Diacritization for Real-World Arabic Texts

Emad Mohamed
Indiana University
emohamed@indiana.edu

Sandra Kübler
Indiana University
skuebler@indiana.edu

Abstract

For Arabic, diacritizing written text is important for many NLP tasks. In the work presented here, we investigate the quality of a diacritization approach, with a high success rate for treebank data but with a more limited success on real-world data. One of the problems we encountered is the non-standard use of the hamza diacritic, which leads to a decrease in diacritization accuracy. If an automatic hamza restoration module precedes diacritization, the results improve from a word error rate of 9.20% to 7.38% in treebank data, and from 7.96% to 5.93% on selected real-world texts. This shows clearly that hamza restoration is a necessary step for improving diacritization quality for Arabic real-world texts.

Keywords

Arabic diacritization, memory-based learning

1 Introduction

The problem of diacritization, or vocalization, is essential to many tasks in Arabic NLP. Arabic is generally written without the short vowels, which means that one written form can have several pronunciations, each pronunciation carrying its own meaning(s). The word form 'mskn' is an example for a highly ambiguous word. Its possible pronunciations include 'maskan' (home), 'musakkin' (analgesic), 'masakn' (they-*fem.* have held), or 'musikn' (they-*fem.* have been held). The importance of diacritization becomes clear when we look at how Google Translate renders 'A$tryt Almskn mn AlSydlyp' (I bought a pain killer from the pharmacy): as 'I bought the home from the pharmacy'. This error would not occur if the input to the translation system were diacritized in a first step before the actual translation process. However, diacritization is far from trivial: the example above shows that the diacritized forms of a single undiacritized word differ in their parts of speech (POS) as well as in their meaning. This shows that to a certain degree, diacritization performs implicit POS tagging and word sense disambiguation. It also shows very clearly that word forms cannot be diacritized in isolation; the task is heavily dependent on the context of the word.

The goal of the work reported here is the creation of a diacritization system that works reliably on real-world data taken from websites. The first step towards this problem is a system that reaches a high accuracy in an *in-vitro* evaluation, i.e. on data from the same treebank on which it was trained. After we obtain a reliable system for the treebank data, we will test it to determine how well it works in an *in-vivo* evaluation, i.e. on real-world data. More importantly, in the second step, we will investigate how the system needs to be modified in order to produce reliable diacritizations in the in-vivo situation.

In the first step, we concentrate on investigating how important different types of context are for diacritization. We follow Zitouni et al. [16] in defining diacritization as a classification problem in which we decide for each character in the undiacritized word whether it is followed by a short vowel. Additionally, the classification task includes the shadda and sokoon (lack of a vowel). The shadda is consonant gemination and is usually found in combination with another vowel, thus resulting in 3 classes, $\tilde{}a, \tilde{}i, \tilde{}u$. The shadda plays an important role in the interpretation of Arabic words because it is used, inter alia, to discriminate between the base form of a verb and its causative form: *kataba* (to write), *kat˜aba* (to make write). At present, we ignore case endings, mood endings, and nunation (syntactic indefiniteness marker for a noun or adjective). In this setting, we also ignore the hamza, the glottal stop.

Most experiments are performed on treebank data, which is preprocessed and standardized to a very high degree. When such a model is used for naturally occurring data, such as newspaper texts published on the world-wide web, the results are considerably lower than the results published by Zitouni et al. [16], for example.

For the second set of experiments, our intention is to investigate how such a treebank trained diacritization system performs on real-world data. For this experiment, we use the diacritization approach that proved optimal on the treebank data (from the Penn Arabic Treebank (ATB) [1]) and use it to diacritize articles from the Agence France Press (AFP) Arabic website. Although, data from AFP were included in the training set, so that there are no cross-genre problems, our first results were disappointing. Examination of the data shows that one major difference between our training data from the ATB and the real-world test data is that the latter collapses the different forms of 'hamza', the glottal stop, into the long vowel alef, a common, but non-standard, practice. In the real-world texts, however, practices vary considerably in terms of whether the hamza is used carefully or carelessly, or even whether it is used at all. Some texts, like those of the AFP news agency do not use hamzas at all. In other texts, such as the Egyptian Al-Ahram [1], they are used correctly. From these findings, we conclude that a hamza restoration program is

[1] www.ahram.org.eg

necessary for the treatment of such texts.

