Improving Arabic Diacritization through Syntactic Analysis

Anas Shahrour, Salam Khalifa and Nizar Habash
Computational Approaches to Modeling Language Lab
New York University Abu Dhabi
United Arab Emirates
{anas.shahrour,salamkhalifa,nizar.habash}@nyu.edu

Abstract

We present an approach to Arabic automatic diacritization that integrates syntactic analysis with morphological tagging through improving the prediction of case and state features. Our best system increases the accuracy of word diacritization by 2.5% absolute on all words, and 5.2% absolute on nominals over a state-of-the-art baseline. Similar increases are shown on the full morphological analysis choice.

1 Introduction

Modern Standard Arabic (MSA) orthography generally omits diacritical marks which encode lexical as well as syntactic (case) information. The task of Arabic automatic diacritization is about the automatic restoration of the missing diacritics. Diacritization improvement in Arabic has important implications for downstream processing for Arabic natural language processing, e.g. speech recognition (Ananthakrishnan et al., 2005; Biadsy et al., 2009), speech synthesis (Elshafei et al., 2002), and machine translation (Diab et al., 2007; Zbib et al., 2010).

Previous efforts on diacritization utilized morphological tagging techniques to disambiguate word forms. Habash et al. (2007a) observe that while such techniques work relatively well on lexical diacritics (located on word stems), they are much worse for syntactic case diacritics (typically word final). They suggest that syntactic analysis may help with automatic diacrtization, but stop short of testing the idea, and instead demonstrate that complex linguistic features and rules are needed to model complex Arabic case using gold syntactic analyses. In this paper, we develop an approach for improving the quality of automatic Arabic diacritization through the use of automatic syntactic analysis. Our approach combines hand-written rules for case assignment and agreement with machine learning of case and state adjustment on the output of a state-of-the-art morphological tagger. Our best system increases the accuracy of word diacrtization by 2.5% absolute overall, and 5.2% absolute on nominals over a state-of-the-art baseline.

2 Linguistic Background

Arabic automatic processing, and specifically diacritization is hard for a number of reasons.

First, Arabic words are morphologically rich. The morphological analyzer we use represents Arabic words with 15 features (Pasha et al., 2014).[1] We focus on case and state in this paper. In our data set, *case* has five values: nominative (n), accusative (a), genitive (g), undefined (u) and not applicable (na). Cases n, a and g are usually expressed with an overt morpheme. Case u is used to mark words without an overt morpheme expression of case (e.g., invariable nouns such as شَكوَى *šakwaý* 'complaint'), or those not assigned a case in the manual annotations. Most of the missing assignments are for foreign proper nouns, which often do not receive case markers. However, this is not done consistently in the training data we use. Case na is used for non-nominals. *State* is a nominal feature that has four values: definite (d), indefinite (i), construct (c) and not applicable (na). State generally reflects the definiteness in nominals (d vs i) and whether a nominal is the head of genitive construction (aka *Idafa*) (c). State na is used for non-nominals.[2] For the most part, case and state realize as a single word-final morpheme, e.g., the suffix اً **Aã** in كِتَابًا *kitAbAã*[3] is a morpheme indicating the word is (cas:a, stt:i).

Second, undiacritized Arabic words are highly ambiguous: in our training data, words had an average of 12.8 analyses per word, most of which are associated with different diacritizations. Some diacritization differences reflect different analysis

[1]Lemma (lex), Part-of-Speech (pos), Gender (gen), Number (num), Case (cas), State (stt), Person (per), Aspect (asp), Voice (vox), Mood (mod), four proclitics (prc$n$), and one enclitic (enc0).

[2]For a detailed discussion of Arabic case and state, see (Smrž, 2007; Habash et al., 2007a).

[3]Arabic transliteration is presented in the Habash-Soudi-Buckwalter scheme (Habash et al., 2007b).

lemmas; while others are due to morpho-syntactic differences. For example, the undiacritized version of the word we used above (كتابا *ktAbA*) has two other diacritizations and analyses: كُتَّابًا *kut∼AbAã* (cas:a stt:i) 'writers' (different lemma) and كِتَابَا *kitAbA* (cas:n stt:c num:d) 'two books of [...]' (different features).

Third, Arabic has complex case/state assignment and agreement patterns that interact with the sentence's syntax. For example, a noun may get its case by being subject of a verb and its state by being the head of an idafa construction; while adjectives modifying this noun agree with it in its case, their state is determined by the state of the last element in the idafa chain the noun heads.

