a word. The Arabic alphabet consists of 28 consonant letters.

Arabic diacritics are located below and above each character within a word. They are vowelization marks and usually absent. “shadda” is the only diacritic which appears in several modern Arabic scripts. Native speakers distinguish the right pronunciation and the correct meaning of a word without diacritic marks in an automatic way by considering the context and the position of the word in a sentence. Their instinctive knowledge of Arabic grammar and vocabulary enable them to correctly vocalize words in written texts based on the context.

For example, the bare form “Elm” may have different meanings (Figure 1) depending on the diacritization: “Eilm” is translated as “science” or “learning”, while “Ealam” means “flag”. Ambiguity may also occur on the grammatical level as diacritics at word endings are correlated with case and verbal information (Maamouri et al., 2006).

without diacritics	with diacritics	meaning	pronunciation
علم	علم	science, learning	Eilm
	علم	flag	Ealam

Figure 1: Ambiguity in Arabic.

Arabic diacritics can be categorized into

- short vowels pronounced as /a/ (fatha), /u/ (damma) and /i/ (kasra),
- double case endings pronounced as /an/ (fathatayn), /un/ (dammatayn) and /in/ (kasratayn),
- syllabification marks.

Double case endings are vowels used at the end of a word to distinguish cases. The syllabification mark “shadda”, which is sometimes inserted in written text, symbolizes the doubling of a consonant. “sukuun”, the second syllabification mark, indicates that a letter does not contain any vowels. We restored the diacritics shown in Figure 2.

3 Previous Work

Different methods such as rule-based (El-Imam, 2004), example-based, hierarchical (Emam and Fis-

Short vowels /a/, /u/, /i/	Double case ending	Syllabification marks
Fatha	fathatayn	shadda
damma	dammatayn	sukuun
kasra	kasratayn	

Figure 2: Arabic Diacritics.

cher, 2005), morphological and contextual-based (Vergyri and Kirchhoff, 2004) as well as methods with Hidden Markov Models (Mustafa Elshafei and Alghamdi, 2006) and weighted finite state machines (Nelken and Shieber, 2005) have been applied for the diacritization of Arabic text.

Some authors treated diacritization as a machine translation problem. El-Sadany and Hashish (1989) and El-Imam (2004) proposed rule-based methods for the translation from non-diacritized text to diacritized text. One drawback of these systems is the difficulty to keep the rules consistent, up-to-date and extend them to other Arabic dialects. Emam and Fischer (2005) suggested an example-based hierarchical top-down approach, similar to example-based translation approaches. In order to translate a sentence in the test set the system tries to find a matching sentence in the training data. If a matching sentence is found, the whole sentence is applied, if not, the system searches for matching phrases. If no matching phrases are found character n-grams are restored.

Other researchers approached diacritization as a sequence labeling problem. In (Vergyri and Kirchhoff, 2004) each word is tagged as one of many possible forms provided by the Buckwalter’s Morphological Analyser (Buckwalter, 2004). In order to learn the tag sequence the Expectation Maximization algorithm is applied. In this approach the morphological and contextual information is combined with an acoustic signal. The authors reported a word error rate (WER) of 27.3% and a diacritization error rate (DER) of 11.54%. Nelken and Shieber (2005) restored diacritics with an algorithm based on weighted finite state machines. Their system was trained on LDC’s Arabic Treebank Data.

Words not occurring in the data are substituted by characters and larger morphological units. Altogether they achieved a WER of 23.61% and a DER of 12.79%. Ignoring vowel endings, a WER of 7.33% and a DER of 6.35% was achieved. Zitouni et al. (2006) proposed a maximum entropy-based approach. Their system works with lexical, segment-based and part-of-speech tag features. They reported a WER of 18.0% and a DER of 5.5% as well as a WER of 7.9% and a DER of 2.5% ignoring the final vowelization. Mustafa Elshafei and Alghamdi (2006) considered the word sequence of non-diacritized Arabic text as an observation sequence and solved the problem with Hidden Markov Models. The hidden states are the possible diacritized expressions of the words. Finally, the optimal sequence of diacritized words or states are obtained by applying the Viterbi Algorithm. They achieved a DER of 4.1% for a test set of 995 words with their system which was trained on the Qur'an text. A further reduction to about 2.5% was reached by using a preprocessing stage and trigrams for a selected number of short and frequent words. A diacritization system based on the combination of a tagger and a lexeme language model was suggested by Habash and Rambow (2007). Analogous to (Vergyi and Kirchhoff, 2004) their system uses the Buckwalter Arabic Morphological Analyzer. The remaining WER was 14.9% and the DER 4.8%. Ignoring vowel endings, a WER of 5.5% and a DER of 2.2% were reported.