In further experiments, we examine the different options of hamza restoration or normalization in search of the optimal settings of diacritization. For this purpose, we conduct two kinds of experiments: 1) in-vitro experiments, in which we test the different settings in a controlled setting on data from the Arabic Treebank, and 2) in-vivo experiments, in which we use the best settings from step 1) for diacritizing real-world data.

2 Related work

The first approaches to the diacritization of Arabic define the problem word-based, i.e. the task is to determine for each word the complete diacritized form. Gal [10] uses a bigram Hidden Markov Model for diacritizing the Qur'an and achieves a word error rate (WER) of 14%. His error analysis shows that the errors result mostly from unknown words. Kirchhoff et al. [12] design a diacritization module for use in a speech recognition system. Their system uses a unigram model extended by a heuristic for unknown words, which retrieves the most similar unlexicalized word and then applies edit distance operations to turn it into the unknown word. They reach a WER (for diacritization) of 16.5% on conversational Arabic. Nelken and Shieber [14] tackle the problem with weighted finite-state transducers. For known words, morphological units are used for retrieving the diacritization while unknown words are diacritized based on the sequence of characters. They reach a WER of 12.8%. Zitouni et al. [16] use a maximum entropy model in combination with a character based classification. Their features are based on single characters of the focus word, morphological segments, and POS tags. They reach a WER of 17.9%. Habash and Rambow [11] perform a full diacritization including case endings and nunation. They use the Buckwalter analyzer [3] to obtain all possible morphological analyses, including all diacritics. Then they train individual classifiers to disambiguate between these analyses. Residual ambiguity is resolved via an n-gram language model. Habash and Rambow reach a WER of 14.9% on the test set of Zitouni et al. [16]. Shaalan et al. [15] compare a lexicon-based approach with an approach using word bigram statistics and a Support Vector Machine (SVM) classifier. The SVM approach uses features from automatic segmentation, POS tagging, and chunk parsing. Shaalan et al. show that the best results for diacritization are reached by combining all three approaches. Unfortunately, they use the Ummah section of the treebank for training and testing, which can be shown to give better results than the An Nahar News section that Zitouni et al. and Habash and Rambow use (see Section 3.3). To our knowledge, Shaalan et al. are the first to include case endings as features in the task. however, a comparison of the SVM approach with and without case endings shows that their inclusion results in a considerable decrease in performace: The WER increases from 16.26% to 69.94%.

A comparison of the different approaches shows that the definition of diacritization as inserting vowels between characters results in the best WER. However, these studies leave the lexical context of words for the most part unexplored. In the present study, we will investigate this area of research. The studies also concentrate on treebank data, which means that it is unclear how well they work on real-world data.

We are not aware of any automatic approaches to hamza restoration. For example, Diab et al. [9] do not consider the hamza in their diacritization since “most Arabic encodings do not count the hamza a diacritic, but rather a part of the letter”. The relaxed attitude towards the hamza in the Arabic orthography is part of what Buckwalter [4] calls “suboptimal orthography”.

3 The baseline system

3.1 Data

For the baseline system, we use the Penn Arabic Treebank (ATB) [1] as the data source. The treebank is encoded in Buckwalter transliteration [3] and is available in a diacritized and an undiacritized version. Based on the treebank, we performed three different experiments: 1) In order to test the limits of the approach, we decided to use a large and varied data set, with 10-fold cross-validation (CV). 2) To ensure that our results can be compared with the results of Zitouni et al. [16], we perform a second experiment on their data set. 3) Based on the results from the first experiment, we selected one fold of this experiment as test set and the other 9 folds as training set for the third experiment. This data set will also be used in the experiments concerning the hamza restoration. In order to keep consistency with the real-world test set from the internet in this second series of experiments, we chose the set including AFP data. An earlier version of the first two experiments described here was published in [13].