For more information on Arabic orthography, morphology, and syntax, see Habash (2010).

3 Related Work

Much work has been done on Arabic diacritization (Vergyri and Kirchhoff, 2004; Nelken and Shieber, 2005; Zitouni et al., 2006; Habash and Rambow, 2007; Alghamdi and Muzafar, 2007; Rashwan et al., 2009; Bebah et al., 2014; Hadj Ameur et al., 2015; Abandah et al., 2015; Bouamor et al., 2015). We refer the reader to the extensive literature review by Abandah et al. (2015), and focus on systems we compare with.

Most of the previous approaches cited above utilize different sequence modeling techniques that use varying degrees of knowledge from shallow letter and word forms to deeper morphological information; none to our knowledge make use of syntax. Habash and Rambow (2007) approach diacritization as a part of the morphological disambiguation problem, where they select the optimal full morphological tag for Arabic in context and use it to select from a list of possible analyses produced by a morphological analyzer. They use independent taggers for all features; and language models for lemmas and diacritized surface forms. Their work is part of the state-of-the-art Arabic morphological tagger MADAMIRA (Pasha et al., 2014). Our paper is most closely related to Habash and Rambow (2007) and Pasha et al. (2014). We extend their work using additional syntactic features to improve morphological disambiguation accuracy. We demonstrate improvements in terms of both full morphological analysis choice (lemmatization, tokenization, all features) as well as word diacritization.

Most recently, Abandah et al. (2015) presented a recurrent neural network approach to diacritize full sentences with impressive results. We do not compare to their effort here but we note that they use an order of magnitude more diacritized data than we do, and they focus on diacritization only as opposed to full morphological analysis.

In related work on modeling Arabic case and syntax, Habash et al. (2007a) compared rule-based and machine learning approaches to capture the complexity of Arabic case assignment and agreement. They demonstrated their results on gold syntactic analyses showing that given good syntactic representations, case prediction can be done at a high degree of accuracy. Alkuhlani et al. (2013) later extended this work to cover all morphological features, including state. Additionally, Marton et al. (2013) demonstrated that in the context of syntactic dependency parsing, case is the best feature to use *in gold settings* and is the worst feature to use *in predicted settings*. In this paper we use automatic (i.e. not gold) syntactic features to improve case prediction, which improves morphological analysis and word diacrtization.

4 Approach

Motivation We are motivated by an error analysis we conducted of 1,200 words of the MADAMIRA system output. We found a large number of surprising *syntactically impossible* case errors such as genitive nouns following verbs or construct nouns followed by non-genitives. We explain these errors by MADAMIRA's contextual models being limited to a small window of neighboring words and with no modeling of syntax, which leads to a much worse performance on case and diacritization compared to lemmas and POS (almost 10% absolute drop from 96% to 86%).

Proposed Solution Our approach is to provide better prediction of case and state using models with access to additional information, in particular syntactic analysis and rules. The predicted case and state values are then used to re-tag the MADAMIRA output by selecting the best match in its ranked morphological analyses. We limit our retagging to nominals. Since what we are learning to predict is how to correct MADAMIRA's baseline choice (as opposed to a generative model of case-state), we also re-apply the model on its output to fix primarily propagated agreement errors in a manner similar to Habash et al. (2007a)'s agreement classifier.[4]

[4]We optimized for a re-application limit, which we found invariably to be +1 time or +2 times in our *DevTest* experi-

5 Experimental Results

We present next our experimental results and compare five case-state prediction techniques. The results for these techniques are compared to our state-of-the-art baseline system, which has been compared to a number of other approaches.

5.1 Experimental Setup

Data We used the Penn Arabic Treebank (PATB, parts 1, 2 and 3) (Maamouri et al., 2004; Maamouri et al., 2006; Maamouri et al., 2009) split into *Train*, *Dev* and (blind) *Test* along the recommendations of Diab et al. (2013) which were also used in the baseline system. The morphological feature representations we use are derived from the PATB analyses following the approach used in the MADA and later MADAMIRA systems (Habash and Rambow, 2005; Habash and Rambow, 2007; Pasha et al., 2014). We further divide *Dev* into two parts with equal number of sentences: *DevTrain* (30K words) for training our case-state classifiers, and *DevTest* (33K words) for development testing. The *Test* set has 63K words.[5]

Evaluation Metrics We report our accuracy in terms of two metrics: a. **Diac**, the percentage of correctly fully diacrtized words; and b. **All**, the percentage of words for which a full morphological analysis (lemma, POS, all inflectional and clitic features, and diacritization) is correctly predicted. We report the results on all words (**All Words**) as well as on nominals[6] with no u case in the gold (henceforth, **Nominals**). We do not report on case and state prediction accuracy, nor on the character-level diacritization.

