MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic

Arfath Pasha, Mohamed Al-Badrashiny[†], Mona Diab[†], Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan M. Roth[‡]

Center for Computational Learning Systems, Columbia University, New York, NY

[†]Department of Computer Science, The George Washington University, Washington, DC

[‡] Google, Inc

madamira@ccls.columbia.edu

Abstract

In this paper, we present MADAMIRA, a system for morphological analysis and disambiguation of Arabic that combines some of the best aspects of two previously commonly used systems for Arabic processing, MADA (Habash and Rambow, 2005; Habash et al., 2009; Habash et al., 2013) and AMIRA (Diab et al., 2007). MADAMIRA improves upon the two systems with a more streamlined Java implementation that is more robust, portable, extensible, and is faster than its ancestors by more than an order of magnitude. We also discuss an online demo (see `http://nlp.ldeo.columbia.edu/madamira/`) that highlights these aspects.

Keywords: Morphological Analysis, Morphological Tagging, Base Phrase Chunking, Named Entity Recognition

1. Introduction

Arabic is a complex language that poses many challenges to Natural Language Processing (NLP). This is due to three important factors. First, Arabic has a rich inflectional and cliticizational morphology system (for a comprehensive discussion, see (Habash, 2010)). For example, the Modern Standard Arabic (MSA) word وسيكتبونها *wsyktbwnhA (wa+sa+ya-ktub-uwna+hA)*[1] 'and they will write it [lit. and+will+they-write-they+it]' has two proclitics (a conjunction and a tense marker), one circumfix (showing agreement in person, gender and number) and one enclitic (a direct object).

Secondly, Arabic has a high degree of ambiguity resulting from its diacritic-optional writing system and common deviation from spelling standards (e.g., Alif and Ya variants) (Buckwalter, 2007; Habash, 2010). The Standard Arabic Morphological Analyzer (SAMA) (Graff et al., 2009) produces 12 analyses per MSA word on average.

Finally, Arabic has a number of modern dialects that significantly diverge from MSA, which is the language of the news and of formal education. Using NLP tools built for MSA to process dialectal Arabic (DA) is possible but is plagued with very low accuracy: for example, a state-of-the-art MSA morphological analyzer only has 60% coverage of Levantine Arabic verb forms (Habash and Rambow, 2006).

In the presence of these challenges, there is a need for tools that address fundamental NLP tasks for MSA and the dialects. For Arabic, these tasks include diacritization, lemmatization, morphological disambiguation, part-of-speech tagging, stemming, glossing, and (configurable) tokenization. Often these tasks are the first or intermediate steps taken as part of a solution to a larger, more complex NLP problem (such as machine translation); for this reason, any tool addressing these tasks needs to be fast, accurate, and easily connected to other software.

[1]Arabic transliteration is presented in the Buckwalter scheme (Buckwalter, 2004).

2. Previous Work

There has been a considerable amount of work on MSA morphological analysis, disambiguation, part-of-speech (POS) tagging, tokenization, lemmatization and diacritization; for an overview, see (Habash, 2010). And more recently, there has been growing body of work on DA (Al-Sabbagh and Girju, 2012; Mohamed et al., 2012; Habash et al., 2012; Habash et al., 2013) among others.

In this paper, we focus on two systems that are commonly used by researchers in Arabic NLP: MADA (Habash and Rambow, 2005; Roth et al., 2008; Habash et al., 2009; Habash et al., 2013) and AMIRA (Diab et al., 2007). MADA was built for MSA; an Egyptian Arabic (EGY) version (MADA-ARZ) was later built by plugging in the CALIMA EGY analyzer and retraining the models on EGY annotations (Habash et al., 2013). MADA uses a morphological analyzer to produce, for each input word, a list of analyses specifying every possible morphological interpretation of that word, covering all morphological features of the word (diacritization, POS, lemma, and 13 inflectional and clitic features). MADA then applies a set of models – Support Vector Machines (SVMs) and N-gram language models – to produce a prediction, per word in-context, for different morphological features, such as POS, lemma, gender, number or person. A ranking component scores the analyses produced by the morphological analyzer using a tuned weighted sum of matches with the predicted features. The top-scoring analysis is chosen as the predicted interpretation for that word in context; this analysis can then be used to deduce a proper tokenization for the word.

The AMIRA toolkit includes a tokenizer, a part of speech tagger (POS), and a base phrase chunker (BPC), also known as a shallow syntactic parser. The technology of AMIRA is based on supervised learning with no explicit dependence on knowledge of deep morphology; hence, in contrast to MADA, it relies on surface data to learn generalizations. In later versions of AMIRA, a morphological analyzer and a named-entity recognition (NER) component were added. In general, both tools use a unified framework which casts each of the component problems as a classification problem to be solved sequentially. AMIRA takes a multi-step approach to tokenization, part-of-speech tagging and lemmatization, as opposed to MADA, which treats all of these and more in one fell swoop. MADA provides a deeper analysis than AMIRA, namely by identifying syntactic case, mood and construct state in the morphological tag, but that comes at the price of a slower speed. In addition, AMIRA provides additional utilities - BPC and NER - that are not supported by MADA. Both tools are somewhat brittle, academic prototypes implemented in Perl; they rely on third-party software utilities which the end-user must install and configure separately.