For the first experiment, we extract 170 000 words from the AFP section (part 1 version 2.0) and approximately 160 000 words from the Ummah section (part 2 version 2.0), in a 10-fold CV setting. This experiment is intended to determine the optimal parameter settings for the machine learner and the optimal context used for diacritization. For the second experiment, which serves as comparison to Zitouni et al., the data are taken from part 3, version 1.0. This data set contains news items from the An Nahar News text, it is split into a training set of approximately 288 000 words and a devtest set of approximately 52 000 words. For the third experiment, we chose a subset of the first experiment, part 1, version 2.0, in order to use data that originates from the same source, AFP, as the real-world test files. In this setting, the 90% from the beginning of the data set serve as training data and the tail 10% are used for testing.

As mentioned previously, we define diacritization as a classification problem: For each character in the focus word, the learner needs to decide whether the character is followed by a short vowel and what the short vowel is. We will call this character the focus character. The task also involves the restoration of the shadda (gemination).

The features used for determining the short vowel following the focus character consist of the focus character itself (c), its local context in terms of neighboring

w_{-5}	w_{-4}	w_{-3}	w_{-2}	w_{-1}	c_{-5}	c_{-4}	c_{-3}	c_{-2}	c_{-1}	c	c_1	c_2	c_3	c_4	c_5	w_1	w_2	w_3	w_4	w_5	v
kl	$y”	tgyr	fy	HyAp	_	_	_	_	_	A	l	m	t	$	r	styfn	knt	EndmA	Evrt	Ely	_
kl	$y”	tgyr	fy	HyAp	_	_	_	_	A	l	m	t	$	r	d	styfn	knt	EndmA	Evrt	Ely	_
kl	$y”	tgyr	fy	HyAp	_	_	_	A	l	m	t	$	r	d	_	styfn	knt	EndmA	Evrt	Ely	u
kl	$y”	tgyr	fy	HyAp	_	_	A	l	m	t	$	r	d	_	_	styfn	knt	EndmA	Evrt	Ely	a
kl	$y”	tgyr	fy	HyAp	_	A	l	m	t	$	r	d	_	_	_	styfn	knt	EndmA	Evrt	Ely	a
kl	$y”	tgyr	fy	HyAp	A	l	m	t	$	r	d	_	_	_	_	styfn	knt	EndmA	Evrt	Ely	ĩ

Fig. 1: *The word 'Almt$rd' represented with one instance per word; the class represents the vowel to be inserted after the character.*

characters within the focus word, and a more global context of neighboring words. For the local context, 5 characters to the left ($c_{-5} \dots c_{-1}$) and 5 characters to the right ($c_1 \dots c_5$) are used; for the lexical context, 5 words to the left ($w_{-5} \dots w_{-1}$), and 5 words to the right ($w_1 \dots w_5$). The last value in the vector (v) provides the correct classification, i.e. the short vowel to be inserted after c, or - in cases where no vowel is inserted in that position. The instances for the Arabic word 'Almt$rd', for example, are shown in Figure 1. The last instance contains an example of the shadda, represented by the tilde sign.

3.2 Methods

For classification, we use a memory-based learner, TiMBL [7]. Memory-based learning is a lazy-learning paradigm, which assumes that learning does not consist of abstraction of the training instances into rules or probabilities. Instead, the learner uses the training instances directly. As a consequence, training consists in storing the instances in an instance base, and classification finds the *k* nearest neighbors in the instance base and chooses their most frequent class as the class for the new instance. Memory-based learning has been proven to have a suitable bias for many NLP problems [5, 6]. One of the reasons for this success is that natural language exhibits a high percentage of subregularities or irregularities, which cannot be distinguished from noise. Eager learning paradigms smooth over all these cases while memory-based learning still has access to the original instance. Thus, if a new instance is similar enough to one of these irregular instances, it can be correctly classified as such.

Memory-based learning was chosen for two reasons: First, this approach weights features based on information gain or gain ratio [7], thus giving some indication of the most and the least important features. Additionally, it is a paradigm that is capable of handling symbolic features with a high number of different feature values. This allows us to use complete context words as features.