Morphological Analysis, Baseline and Topline For our baseline, we use the morphological analysis and disambiguation tool MADAMIRA (Pasha et al., 2014), which produces a contextually ranked list of analyses for each word. We computed an oracular topline using the PATB gold case and state values in the retagging process.

ments. We do not report more on this due to space limitations. Because of the need to have the same of number of parse tree tokens for reapplication in the models using syntax, we constrain the retagging to maintain the same clitic signature (number of clitics) in the MADAMIRA baseline top analysis.

[5]To address the concern that we are using more "training" data than our baseline, we compared the performance of MADAMIRA's release (baseline) to a version that was trained on Train + *DevTrain*. The small increase in training data made no significant difference from the baseline system in terms of our metrics.

[6]Nominals consists of nouns (including noun_quant and noun_num), adjectives, proper nouns, adverbs, and pronouns.

	All Words		**Nominals**	
System	**Diac**	**All**	**Diac**	**All**
Oracle Topline	96.2	94.2	98.0	95.7
Baseline	87.4	85.0	81.9	78.2
Morphology Rules	88.0	85.5	83.1	79.4
Morphology Classifier	88.4	85.9	83.7	80.1
Syntax Rules	87.4	84.6	**86.4**	82.2
Syntax Classifier	89.1	86.6	85.2	81.4
Syntax Rules+Classifier	**89.7**	**87.1**	**86.4**	**82.5**

Table 1: Results on *DevTest*.

Syntactic Analysis For syntactic features, we trained an Arabic dependency parser using MaltParser (Nivre et al., 2007) on the Columbia Arabic Treebank (CATiB) version of the PATB (Habash and Roth, 2009). The *Train* data followed the same splits mentioned above. The Nivre "eager" algorithm was used in all experiments. The CATiB dependency tree has six simple POS tags and eight relations (Habash and Roth, 2009; Habash et al., 2009). The PATB tokenization as well as the CATiB POS tags were produced by the baseline system MADAMIRA and used as input to the parser. We also used the well performing yet simple CATiBex expansion of the CATiB tags as implemented in the publicly available parsing pipeline from Marton et al. (2013). Our parser's performance on the PATB *Dev* set is comparable to Marton et al. (2013): 84.2% labeled attachment, 86.6% unlabeled attachment, and 93.6% label accuracy.

Machine Learning Technique Given the small size of *DevTrain*, we opted to train unlexicalized models that we expect to capture morphosyntactic abstractions. We tried a number of machine learning techniques and settled on using the J48 Decision tree classifier with its default settings in WEKA (Hall et al., 2009; Quinlan, 1993) for all of the classification experiments in this paper.

5.2 Case and State Classification Techniques

We detail and report on five case-state classification techniques we experimented with. All results on the *DevTest* are presented in Table 1.

Morphology Rules We created a simple manual word-morphology based classifier that handled the most salient case errors seen in our pilot study (Section 4) and whose correction has a high precision. The scope of the rules was limited to word bigrams and included three conditions: (i) postverbal genitive nouns are changed to the first non-genitive analysis available from MADAMIRA; (ii) post-construct state non-genitive nouns should become genitive; and (iii) adjectives agreeing with the nouns they follow in gender, number, and definiteness, but not in case should match the nouns' case. The collective improvement of all the rules applied in the order presented above adds up to 0.6% absolute on All Word Diac, and 1.2% absolute on Nominal Diac.

Morphology-based Classifier We trained a classifier to predict a correction of the baseline case and state of a word using the *DevTrain* data set. For features, we included all the non-lexical morphological features (all features mentioned in *footnote 1* except for lemma). Clitic feature values were binarized to indicate if a clitic is present or not. We used *Nil* values for all out-of-vocabulary words in MADAMIRA. BOS and EOS placeholders were used for sentence boundaries. We excluded all *DevTrain* words whose predicted POS switches from nominal to non-nominal or vice versa, but kept them as part of other words' context. This minimizes noise and sparsity in the training data, especially given its small size and the rarity of such examples. We experimented with adding features from neighboring words within a window size of +/- 2 words. The best performing setup was with window size +/- 1. This classifier makes around 0.5% absolute gain over the simple word morphology rules.