4 Automatic Diacritization

In this section we present our methods for the automatic diacritization.

4.1 Diacritization as Translation

A straightforward approach is treating the undiacritized text as source text and the diacritized text as target text to build a translation model for a phrase-based statistical machine translation system. We create a phrase table with non-diacritized entries on the source side and diacritized entries on the target side as well as a language model with diacritized text. As the alignment is very simple – it is strictly monotone and each diacritized word or phrase has exactly one non-diacritized form – gener-

ating the phrase table is very easy. Starting from the diacritized training corpus, we run over all sentences and all positions in each sentence and write all word n -grams, $n = 1 \dots N$, starting in this position. Removing then the diacritics gives us a list of phrase pairs (with repetitions), from which the phrase table can be build. As phrase translation scores we use the relative frequencies.

Using appropriate representations of the non-diacritized and diacritized text we can operate on the character level, on the word level, or combined on both levels.

4.1.1 Diacritization on the Word Level

The following figure illustrates Arabic text on the word level. The characters are ASCII characters since we are using the Buckwalter Transliteration.

without diacritics	mwskw	Jf	b
with diacritics	muwsokuw	Jaf	b

For many words there are different diacritized forms available. Both, the language model and also the context given within phrases can help to disambiguate in these cases. Due to the limited training data there are words in the test set, which neither occur in the phrase table nor in the language model. Our system "translates" unknown words by inserting them into the output. However, this typically leads to an error.

4.1.2 Diacritization on the Character Level

As proposed for the restoration of diacritics in (Mihalcea, 2002), we also developed a system which works on the character level. The representation of the Arabic text on character level splits undiacritized text into individual consonants and diacritized text into consonant-vowel compounds.

m	w	s	k	w	space	J	f	space
mu	w	so	ku	w	space	Ja	f	space

We insert a special word separator to be able to restore the words after diacritization has been done.

The advantage of operating on the character level is that all words can be diacritized. I.e. we do not have any unknown words, as in the case, when the phrase table contains words. A drawback, however,

is that much less context is covered by the character-based system. A 5-gram covers now perhaps only one word, or part of a word. Therefore, dependencies between words are not well captured.

4.1.3 Diacritization on both Levels

By creating a diacritizer that functions on both levels we combine the benefits of both the diacritization on word level and on character level.

The SMT system used in our experiments can take lattices as input. The lattice representation allows edges labeled with words or with characters. Therefore, the decoder is able to switch from word level to character level and back. Due to the phrase count feature in the decoder translations composed from fewer phrases are preferred. This introduces a bias toward using phrases on the word level. However, if a word has not been seen in the training data the system can use the diacritization generated on the character level.

An example for a lattice, which enables the approach on both levels is illustrated in Figure 3.

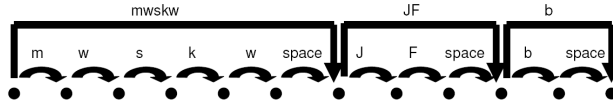


Figure 3: Lattice with Edges from Character to Character and from Word to Word.

4.1.4 Diacritization: Rule-based Translation with Statistical Phrase-based Post-Editing

The approaches described so far do not use linguistic rules for the restoration of diacritics. Recently, there have been a number of studies showing that a statistical machine translation system can successfully be used to post-edit and thereby improve the output of a rule-based translation system (Simard et al., 2007). If appropriate training material is provided, it is possible to train a SMT system to automatically correct systematic errors made by rule-based systems. A similar approach can be used in our case: given the output of a rule-based diacritizer, we can use the statistical approach to perform a post-editing step.

AppTek¹, a company specializing in the development for human language technology, including speech recognition and machine translation, provided diacritized data for this kind of automatic post-editing experiments. The output from their rule-based diacritizer and the corresponding manually diacritized text was made available.

We build a statistical phrase-based system by aligning the output of the rule-based system and the correct vowelized text. This system is then used to operate on test sentences, which are already diacritized with the rule-based diacritizer, as a post-editing step. The benefits are that the rule-based system excludes a large number of possible forms.