Parameter settings for TiMBL need to be determined first. The best results are obtained for all experiments with the IB1 algorithm with similarity computed as weighted overlap, i.e. with a standard city block metric as distance measure. Relevance weights are computed with gain ratio, and the number of *k* nearest neighbors (or in TiMBL's case, nearest distances) is set to 1. The latter setting is noteworthy in that it signals that only the closest training examples provide reliable information for classifying a character. Normally, higher values of *k* are beneficial since they provide a certain smoothing factor. A more detailed investigation of different options in feature settings, as well as optimal training set sizes can be found in [13].

For the experiments to determine the effect of context on diacritization accuracy, we use 10-fold cross validation (CV). For the experiments with the data set from Zitouni et al. and for the AFP set, we use a designated test set. For the 10-fold CV experiments, we do not build the folds randomly but rather sequentially, thus ensuring that a single fold contains consecutive articles. This may result in folds which cover different topics from the other folds. We are convinced that this approach is closer to real-world situations since we cannot be sure that the training data will be from the same time period and thus cover the same topics. However, we make sure that all instances of a word are put in the same fold.

For evaluation, we calculate the error rate based on characters (CER) and based on words (WER). A decision of the classifier is considered correct if both the vowel to be inserted and the shadda (if present) are correct. Since previous work has been evaluated on all words in the texts, including punctuation and other non-diacritized words such as numbers, we will present these results to allow a comparison to previous work. However, we believe that words that are never diacritized (such as numbers or punctuation signs) should not be considered in the evaluation. For this reason, we also provide results where such words were present in the classification, but were excluded in the evaluation.

3.3 Baseline results for treebank data

Before we start with the evaluation proper, we need to establish the level of difficulty of the task. One measure for difficulty is the average number diacritizations per word. A closer look at the large data set used for the fist experiment shows that a word on average has only 1.67 diacritizations. This figure is considerably lower than the average in normal texts. Debili et al. [8] found that on average, each undiacritized word type has 2.9 diacritized versions, and there is an average of 11.6 diacritized versions per word token in a text. We assume that the difference is a consequence of the different text genres.

The results of our experiments with regard to different contexts on the large data set are shown in the first part of Table 1. The first experiment uses only a character context of 5 characters to each side of the focus character but ignores the context words, i.e. the features from c_{-5} to c_5 in Figure 1 are used. The next experiment uses the lexical context to the left of the

	With punct.		W/o punct.	
	CER	WER	CER	WER
Character context	2.22	6.64	6.20	14.40
Left word context	2.26	7.06	2.33	7.57
Word context	2.35	6.86	2.64	9.91
Zitouni data set	5.70	17.50	5.70	20.49
Zitouni et al.	5.5	18.0	n/a	n/a
AFP data set	2.53	7.39	2.83	13.65

Table 1: *The results of the diacritization experiments on different parts of the ATB*

focus word in addition to the character context but ignores the context words on the right, i.e. the features from w_{-5} to c_5 are used. Finally, the last experiment uses all features shown in Figure 1, i.e. it uses the character context as well as the lexical features to the left and to the right of the focus word. The results show that in the evaluation including punctuation and words that are never diacritized, both the CER and the WER are slightly worse when context is added. However, in the evaluation where punctuation and non-diacritized words are excluded, we find a considerable improvement when words from the left context are added. Adding the right context, in contrast, has a negative effect on both error rates. From these results, we conclude that the left context is more important (which is also corroborated by the weights assigned to the word features). And we assume that the inclusion of words from the right context lead to data sparseness problems. For this reason, all the remaining experiments are carried out with the character context plus 5 words on the left of the focus word.

The experiment on the data set by Zitouni et al. shows that there is considerable variability in the data sets of the Penn Arabic Treebank. Here, the error rates are more than twice as high as on the first, larger data set. A comparison of our approach on this data set shows that the results are comparable to those by Zitouni et al., which are listed in the row below ours. This is notable since our system does not have access to any linguistic preprocessing. The system by Zitouni et al., in contrast, has access to morphological segments and POS tags.

The third experiment is performed on a data set that is a subset of the first set, namely the AFP section from part 1, version 2.0. Here, the error rates are slightly worse than for the first set. This is the data set that will be used for the experiments in the following section.