Syntax Rules Inspired by Habash et al. (2007a)'s simple set of rules for determining case on gold dependency trees, we re-implemented these rules to work with our different dependency representation and extended them to include state assignment in a manner similar to Alkuhlani et al. (2013).[7] These rules improve over the baseline by 4.5% absolute in Nominal Diac accuracy, but produce no gains in All Words setting. An investigation of the error patterns reveals the main reason to be the rules' inability to predict the problematic u case.

Syntax-based Classifier We trained a classifier to predict a correction of the case and state of a word on a parsed version of the *DevTrain* data set. Since the syntactic parse separates most clitics in the PATB tokenization, we align the tree tokens with the word morphology before extracting classifier features. The features we used are the token's morphological features (same as those used for words in the morphology-based classifier), the parent and grandparent's features, the relations between token and parent, and between parent and grandparent, and the features of the neighboring +/- n tokens. The tokenized clitics are only used as context features (tree and surface). We tested all combinations of values of n as 0, 1, 2, and 3 and of including parent and grandparent features. The best performing setting was with including the parent and grandparent as well as a window of +/- 1 tokens. The syntax-based classifier improves over the morphology-based classifier but it trails behind syntax rules in terms of Nominal Diac. Its All Word Diac accuracy is the highest so far.

Combination of Syntax Rules and Classifier We combine the last two approaches to exploit their complementary performance by including the syntax rule predictions as features in the syntax-based classifier. The resulting system is our best performer achieving *DevTest* accuracy improvements of 2.3% absolute (All Word Diac), and 4.5% absolute (Nominal Diac).

5.3 Blind Test Results

The results of applying the best approach and settings on our blind *Test* set are presented in Table 2. While the baseline on *Test* is slightly higher than *DevTest*, the performance on all metrics are comparable. The increase in Diac accuracy on All Words is 2.5% absolute and on Nominals is 5.2% absolute. The corresponding relative reductions in error to the oracle toplines are 30% and 34%. Similar increases are shown on the full morphological analysis choice.

	All Words		**Nominals**	
System	**Diac**	**All**	**Diac**	**All**
Oracle Topline	96.3	94.4	97.9	95.6
Baseline	88.1	86.0	82.4	79.4
Syntax Rules+Classifier	**90.6**	**88.5**	**87.6**	**84.5**

Table 2: Results on the blind *Test* set.

[7]For nominal case assignment, our rules are: (i) default case is a; (ii) children of $root$ are n; (iii) object of preposition and idafa children are assigned g; (iv) predicates not headed by verbs are assigned n; and (v) subjects and topics not headed (or grand-headed) by a class of words called *Ân∼a and her sisters* are assigned n. After case assignment, we apply two case agreement rules: (i) for all nominals modifying case-assigned nominals (i.e., with tree relation mod), copy the case of the modified nominal; and (ii) for all nominals conjoined to case-assigned nominals (i.e., with tree chain $nominal\text{-}conjunction\text{-}nominal$), copy the case of the heading nominal. For state, we assign d to words with the definite article proclitic, and c to words heading an idafa construction. All other nominals are assigned state i.

(a) تعليقي الأساسي أن الأمن في لبنان جَيِّدٌ tςlyqy AlÂsAsy Ân AlÂmn fy lbnAn jay∼idũ

My main comment is that security in Lebanon is good

Word	Dependency	Gold	Baseline	Best System
جيد jyd	PRD of Ân 'that'	jay∼idũ [NOM]	jay∼idĩ [GEN]	jay∼idũ [NOM]

(b) كان جان غانم يَدَهُ اليُمنَى kAn jAn γAnm ydh Alymný

Jean Ghanem was his right hand

Word	Dependency	Gold	Baseline	Best System
يده ydh	PRD of kAn 'was'	yadahu [ACC]	yadihi [GEN]	yadahu [ACC]
اليمنى Alymný	MOD of yd 'hand'	Alyum.naý [U]	Alyamaniy∼i [GEN]	Alyum.naý [U]

(c) سيعزز العَلَاقَاتِ الاَقتِصَادِيَّةَ مع أوروبا syςzz AlςlAqAt AlAqtSAdyħ mς ÂwrwbA

It will reinforce the economic relations with Europe

Word	Dependency	Gold	Baseline	Best System
العلاقات AlςlAqAt	OBJ of yςzz 'reinforce'	AlςalAqAti [ACC]	AlςalAqAti [GEN]	AlςalAqAti [ACC]
الاقتصاديّة AlAqtSAdyħ	MOD of AlςlAqAt 'relations'	AlAiqtiSAdiy∼aħa [ACC]	AlAiqtiSAdiy∼aħi [GEN]	AlAiqtiSAdiy∼aħa [ACC]

(d) أكد وزير الاقتصاد باسل فليحان إثر لقائه أمس رَئِيسَ الوزراء رفيق الحريري أن اتفاق ...