4.2 Diacritization as Sequence Labeling Problem

mwskw	X:	m	w	s	k	w
muwsokuw	Y:	u	ε	o	u	ε

The above figure shows an example of an undiacritized word “mwskw” and its diacritization “muwsokuw”. Again, we represent the undiacritized word as a sequence of characters $X = (m, w, s, k, w)$. As presented in the figure, we label each consonant in X with the vowels or diacritics, which should follow that consonant in the diacritized form. Note that a consonant might be followed by more than one vowel or diacritic. Consonants without diacritics are marked with an epsilon (ϵ). Thus the task of diacritization of X is actually finding its sequence $Y = (u, \epsilon, o, u, \epsilon)$. In this way, the problem of diacritization can be considered as the sequence labeling problem.

Conditional random fields (CRF) (Lafferty et al., 2001) have been successful in similar NLP problems such as parts-of-speech tagging (Lafferty et al., 2001) and noun phrase chunking. CFRs can model overlapping features and it is easy to integrate linguistic features such as parts-of-speech. The CRF model estimates the parameters $\bar{\theta}^*$ to maximize the conditional probability of the sequence of tags given the sequence of the consonants in the training data

¹Application Technology Inc., US company

\mathbf{T} as given by the following equation:

$$\bar{\theta}^* = \operatorname{argmax}_{\bar{\theta}} \sum_{(X,Y) \in \mathbf{T}} \log p(Y|X, \bar{\theta}) \quad (1)$$

where $\log p(X|Y, \bar{\theta}) = \sum_i \theta_i f_i(X_q, Y_q)$, f_i is a feature function, and X_q, Y_q are sub-sequences of X, Y respectively.

At the test time, given a sequence of consonants X , and parameter θ^* found at the training time we decode X into the sequence Y^* .

$$Y^* = \operatorname{argmax}_Y p(X|Y, \bar{\theta}^*)$$

In our model, we use the features which are the combinations of lexicals, part-of-speeches of current word, next and previous word. We also use the n neighbor characters as context features. In the experiment section, we will report the results for different context sizes.

5 Data

We work with two data sources: The diacritized LDC's Arabic Treebank data for the translation approaches and the conditional random fields approach as well as data provided by AppTek for running post-editing experiments. In addition to the information on the used data we describe the representation of the Arabic text in Buckwalter Transliteration in this section.

5.1 LDC's Arabic Treebank

The data to train, tune and test the translation- and conditional random fields-based diacritizers are extracted from the LDC's Arabic Treebank of diacritized An Nahar News stories. This data set has been used by several researchers (Maamouri et al., 2006), (Nelken and Shieber, 2005), (Zitouni et al., 2006).

Our sentences are in Buckwalter Transliteration, as pictured in Section 5.3, and do not include any punctuation marks. Since the corpus contains complete vowelization including case endings, the diacritics a, u, i, F, N, K, B and o had to be deleted in order to create the non-vowelized part of the parallel corpus.

All systems except the post-editing system used a training set of 23 k sentences with 613 k words

from this data source. A development set to tune the parameters of the systems and a test set, each consisting of 1,190 sentences with approximately 32k words, were used.

5.2 AppTek Data

Applications Technology (<http://www.apptek.com/>), a company specializing in the development for human language technology, including speech recognition and machine translation, provided diacritized data for automatic post-editing experiments. The output of a rule-based diacritizer and manually generated reference diacritization, consisting of 116 k words in 40 k sentences, were split into a training set of 36 k sentences (104 k words), a development set to tune the system and a test set, each containing 2 k sentences with approximately 6 k words.

As the sentences provided by AppTek are more similar to each other and also rather short, often containing a single word, the error rates with AppTek's data are lower than those obtained with the LDC's Arabic Treebank data.

5.3 Data Representation

For processing purposes the data we work with are in Buckwalter Transliteration (Buckwalter, 2004). Roman character equivalents were chosen to be reasonably mnemonic. From Unicode to Buckwalter Transliteration and back it is a one-to-one mapping without gain or loss of ambiguity.

Figure 4 describes the function of each diacritic, their pronunciation as well as their corresponding character in Buckwalter Transliteration. (We use "B" to represent the "shadda" for processing reasons.)