3. MADAMIRA 1.0

MADAMIRA follows the same general design as MADA (see Figure 1), with some additional components inspired by AMIRA. Input text (either MSA or EGY) enters the Preprocessor, which cleans the text and converts it to the Buckwalter representation used within MADAMIRA. The text is then passed to the Morphological Analysis component, which develops a list of all possible analyses (independent of context) for each word. The text and analyses are then passed to a Feature Modeling component, which applies SVM and language models to derive predictions for the word's morphological features. SVMs are used for closed-class features, while language models predict open-class features such as lemma and diacritic forms. An Analysis Ranking component then scores each word's analysis list based on how well each analysis agrees with the model predictions, and then sorts the analyses based on that score. The top-scoring analysis of each word can then be passed to the Tokenization component to generate a customized tokenization (or several) for the word, according to the schemes requested by the user. The chosen analyses and tokenizations can then be used by the Base Phase Chunking component to divide the input text into chunks (using another SVM model). Similarly, the Named Entity Recognizer component uses a SVM to mark and categorize named entities within the text. When all the requested components have finished, the results are returned to the user. Users can request specifically what information they would like to receive; in addition to tokenization, base phrase chunks and named entities, the diacritic forms, lemmas, glosses, morphological features, parts-of-speech, and stems are all directly provided by the chosen analysis.

In addition to duplicating the capability of the previous tools, MADAMIRA was designed to be fast, extensible, easy to use and maintain. MADAMIRA is implemented in Java, which provides substantially greater speed than Perl and allows new features to be quickly integrated with the existing code. The third-party language model and NLP SVM utilities used by MADA and AMIRA were discarded and replaced; in addition to improving performance and making the software easier to maintain, this removes the need for the user to install any additional third-party software. MADAMIRA makes use of fast, linear SVMs built using LIBLINEAR (Fan et al., 2008; Waldvogel, 2008); a Java implementation of this tool is packaged with MADAMIRA. We also developed a utility called COGENT[2] to provide an interface between the text-based NLP information and numerical SVM modeling tools such as LIBLINEAR. MADAMIRA still makes use of the same external morphological analyzers used by MADA: SAMA for MSA (Graff et al., 2009) and CALIMA-ARZ for EGY (Habash et al., 2012).

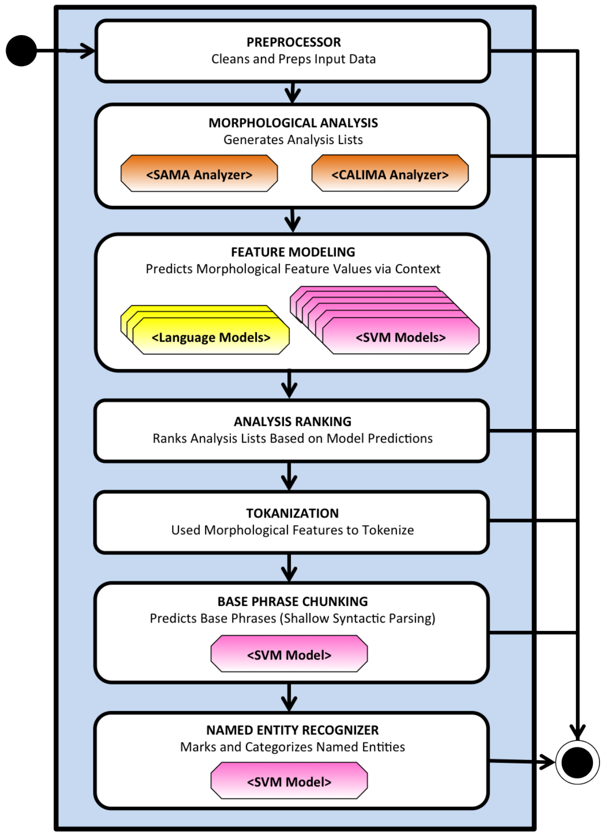

Figure 1: Overview of the MADAMIRA Architecture. The significant system resources (analyzers and models) are indicated. Input text enters the Preprocessor, and flows through the system, with each component adding additional information that subsequent components can make use of. Depending on the requested output, the process can exit and return results at different positions in the sequence.