Âkd wzyr AlAqtSAd bAsl flyHAn Ǎθr lqAŷh Âms rŷys AlwzrA' rfyq AlHryry Ân AtfAq ...

Following his meeting with the Prime Minister Rafiq Alhariri,
the Minister of Economy Basil Flaihan has confirmed that the agreement ...

Word	Dependency	Gold	Baseline	Best System
رئيس rŷys	SBJ of Âkd 'confirmed'	raŷiysa [ACC]	raŷiysa [ACC]	raŷiysu [NOM]

Figure 1: Examples of corrections from our best performing system.

5.4 Error Analysis and Examples

We manually investigated the types of errors in the first 100 errors in the *DevTest* in our best system's output. In about a quarter of the cases (23%), all of which proper nouns, the gold reference was not fully diacritized, making it impossible to evaluate. In an additional 7%, typographical errors in the input including missing sentence breaks led to bad parses which are the likely cause of error. The rest of the errors are system failures: 31% are connected with syntactic tree errors (although a third of these are due to agreement-propagated errors over correct trees); 28% are due to other morphological analysis issues (half are out-of-vocabulary and 4% are no-analysis cases); and 11% are other case selection errors unrelated to the above-mentioned issues.

Figure 1 shows four examples from the *DevTest* where the analysis of our best system for the underlined words is different from baseline. Examples (a), (b), and (c) are cases where our best system's analysis matched the gold. The dependency relation also matched the gold and was the likely cause of correction. In example (d), our best system incorrectly changed the correct baseline analysis in agreement with the wrong dependency relation provided by the parser, which is the likely cause of error.

6 Conclusion and Future Work

We have demonstrated the value of using automatic syntactic analysis as part of the task of automatic diacritization and morphological tagging of Arabic. Our best solution is a hybrid approach that combines statistical parsing and manual syntactic rules as part of a machine learning model for correcting case and state features.

In the future, we plan to investigate the development of joint morphological disambiguation and syntactic parsing models. We will also work on improving the quality of Arabic parsing which is behind many of the errors according to our error analysis. Other possible directions include using more sophisticated machine learning techniques and richer lexical features. We also plan to host a demo and make our system available through the website of the Computational Approaches to Modeling Language (CAMeL) Lab: `www.camel-lab.com`.

Acknowledgments

We would like to thank Arfath Pasha for his helpful support overall, and especially for retraining the special version of MADAMIRA we discuss in *footnote 5*.

References

Gheith A Abandah, Alex Graves, Balkees Al-Shagoor, Alaa Arabiyat, Fuad Jamour, and Majid Al-Taee. 2015. Automatic diacritization of Arabic text using recurrent neural networks. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(2):183–197.

Mansour Alghamdi and Zeeshan Muzafar. 2007. KACST Arabic diacritizer. In *First International Symposium on Computers and Arabic Language*, pages 25–28.

Sarah Alkuhlani, Nizar Habash, and Ryan Roth. 2013. Automatic morphological enrichment of a morphologically underspecified treebank. In *HLT-NAACL*, pages 460–470.

S. Ananthakrishnan, S. Narayanan, and S. Bangalore. 2005. Automatic diacritization of Arabic transcripts for ASR. In *Proceedings of ICON*, Kanpur, India, December.

Mohamed Bebah, Chennoufi Amine, Mazroui Azzeddine, and Lakhouaja Abdelhak. 2014. Hybrid approaches for automatic vowelization of Arabic texts. *arXiv preprint arXiv:1410.2646*.

Fadi Biadsy, Nizar Habash, and Julia Hirschberg. 2009. Improving the Arabic Pronunciation Dictionary for Phone and Word Recognition with Linguistically-Based Pronunciation Rules. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 397–405, Boulder, Colorado.