6 Experiments and Results

6.1 The Translation Systems

For the translation on word and character level we work with a phrase-based translation system as described in (Vogel et al., 2003). As starting points both a **word-based** and a **character-based** baseline SMT system were established and evaluated. **Both systems contain 10-gram Suffix Array Language Models (Zhang, 2006).** The phrase tables contain up to 5-gram entries and appropriate phrase translation









Name		Buckwalter Transliteration	Pronunciation
Short vowels /a/, /u/, /i/			
Fatha		a	/a/
damma		u	/u/
kasra		i	/i/
Double case ending			
fathatayn		F	/an/
dammatayn		N	/un/
kasratayn		K	/in/
Syllabification marks			
shadda		B (normally ~)	consonant doubling vowel
sukuun		o	vowel absence

Figure 4: Buckwalter Transliteration.

probabilities. Later we tested a system with additional lexical scores and 7-gram character entries in the phrase table. In our system which operates on both levels we experimented with longer n-grams in the Suffix Array Language Model as well as with the SRI Language Model Toolkit (Stolcke, 2002).

We evaluated our systems through word error rates and diacritization error rates. Since the error rates are higher at the word endings and since we tested methods to improve especially the final vowelization, we report the error rates regarding as well as ignoring final vowelization.

6.1.1 The Word Level System and the Character Level System

As shown in Table 1 the results of the system on character level are better than those of the word-based system since the word-based system was not able to translate many words.

As the system on character level outperformed the word level system, we tried to enhance the character level system first. Our focus was on the translation model. So far the phrase table only contained phrase translation probabilities. As relative

		word-based	char-based
final_ vow	WER	22.8	21.8
	DER	7.4	4.8
no_final_ vow	WER	9.9	7.4
	DER	4.3	1.8

Table 1: Results of Word-based and Character-based Baseline Systems.

		baseline system	max. phrase length 7	lexical score
final_ vow	WER	21.8	21.6	21.5
	DER	4.8	4.8	4.7
no_final_ vow	WER	7.4	7.5	7.4
	DER	1.8	1.9	1.8

Table 2: Results for the Character-based System using longer phrases and additional Lexical Weights.

frequencies are unreliable for low frequency event we added **so-called lexical scores**. Given a source phrase $f_1 \dots f_J$ and a target phrase $e_1 \dots e_I$, and a word-to-word alignment between source and target words, we calculate:

$$lex(f_1^J|e_1^I, a) = \prod_{j=1}^J \frac{1}{|\{i|(j, i) \in a\}|} \sum_{(j, i) \in a} w(f_j|e_i)$$

As in the case of diacritization the alignment is strictly monotone and one-to-one, the factor in front of the sum reduces to 1.

To generate a phrase table containing both the phrase translation probabilities $\varphi(vow/non_vow)$ and the lexical weights $lex(vow/non_vow)$ we used the Moses Package (Koehn et al., 2007) and GIZA++ (Och and Ney, 2003). By default a phrase table containing up to 7-gram entries is created. In order to have a comparable phrase table to see the improvement resulting from the additional lexical scores we also used a 7-gram phrase table in our baseline system. Table 2 demonstrates that enlarging the maximum number of tokens in a phrase within the phrase table from five to seven characters reduces the word error rate by 0.2% while the lexical weights reduce it by 0.1%.

6.1.2 The System on both Levels

We build an SMT system which operates on word level for known words and on character level for un-

language model		char 5	word 3 SRI	word 4 SRI	word 6 SA
final_ vow	WER	20.1	19.9	20.0	20.0
	DER	4.3	4.3	4.3	4.3
no_final_ vow	WER	6.6	6.8	6.9	6.9
	DER	1.6	1.7	1.7	1.7

Table 3: Results of the Systems on both Levels, using different Language Models (Character and Word Level, different n-gram Lengths, and build with the SRILM Toolkit or the SALM Toolkit).

known words. To have a common representation on the target side for the language model, we converted the target side of the word-based phrase translation table into the character representation. That is to say, while matching on the source side operates on words, the output is segmented into characters. Ideally, the decoder should allow to apply character-level and word-level simultaneously. Unfortunately, the decoder available for the experiments did not provide this functionality. To still have the benefit of larger context from a word-based language model, we applied n-best list rescoring. We generated 1000-best lists and converted from character representation back to word representation. An additional language model score was then calculated. We experimented with the Suffix Array Language Model Toolkit and the SRI Language Model Toolkit. The development data was used to optimize the weights for the combination of the decoder scores and the additional language model score.