4 Beyond treebank data

In this section, we will investigate the question how the system presented in the previous section needs to be modified in order to be usable for real-world texts, which is not as clean as the treebank data. A first informal evaluation of some real-world texts shows a surprisingly high number of errors in diacritization. A closer look at these errors shows that many of them involved a non-standard use of the hamza. For this reason, we decided to carry out a series of experiments to determine a usable strategy to improve the results.

As mentioned before for all experiments reported in this section, we use the Arabic Treebank part 1, version 2.0, where the 90% from the beginning serve as training data and the tail 10% are used for testing. This section is based on AFP news, the same genre as the real-world texts we use for evaluation. This ensures that the effects we will see in the results are due to our modifications and not due to out-of-domain phenomena. All the experiments are conducted with the optimal parameter and feature settings as determined in the first experiment described in Section 3.

We conduct four experiments to test the effect of hamza presence or absence on diacritization as detailed below:

1. **Pure treebank**: In order to determine the upper bound, we train and test the diacritization system on the treebank in its original form.

2. **Normalized treebank**: For this experiment, we normalized all hamzas both in the training and the test set. To this end, we replaced all hamzas with the alef (A in Buckwalter transliteration [2]). The intuition behind this experiment is to test a simple normalization of non-standard uses of the hamza, which might solve the problems with non-conventional diacritization. If this experiment reaches the same results as the upper bound from experiment 1, hamzas are not important for diacritization.

3. **Hamza-free test set**: Here, we train on the original training set (with all hamzas) and test on the test set with all hamzas removed. This experiment is intended to show whether the variation in hamzas between the training and test sets has any bearing on diacritization. If there are no differences, then we can reach good results on real-world texts by simply removing all hamzas.

4. **Hamza-free training set**: This experiment is the exact opposite of experiment 3. Here, we train on the training set without hamzas and test on the original test set (containing all hamzas).

4.1 Results

The results of the experiments presented above are shown in Table 2. For the upper bound experiment, in which the original treebank data are used for both training and testing, the memory-based diacritization system reached an overall CER of 2.53% and a WER of 7.39% when punctuation is included in the evaluation and a CER of 2.83 and WER of 13.65 when punctuation is excluded. A word is considered ill-diacritized if any of its vowels is wrong. These results are in the range of other published results although they are slightly worse than the best results of the first experiment described in Section 3. The reason for this can be found in the choice of the training set, which is here restricted to one part of the treebank. Additionally, in the present experiments, we do not use 10-fold CV but a single fold for testing.

When we remove all hamzas from the training and test set (normalized treebank), the diacritization system reaches a CER of 3.08% and a WER of 9.20%

	With punct.		W/o punct.	
	CER	WER	CER	WER
Pure treebank (AFP set)	2.53	7.39	2.83	13.65
Normalized treebank	3.08	9.20	3.43	17.02
Hamza-free test set	6.51	18.11	7.26	33.48
Hamza-free training set	3.53	10.48	3.94	19.38
Hamza-restored	2.54	7.38	2.83	13.66

Table 2: *The results of the in-vitro experiments with the AFP part of the ATB*

including punctuation. When punctuation is not included in evaluation, the WER increases by approximately 3.5 percent points. These results show that removing the hamzas from both the training and testing sets decreases the accuracy of diacritization at both the character and word level so that we have to conclude that the hamza is important for diacritization and should not be normalized. The normalization option may also be dis-preferred for lingustic reasons as it collapses different characters into one, and these characters may at times distiniguish minimal pairs. For example, the words vAr and v>r (in Buckwalter transliteration) mean ”to revolt” and ”revenge” respectively. The difference is not merely allophonic. Similar meaning-distinguishing hamzas include the pairs: fAr (fugitive) and f>r (mouse), <rm (a city name) and Arm (throw, imperative), mAl (money) and m|l (destination).

The experiments in which either the training set or the test set contained hamzas show the worst results: For the hamza-free training set, the WER reached 10.48% including punctuation and 19.38% when punctuation is excluded. For the experiment with the hamza-free test set, both the CER (6.51% with punctuation and 7.26% without punctuation) and the WER (18.11% with punctuation and a staggering 33.48% without punctuation) are the highest for all experiments. This shows that the standard approach for diacritizing real-world data, i.e. training on the ATB treebank and testing on texts that may not contain hamzas is the worst possible setting.