Houda Bouamor, Wajdi Zaghouani, Mona Diab, Ossama Obeid, Kemal Oflazer, Mahmoud Ghoneim, and Abdelati Hawwari. 2015. A pilot study on Arabic multi-genre corpus diacritization. In *Proceedings of the Second Workshop on Arabic Natural Language Processing*, pages 80–88, Beijing, China, July. Association for Computational Linguistics.

Mona Diab, Mahmoud Ghoneim, and Nizar Habash. 2007. Arabic Diacritization in the Context of Statistical Machine Translation. In *Proceedings of Machine Translation Summit (MT-Summit)*, Copenhagen, Denmark.

Mona Diab, Nizar Habash, Owen Rambow, and Ryan Roth. 2013. LDC Arabic treebanks and associated corpora: Data divisions manual. *arXiv preprint arXiv:1309.5652*.

Moustafa Elshafei, Husni Al-Muhtaseb, and Mansour Al-Ghamdi. 2002. Techniques for high quality Arabic speech synthesis. *Information sciences*, 140(3):255–267.

Nizar Habash and Owen Rambow. 2005. Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 573–580, Ann Arbor, Michigan.

Nizar Habash and Owen Rambow. 2007. Arabic Diacritization through Full Morphological Tagging. In *Proceedings of the 8th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL07)*.

Nizar Habash and Ryan M Roth. 2009. CATiB: The Columbia Arabic Treebank. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 221–224. Association for Computational Linguistics.

Nizar Habash, Ryan Gabbard, Owen Rambow, Seth Kulick, and Mitchell P Marcus. 2007a. Determining case in Arabic: Learning complex linguistic behavior requires complex linguistic features. In *EMNLP-CoNLL*, pages 1084–1092.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007b. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Nizar Habash, Reem Faraj, and Ryan Roth. 2009. Syntactic Annotation in the Columbia Arabic Treebank. In *Proceedings of MEDAR International Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

Nizar Habash. 2010. *Introduction to Arabic Natural Language Processing*. Morgan & Claypool Publishers.

Mohamed Seghir Hadj Ameur, Youcef Moulahoum, and Ahmed Guessoum. 2015. Restoration of Arabic diacritics using a multilevel statistical model. In *Computer Science and Its Applications*, volume 456 of *IFIP Advances in Information and Communication Technology*, pages 181–192. Springer International Publishing.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, pages 102–109, Cairo, Egypt.

Mohamed Maamouri, Ann Bies, and Seth Kulick. 2006. Diacritization: A challenge to Arabic treebank annotation and parsing. In *Proceedings of the Conference of the Machine Translation SIG of the British Computer Society*.

Mohamed Maamouri, Ann Bies, and Seth Kulick. 2009. Creating a Methodology for Large-Scale Correction of Treebank Annotation: The Case of the Arabic Treebank. In *Proceedings of MEDAR International Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

Yuval Marton, Nizar Habash, and Owen Rambow. 2013. Dependency parsing of modern standard Arabic with lexical and inflectional features. *Computational Linguistics*, 39(1):161–194.

Rani Nelken and Stuart Shieber. 2005. Arabic Diacritization Using Weighted Finite-State Transducers. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages at 43rd Meeting of the Association for Computational Linguistics (ACL'05)*, pages 79–86, Ann Arbor, Michigan.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: a language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Arfath Pasha, Mohamed Al-Badrashiny, Ahmed El Kholy, Ramy Eskander, Mona Diab, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In *In Proceedings of LREC*, Reykjavik, Iceland.

Ross Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA.

Mohsen Rashwan, Mohammad Al-Badrashiny, Mohamed Attia, and S Abdou. 2009. A hybrid system for automatic Arabic diacritization. In *The 2nd International Conference on Arabic Language Resources and Tools*. Citeseer.

Otakar Smrž. 2007. *Functional Arabic Morphology. Formal System and Implementation*. Ph.D. thesis, Charles University in Prague, Prague, Czech Republic.

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 Workshop on Computational Approaches to Arabic Script-based Languages*, pages 66–73, Geneva, Switzerland.

Rabih Zbib, Spyros Matsoukas, Richard Schwartz, and John Makhoul. 2010. Decision trees for lexical smoothing in statistical machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 428–437, Uppsala, Sweden, July. Association for Computational Linguistics.

Imed Zitouni, Jeffrey S. Sorensen, and Ruhi Sarikaya. 2006. 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*, pages 577–584, Sydney, Australia, July. Association for Computational Linguistics.