After n-best list rescoring the new first-best translation is evaluated. The results in Table 3 show that working on both levels yields a significant improvement in comparison to the systems on character level. A large context in the language model on word level does not perform any better. The word-based 3-gram SRI Language Model scored better than a 4- or 6-gram SRI Language Model and any Suffix Array Language Model with equivalent n-grams.

6.1.3 The Post-Editing System

To test the automatic post-editing approach the output of AppTek’s rule-based diacritizer together with the manually diacritized data was used to build the phrase table and the language model. The phrase

		baseline	post-editing
final_ vow	WER	15.6	13.8
	DER	5.5	4.9
no_final_ vow	WER	10.3	9.3
	DER	3.5	3.2

Table 4: Results of the Post-Editing System, compared to the purely Statistical Baseline System.

table was build using GIZA++ and scripts from the Moses Package. The conditional phrase translation probabilities as well as the lexical weights were used to score the phrase pairs.

Table 4 shows the post-editing results. While the output of the rule-based system gave rather high error rates (not shown in the table) we see acceptable error rates for the purely statistical approach. However, it was outperformed by the combined system, in which the output of AppTek’s rule-based diacritizer was processed by the automatic post-editing system.

6.2 The Diacritization as Sequence Labeling System

Considering our results so far it is evident that the error rates at the word endings are significantly higher than at the word stems. To overcome the limits of the phrase-based translation approach we explored the integration of grammatical information into a statistical approach for the diacritization with the help of conditional random fields. Parts-of-speech were assigned to each word using the Stanford Parts-of-Speech Tagger (Toutanova and Manning, 2000). In addition to the non-vowelized input, the output sequence depended on previous and following source word as well as on the parts-of-speech for the previous, current, and the following word. The system uses the CRF++ toolkit (Kudo, 2007) to train and test our model. However, due to memory constraints we could only use the most effective features. To incorporate wider context we had to reduce the amount the training data to 75%. The results are shown in Table 5. We see that using wider context, which is counted in terms of characters, is crucial in improving the performance of the system. Unfortunately, we did not observe that the additional information, e.g. word identities and parts-of-speech, could reduce the diacritization errors at the word endings.

		data	100%	75%				
		context	4	4	6	8	10	12
final_vow	WER	22.8	24.1	22.6	22.2	22.0	21.9	
	DER	5.1	5.4	4.9	4.8	4.7	4.7	
no_final_vow	WER	9.4	10.0	8.5	8.3	8.3	8.4	
	DER	2.2	2.4	2.0	1.9	1.9	1.9	

Table 5: Results of the CRF Approach for different amounts of the data and different context.

7 Conclusion and Future Work

We presented two approaches to diacritize Arabic text. In the first approach diacritization was regarded as a translation problem, and we used techniques from phrase-based translation to "translate" from undiacritized Arabic into diacritized Arabic. We experimented with word-level and character-level processing, and also with a combined system, where the input is represented on both levels in the form of input lattices. Benefits were gained by operating both on word and character level. As was expected, longer n-grams in the phrase table gave better results, as did the addition of lexical scores.

In the second approach we treated diacritization as a sequence labeling problem. Using CRFs allowed us to integrate additional features, like parts-of-speech. Due to memory limitations this approach could not be fully explored, i.e. we had to trade-off between training corpus size and number of features. We expect that with more data and additional features this approach will perform on the same level, or even better than the translation approach.

We also experimented with automatically post-editing the output of a rule-based system. The combined system outperformed both the rule-based and the purely statistical system.

Why is this?

n-gram error due to asymmetries at the end of a word??

A major problem in diacritization are the errors in the word endings. For example, in our phrase-based diacritization systems we detected that the word ending "pi" (ta marbouta with kasra) occurs almost 2% and "i" (kasra) even more than 5.5% more frequently in our hypothesis than in the reference or in the training data. The word endings depend on the grammatical role of the word within the sentence. This leads to long-range dependencies, which are not well captured by the current models. Future experiments will explore, which features are useful to reduce these error.

Another point of interest would be to find out whether the integration of the proposed diacritization features enhances the Arabic-English or English-Arabic translation systems.

Acknowledgments

We would like to thank Hassan Sawaf, AppTek, for providing the data used in our post-editing experiments. We would also like to thank Anna Owen for her useful comments during the writing of this paper. This work was in part supported by DARPA under HR0011-06-2-0001 (GALE).