5 Creating a hamza restoration module

The experiments described above lead to the conclusion that we need a hamza module that can take the raw text and restore the hamzas if necessary, before the text is passed to the diacritization system. However, such an approach is only feasible if the hamza restoration module reaches a high accuracy. Otherwise, incorrectly placed hamzas may even further harm diacritization results.

In order to test the usefulness of hamza restoration, we designed a hamza restoration module using the same training and testing sets as the AFP experiment in Section 3. The module uses TiMBL with the following settings, which proved to be optimal in a non-exhaustive search: IB1, with similarity computed as weighted overlap, relevance weights computed with gain ratio, and the number of k nearest neighbors set to ???. Similar to diacritization, hamza restoration is treated as a classification problem, in which the classifier assigns one of the four classes A, |, <, > (alef and the 3 hamza forms) to each potential hamza location, whether it occurs word-initially, word-medially, or word-finally. In order to remove non-standard diacritization, the module removes all hamza forms first, including the alef, and then re-assigns a hamza form or an alef to each location. We use the focus hamza location and a context of the previous and following 5 characters as features. The hamza restoration accuracy we obtain using this module is 98.09%.

Given a hamza restoration module with sufficient accuracy, we need to test the effect of hamza restoration on diacritization. For this experiment, we use the normalized treebank version from Section 4 (with all hamzas replaced by alephs). This version is passed to the hamza restoration module, and subsequently to the diacritization system, now with the original training set, and the hamza-restored file as the test set. We obtain considerably improved results: a CER of 2.54% and a WER of 7.38% including punctuation and a CER of 2.83% and a WER of 13.66% without punctuation (cf. experiment *hamza-restored* in Table 2). This means, our results are nearly identical to the results when both training and test data contain perfect hamza information, at least on treebank data.

In the next section, we describe the in-vivo experiments, using our hamza restoration module for diacritizing text obtained from the AFP website. This experiment will show whether the method presented here can also improve results for real-world texts.

6 Diacritizing real-world texts

The test corpus we selected for this experiment consists of four news stories from the AFP Arabic website published on December 2, 2008. The resulting test set consists of 1 332 words. None of these texts have any hamza represented in the texts.

We conduct two experiments on these four files: The first one uses the texts in the original form as taken from the website as input for the diacritization system. The second experiment passes the texts through the hamza restoration module before sending them to the diacritization system. We use the same parameter settings and features as for the the experiments in Section 4.

The resulting diacritized versions were given to two independent raters for evaluation. Both raters are native speakers of Arabic and graduate students; one of them teaches undergraduate Arabic courses. Each rater was familiarized with the task and was given

Text	No. of words	No hamza restoration		With hamza restoration	
		No. incorrect words	WER	No. incorrect words	WER
1	272	20	7.35	11	4.04
2	556	51	9.17	37	6.65
3	252	19	7.54	20	7.94
4	252	16	6.35	11	4.37
Total	1332	106	7.96	79	5.93

Table 3: *Diacritization of real-world texts with and without hamza restoration*

instructions to correct all wrongly diacritized words. Then the two texts were compared, and the first author, a native speaker of Arabic, served as arbitrator.

It is worth noting that rater A found 3 errors that require a very deep knowledge at the discourse level. The news item talks about prisoners. The number of the prisoners is declared to be two towards the end of the story. This requires some change in the diacritization of the word "prisoners" to reflect the dual. Even an expert Arabicist would not find the correct diacritization for these three consecutive words (1 noun and 2 adjectives) without reading the whole story first. This indicates that diacritization is sometimes challenging even for native speakers. These three errors are not included in the calculation.

The results of the experiments on real-world texts are shown in Table 3. The comparison of the results with and without hamza restoration before diacritization show that using hamza restoration reduces the word error rate by 2 percent points. This corresponds to an error reduction of 25.5%.

