Arabic Morphological Tagging, Diacritization, and Lemmatization Using Lexeme Models and Feature Ranking

Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin
Center for Computational Learning Systems
Columbia University
New York, NY 10115 USA
{ryanr,rambow,habash,mdiab,rudin}@ccls.columbia.edu,

Abstract

We investigate the tasks of general morphological tagging, diacritization, and lemmatization for Arabic. We show that for all tasks we consider, both modeling the lexeme explicitly, and retuning the weights of individual classifiers for the specific task, improve the performance.

1 Previous Work

Arabic is a morphologically rich language: in our training corpus of about 288,000 words we find 3279 distinct morphological tags, with up to 100,000 possible tags.[1] Because of the large number of tags, it is clear that morphological tagging cannot be construed as a simple classification task. Hajič (2000) is the first to use a dictionary as a source of possible morphological analyses (and hence tags) for an inflected word form. He redefines the tagging task as a choice among the tags proposed by the dictionary, using a log-linear model trained on specific ambiguity classes for individual morphological features. Hajič et al. (2005) implement the approach of Hajič (2000) for Arabic. In previous work, we follow the same approach (Habash and Rambow, 2005), using SVM-classifiers for individual morphological features and a simple combining scheme for choosing among competing analyses proposed by the dictionary. Since the dictionary we use, BAMA (Buckwalter, 2004), also includes diacritics (orthographic marks not usually written), we extend this approach to the diacritization task in (Habash and Rambow, 2007). The work presented in this paper differs from this previous work in that (a) we introduce a new task for Arabic, namely **lemmatization**; (b) we use an explicit modeling of lexemes as a component in all tasks discussed in this paper (morphological tagging, diacritization, and lemmatization); and (c) we tune the weights of the feature classifiers on a tuning corpus (different tuning for different tasks).

[1]This work was funded under the DARPA GALE, program, contract HR0011-06-C-0023. We thank several anonymous reviewers for helpful comments. A longer version of this paper is available as a technical report.

2 Morphological Disambiguation Tasks for Arabic

We define the task of **morphological tagging** as choosing an inflectional morphological tag (in this paper, the term "morphological tagging" never refers to derivational morphology). The morphology of an Arabic word can be described by the 14 (nearly) orthogonal features shown in Figure 1. For different tasks, different subsets may be useful: for example, when translating into a language without case, we may want to omit the case feature. For the experiments we discuss in this paper, we investigate three variants of the morphological tagging tasks: **MorphPOS** (determining the feature POS, which is the core part-of-speech – verb, noun, adjective, etc.); **MorphPart** (determining the set of the first ten basic morphological features listed in Figure 1); and **MorphAll** (determining the full inflectional morphological tag, i.e., all 14 features).

The task of **diacritization** involves adding diacritics (short vowels, gemination marker shadda, and indefiniteness marker nunation) to the standard written form. We have two variants of the diacritization

Proceedings of ACL-08: HLT, Short Papers (Companion Volume), pages 117–120,
Columbus, Ohio, USA, June 2008.

Feature name	Explanation
POS	Simple part-of-speech
CNJ	Presence of a conjunction clitic
PRT	Presence of a particle clitic
PRO	Presence of a pronominal clitic
DET	Presence of the definite determiner
GEN	Gender
NUM	Number
PER	Person
VOX	Voice
ASP	Aspect
MOD	Mood
NUN	Presence of nunation (indefiniteness marker)
CON	Construct state (head of a genitive construction)
CAS	Case

Figure 1: List of (inflectional) morphological features used in our system; the first ten are features which (roughly) can be determined with higher accuracy since they rely less on syntactic context and more on visible inflectional morphology

tasks: **DiacFull** (predicting all diacritics of a given word), which relates to lexeme choice and morphology tagging, and **DiacPart** (predicting all diacritics of a given word except those associated with the final letter), which relates largely to lexeme choice.

Lemmatization (**LexChoice**) for Arabic has not been discussed in the literature to our knowledge. A **lexeme** is an abstraction over a set of inflected word forms, and it is usually represented by its **citation form**, also called **lemma**.

Finally, **AllChoice** is the combined task of choosing all inflectional and lexemic aspects of a word in context.

This gives us a total of seven tasks. **AllChoice** is the hardest of our tasks, since it subsumes all other tasks. **MorphAll** is the hardest of the three morphological tagging tasks, subsuming **MorphPart** and **MorphPOS**, and **DiacFull** is the hardest lexical task, subsuming **DiacPart**, which in turn subsumes **LexChoice**. However, **MorphAll** and **DiacFull** are (in general) orthogonal, since **MorphAll** has no lexemic component, while **DiacFull** does.