References

- T. Buckwalter. 2004. Arabic Morphological Analyzer version 2.0. LDC2004L02.
- Mona Diab, Mahmoud Ghoneim, and Nizar Habash. 2007. Arabic Diacritization in the Context of Statistical Machine translation. In *Proceedings of the MT-Summit*, Copenhagen, Denmark.
- Yousif A. El-Imam. 2004. Phonetization of Arabic: Rules and Algorithms. *Computer Speech and Language*, 18(4).
- Tarek A. El-Sadany and Mohamed A. Hashish. 1989. An Arabic Morphological System. *IBM Systems Journal*, 28(4).
- Ossama Emam and Volker Fischer. 2005. Hierarchical Approach for the Statistical Vowelization of Arabic Text. Technical report, IBM Corporation Intellectual Property Law, Austin, TX, US.
- Ya'akov Gal. 2002. An HMM Approach to Vowel Restoration in Arabic and Hebrew. In *Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages*, Philadelphia, PA, USA.
- Nizar Habash and Owen Rambow. 2007. Arabic Diacritization through Full Morphological Tagging. In *Proceedings of NAACL/HLT 2007. Companion Volume, Short Papers*, Rochester, New York, April.
- Philipp Koehn, Hieu Hoang, Alexandra Birch and Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard

- Zens, Chris Dyer, Ondrej Bojar and Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Annual Meeting of ACL, demonstration session*, Prag, Czech Republic, June.
- John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th ICML*.
- Mohamed Maamouri, Ann Bies, and Seth Kulick. 2006. Diacritization: A Challenge to Arabic Treebank Annotation and Parsing. In *Proceedings of the British Computer Society Arabic NLP/MT Conference*.
- Rada Mihalcea. 2002. Diacritics Restoration: Learning from Letters versus Learning from Words. In *Proceedings of the 3rd CICLing*, London, UK.
- Husni Al-Muhtaseb Mustafa Elshafei and Mansour Alghamdi. 2006. Statistical Methods for Automatic Diacritization of Arabic Text. In *Proceedings of the Saudi 18th National Computer Conference (NCC18)*, Riyadh, Saudi Arabia, March.
- Rani Nelken and Stuart M. Shieber. 2005. Arabic Diacritization Using Weighted Finite-State Transducers. In *Proceedings of the ACL 2005 Workshop On Computational Approaches To Semitic Languages*, Ann Arbor, Michigan, USA.
- Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th ACL*, Philadelphia.
- Michel Simard, Nicola Ueffing, Pierre Isabelle, and Roland Kuhn. 2007. Rule-Based Translation with Statistical Phrase-Based Post-Editing. In *Proceedings of the Second Workshop on Statistical Machine Translation*, Prague, Czech Republic, June.
- Andreas Stolcke. 2002. SRILM – an Extensible Language Modeling Toolkit. In *International Conference on Spoken Language Processing*, Denver, USA.
- Kristina Toutanova and Christopher D. Manning. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, Hong Kong.
- Dan Tufis and Adrian Chitu. 1999. Automatic Diacritics Insertion in Romanian Texts. In *Proceedings of COMPLEX'99 International Conference on Computational Lexicography*, Pecs, Hungary, June.
- Dimitra Vergyri and Katrin Kirchhoff. 2004. Automatic Diacritization of Arabic for Acoustic Modeling in Speech Recognition. In *COLING 2004 Computational Approaches to Arabic Script-based Languages*, Geneva, Switzerland, August 28th.
- Stephan Vogel, Ying Zhang, Fei Huang, Alicia Tribble, Ashish Venugopal, Bing Zhao, and Alex Waibel. 2003. The CMU Statistical Machine Translation System. In *Proceedings of MT-Summit IX*, New Orleans, Louisiana, USA, September.
- Ying Zhang. 2006. SALM: Suffix Array and its Applications in Empirical Language Processing. Technical Report CMU-LTI-06-010, LTI, Carnegie Mellon University, Pittsburgh PA, USA.
- Imed Zitouni, Jeffrey S. Sorensen, and Ruhi Sarikaya. 2006. Maximum Entropy Based Restoration of Arabic Diacritics. In *Proceedings of the 44th Annual Meeting of the ACL*, Sydney, Australia, July.