The results for diacritization with hamza restoration are clearly better than the results for the same approach on the treebank data (experiment hamza-restored in Table 2). We assume that the high results are due to the selection of a small number of short news stories in order not to overtax the human raters. These texts are in general easier to diacritize than the treebank texts.

7 Conclusion and Future Work

We presented a system for diacritization trained on the Penn Arabic Treebank. After parameter and feature optimization, the system reaches competitive error rates as compared to systems such as the one by Zitouni et al. [16] although it does not use any linguistic preprocessing. In a next step, we investigated how well such a treebank trained system performs on texts that differ in whether the hamza is present or not. The results show that inconsistencies between training and test set in this respect lead to a higher number of errors in diacritization. In a last experiment, we tested our approach on real-world data taken from the AFP website, with parallel improvements for the version with the integrated hamza restoration module. The experiments here show clearly that hamza restoration is a necessary step for improving diacritization quality for Arabic real-world texts. Treebanks tend to be more perfect than naturally occurring texts in terms of adherence to spelling conventions, especially with regard to diacritization. This makes them suboptimal for training modules that are intended for real-world texts.

For the future, we are planning to investigate a post-processing module for hamza restoration, which checks whether the suggested hamza belongs to the confusion set allowed in a certain context. We are also planning to integrate linguistic information such as segmentation and POS tagging into the diacritization module. Additionally, we are planning to investigate how diacritization affects POS tagging. We assume that reliable POS tags will improve diacritization while at the same time, reliable diacritics will improve POS tagging, thus leading to a circular optimization problem.

References

[1] A. Bies and M. Maamouri. Penn Arabic Treebank guidelines. Technical report, LDC, University of Pennsylvania, 2003.

[2] T. Buckwalter. Arabic morphological analyzer version 1.0. Linguistic Data Consortium, 2002.

[3] T. Buckwalter. Arabic morphological analyzer version 2.0. Linguistic Data Consortium, 2004.

[4] T. Buckwalter. Issues in Arabic morphological analysis. In A. Soudi, A. van den Bosch, and G. Neumann, editors, *Arabic Computational Morphology: Knowledge-Based and Empirical Methods*. Springer Verlag, 2007.

[5] W. Daelemans and A. van den Bosch. *Memory Based Language Processing*. Cambridge University Press, 2005.

[6] W. Daelemans, A. van den Bosch, and J. Zavrel. Forgetting exceptions is harmful in language learning. *Machine Learning*, 34:11–43, 1999. Special Issue on Natural Language Learning.

[7] W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. TiMBL: Tilburg memory based learner – version 6.1 – reference guide. Technical Report ILK 07-07, Induction of Linguistic Knowledge, Computational Linguistics, Tilburg University, 2007.

[8] F. Debili, H. Achour, and E. Souissi. De l'etiquetage grammatical a la voyellation automatique de l'arabe. Technical report, Correspondances de l'Institut de Recherche sur le Maghreb Contemporain, 2002.

[9] M. Diab, M. Ghoneim, and N. Habash. Arabic diacritization in the context of statistical machine translation. In *Proceedings of the Machine Translation Summit (MT-Summit)*, Copenhagen, Denmark, 2007.

[10] Y. Gal. 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, 2002.

[11] N. Habash and O. Rambow. Arabic diacritization through full morphological tagging. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Rochester, NY, 2007.

[12] 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 speech recognition models for Arabic - final report of the JHU summer workshop. Technical report, Johns Hopkins University, 2002.

[13] S. Kübler and E. Mohamed. Memory-based vocalization of Arabic. In *Proceedings of the LREC Workshop on HLT and NLP within the Arabic World*, Marrakech, Morocco, 2008.

[14] R. Nelken and S. Shieber. Arabic diacritization using weighed finite-state transducers. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Language*, Ann Arbor, MI, 2005.

[15] K. Shaalan, H. Abo Bakr, and I. Ziedan. A hybrid approach for building Arabic diacritizer. In *Proceedings of the EACL Workshop on Computational Approaches to Semitic Languages*, Athens, Greece, 2009.

[16] I. Zitouni, J. Sorensen, and R. Sarikaya. Maximum entropy based restoration of Arabic diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, COLING-ACL-2006*, Sydney, Australia, 2006.