3 Our System

Our system, MADA, makes use of 19 orthogonal features to select, for each word, a proper **analysis** from a list of potential analyses provided by the BAMA dictionary. The BAMA analysis which matches the most of the predicted features wins; the weighting of the features is one of the topics of this paper. These 19 features consist of the 14 morphological features shown in Figure 1, which MADA predicts using 14 distinct Support Vector Machines trained on ATB3-Train (as defined by Zitouni et al. (2006)). In addition, MADA uses five additional features. **Spellmatch** determines whether the diacritized form of the suggested analysis and the input word match if both are stripped of all of their diacritics. This is useful because sometimes BAMA suggests analyses which imply a different spelling of the undiacritized word, but these analyses are often incorrect. **Isdefault** identifies those analyses that are the default output of BAMA (typically, these are guesses that the word in question is a proper noun); these analyses are less likely to be correct than others suggested by BAMA. MADA can derive the values of **Spellmatch** and **Isdefault** by direct examination of the analysis in question, and no predictive model is needed. The fourteen morphological features plus **Spellmatch** and **Isdefault** form a feature collection that is entirely based on morphological (rather than lexemic) features; we refer to this collection as **BASE-16**. **UnigramDiac** and **UnigramLex** are unigram models of the surface diacritized form and the lexeme respectively, and contain lexical information. We also build 4-gram lexeme models using an open-vocabulary language model with Kneser-Ney smoothing, by means of the SRILM toolkit (Stolcke, 2002). The model is trained on the same corpus used to train the other classifiers, ATB3-Train. (We also tested other n-gram models, and found that a 4-gram lexeme model outperforms the other orders with $n \leq 5$, although the improvement over the trigram and 5-gram models was less than 0.01%.) The 4-gram model, on its own, correctly selects the lexeme of words in ATB3-DevTest 94.1% of the time. The 4-gram lexeme model was incorporated into our system as a full feature (**NGRAM**). We refer to the feature set consisting of **BASE-16** plus the two unigram mod-

els and **NGRAM** as **FULL-19**.

Optimizing the feature weights is a machine learning task. To provide learning data for this task, we take the ATB3-DevTest data set and divide it into two sections; the first half (∼26K words) is used for tuning the weights and the second half (∼25K words) for testing. In a pre-processing step, each analysis in appended with a set of labels which indicate whether the analysis is correct according to seven different evaluation metrics. These metrics correspond in a one-to-one manner to the seven different disambiguation tasks discussed in Section 2, and we use the task name for the evaluation label. Specifically, the **MorphPOS** label is positive if the analysis has the same POS value as the correct analysis in the gold standard; the **LexChoice** label provides the same information about the lexeme choice. The **MorphPart** label is positive if the analysis agrees with the gold for each of the 10 basic features used by Habash and Rambow (2005). A positive **MorphAll** label requires that the analysis match the gold in all morphological features, i.e., in every feature except the lexeme choice and diacritics. The **DiacFull** label is only positive if the surface diacritics of the analysis match the gold diacritics exactly; **DiacPart** is less strict in that the trailing sequence diacritic markers in each surface diacritic are stripped before the analysis and the gold are compared. Finally, **AllChoice** is only positive if the analysis was one chosen as correct in the gold; this is the strictest form of evaluation, and there can be only one positive **AllChoice** label per word.

In addition to labeling as described in the preceding paragraph, we run MADA on the tuning and test sets. This gives us a set of model predictions for every feature of every word in the tuning and test sets. We use an implementation of a Downhill Simplex Method in many dimensions based on the method developed by Nelder and Mead (1965) to tune the weights applied to each feature. In a given iteration, the Simplex algorithm proposes a set of feature weights. These weights are given to a weight evaluation function; this function determines how effective a particular set of weights is at a given disambiguation task by calculating an **overall score** for the weight set: the number of words in the tuning set that were correctly disambiguated. In order to compute this score, the weight evaluation function examines each proposed analysis for each word in the tuning set. If the analysis and the model prediction for a feature of a given word agree, the **analysis score** for that analysis is incremented by the weight corresponding to that feature. The analysis with the highest analysis score is selected as the proper analysis for that word. If the selected analysis has a positive task label (i.e., it is a good answer for the disambiguation task in question), the overall score for the proposed weight set is incremented. The Simplex algorithm seeks to maximize this overall score (and thus choose the weight set that performs best for a given task).

Once the Simplex algorithm has converged, the optimal feature weights for a given task are known. Our system makes use of these weights to select a correct analysis in the test set. Each analysis of each word is given a score that is the sum of optimal feature weights for features where the model prediction and the analysis agree. The analysis with the highest score is then chosen as the correct analysis for that word. The system can be evaluated simply by comparing the chosen analysis to the gold standard. Since the Simplex weight evaluation function and the system use identical means of scoring analyses, the Simplex algorithm has the potential to find very optimized weights.

4 Experiments

We have three main research hypotheses: (1) Using lexemic features helps in all tasks, but especially in the diacritization and lexeme choice tasks. (2) Tuning the weights helps over using identical weights. (3) Tuning to the task that is evaluated improves over tuning to other tasks. For each of the two feature sets, **BASE-16** and **FULL-19**, we tune the weights using seven **tuning metrics**, producing seven sets of weights. We then evaluate the seven automatically weighted systems using seven **evaluation metrics**. The tuning metrics are identical to the evaluation metrics and they correspond to the seven tasks described in Section 2. Instead of showing 98 results, we show in Figure 2 four results for each of the seven tasks: for both the **BASE-16** and **FULL-19** feature sets, we give the untuned performance, and then the best-performing tuned performance. We indicate which tuning metric provided the best tun-

Task	Baseline	**BASE-16** (Morph Feats Only)			**FULL-19** (All Feats)		
		Not Tuned	Tuned	Tuning metric	Not Tuned	Tuned	Tuning metric
MorphPOS	95.5	95.6	96.0	**MorphAll**	96.0	96.4	**MorphPOS**
MorphPart	93.8	94.1	94.8	**AllChoice**	94.7	95.1	**DiacPart**
MorphAll	83.8	84.0	84.8	**AllChoice**	82.2	85.1	**MorphAll**
LexChoice	85.5	86.6	87.5	**MorphAll**	95.4	96.3	**LexChoice**
DiacPart	85.1	86.4	87.3	**AllChoice**	94.8	95.4	**DiacPart**
DiacFull	76.0	77.1	78.2	**MorphAll**	82.6	86.1	**MorphAll**
AllChoice	73.3	74.5	75.6	**AllChoice**	80.3	83.8	**MorphAll**

Figure 2: Results for morphological tagging tasks (percent correct); the baseline uses only 14 morphological features with identical weights; "Tuning Metric" refers to the tuning metric that produced the best tuned results, as shown in the "Tuned" column

ing performance. The Baseline indicated in Figure 2 uses the 14 morphological features (listed in Figure 1) only, with no tuning (i.e., all 14 features have a weight of 1). The untuned results were determined by also setting almost all feature weights to 1; the only exception is the **Isdefault** feature, which is given a weight of -(8/14) when included in untuned sets. Since this feature is meant to penalize analyses, its value must be negative; we use this particular value so that our results can be readily compared to previous work. All results are the best published results to date on these test sets; for a deeper discussion, see the longer version of this paper which is available as a technical report.

We thus find our three hypotheses confirmed: (1) Using lexemic features reduces error for the morphological tagging tasks (measured on tuned data) by 3% to 11%, but by 36% to 71% for the diacritic and lexeme choice tasks. The highest error reduction is indeed for the lexical choice task. (2) Tuning the weights helps over using identical weights. With only morphological features, we obtain an error reduction of between 4% and 12%; with all features, the error reduction from tuning ranges between 8% and 20%. (3) As for the correlation between tuning task and evaluation task, it turned out that when we use only morphological features, two tuning tasks worked best for all evaluation tasks, namely **MorphAll** and **AllChoice**, thus not confirming our hypothesis. We speculate that in the absence of the lexical features, more features is better (these two tasks are the two hardest tasks for morphological features only). If we add the lexemic features, we do find our hypothesis confirmed, with almost all evaluation tasks performing best when the weights are tuned for that task. In the case of the three exceptions, the differences between the best performance and performance when tuned to the same task are very slight ($< 0.06\%$).

References

Tim Buckwalter. 2004. Buckwalter Arabic morphological analyzer version 2.0.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *ACL'05*, Ann Arbor, MI, USA.

Nizar Habash and Owen Rambow. 2007. Arabic diacritization through full morphological tagging. In *NAACL HLT 2007 Companion Volume, Short Papers*, Rochester, NY, USA.

Jan Hajič, Otakar Smrž, Tim Buckwalter, and Hubert Jin. 2005. Feature-based tagger of approximations of functional Arabic morphology. In *Proceedings of the Workshop on Treebanks and Linguistic Theories (TLT)*, Barcelona, Spain.

Jan Hajič. 2000. Morphological tagging: Data vs. dictionaries. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL'00)*, Seattle, WA.

J.A Nelder and R Mead. 1965. A simplex method for function minimization. In *Computer Journal*, pages 303–333.

Andreas Stolcke. 2002. Srilm - an extensible language toolkit. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*.

Imed Zitouni, Jeffrey S. Sorensen, and Ruhi Sarikaya. 2006. Maximum entropy based restoration of arabic diacritics. In *Coling-ACL'06*, pages 577–584, Sydney, Australia.