MADAMIRA also provides XML and HTTP support that was not present in MADA or AMIRA: input and output text can be supplied as plain text or in XML. Furthermore, MADA and AMIRA were designed to start, load all their required resources, process a single input document, and then exit. MADAMIRA can do this as well (Stand-alone mode), but it also can be run in a new, faster Server-client mode of operation. The server-client mode was used with the web demo described in Section 6..

For training data, we used the Penn Arabic Treebank corpus (parts 1, 2 and 3) for MSA (Maamouri et al., 2009); and we used the Egyptian Arabic Treebanks (parts 1 through 6) for EGY (Maamouri et al., 2012). For details of splits of these corpora in terms of training, development and test, see (Diab et al., 2013).

[2] We plan to describe COGENT fully in a future publication.

Alias	Description
ATB	Tokenizes all clitics except for the definite article, normalizes Alif/Ya, uses '+' as clitic markers, and normalizes '(' and ')' characters to "-LRB-" and "-RRB-".
ATB_EVAL	This is the same as ATB, except that the tokens are output in Buckwalter, and tokens of the same word are connected to each other with underscores. This scheme is sometimes used in tokenization evaluations.
ATB_BWFORM	This is the same as ATB, except that it is developed using the BWFORM method.
ATB_BWFORM_EVAL	This is the same as ATB_BWFORM, except that the tokens are output in Buckwalter, and tokens of the same word are connected to each other with underscores. This scheme is sometimes used in tokenization evaluations.
ATB4MT	A large scheme consisting of 6 forms. Tokenizes the same clitics as ATB. Form 0 is the basic token form, without normalizations. Form 1 is the same, but it also normalizes Alif/Ya; Form 2 is the token/word lemma, using '+' clitic markers; Forms 3, 4, and 5 are the part-of-speech tags in the CATiB (Habash and Roth, 2009), Penn ATB (Kulick et al., 2006) and Buckwalter (Buckwalter, 2004) POS tagsets respectively.
D1	Tokenizes QUES and CONJ proclitics only ; uses '+' as a clitic marker, normalizes Alif/Ya, and normalizes '(' and ')' characters to "-LRB-" and "-RRB-".
D2	Same as D1, but also tokenizes PART clitics.
D3	Same as D2, but also tokenizes all articles and enclitics (basically all clitics are tokenized).
D3_BWFORM	This is the same as D3, except that it is developed using the BWFORM method.
D3_BWPOS	This is a 2-form scheme. It tokenizes all clitics and uses '+' as a clitic marker. The first form is the token in Buckwalter; and normalizes Alif/Ya. The second form is the Buckwalter part-of-speech tag.
D34MT	Another large 6-form scheme. Effectively the same as ATB4MT, except that all the clitics are tokenized.

Table 1: Tokenization Alias descriptions. Currently, MADAMIRA for Egyptian Arabic can only use the aliases containing "BWFORM".

4. Tokenization

MADAMIRA currently provides 11 different ways (*schemes*) for tokenizing input text, each specified with an alias term (listed in Table 1). Tokenization schemes are described in terms of what elements are tokenized/separated from the base word, and what format the tokens are presented in. In addition, MADA has two methods of tokenizing: the default, generation-based method, and a simpler, less accurate method based on heuristics (the *BWFORM* method). Currently, only the BWFORM methods are supported for the EGY version.

Table 1 describes what each alias will produce. *Tokenized* indicates that a particular Arabic proclitic or enclitic is separated from the base word and any required spelling adjustments are applied. *Normalize* indicates that a subset of related Arabic characters, when encountered in the tokens, are replaced by a representative character or string; the most common case is normalizing Arabic Alif and Ya characters. *Clitic markers* are extra characters (usually a single "+") that are attached to tokenized clitics to indicate on which side of the clitic the base word lies (that is, which side the clitic was attached to).

Token schemes can provide multiple *forms* for each token; these are the same token represented with different normalizations or different transliterations. Alternatively, a form can show the part-of-speech of the token in one of several tagsets, the word lemma, or other morphological information. A token scheme with more than one form is called a *multi-form scheme*.

Users may specify a configuration that tells MADAMIRA what results are required. Table 2 describes the configuration options. These configuration options are an extension to the options in the MADA system.

5. Evaluation

To evaluate MADAMIRA, a blind test data set (about 25K words for MSA and about 20K words for EGY) was run through MADAMIRA, and the result was compared to a gold, annotated version. The evaluation was conducted across several accuracy metrics (all on the word level):

- **EVALDIAC** – Percentage of words where the analysis chosen by MADAMIRA has the correct fully-diacritized form (with exact spelling).
- **EVALLEX** – Percentage of words where the chosen analysis has the correct lemma.
- **EVALPOS** – Percentage of words where the chosen analysis has the correct part-of-speech , taken from a small set of core part-of-speech tags (verb, noun, adjective, etc).
- **EVALFULL** – Percentage of words where the chosen analysis is perfectly correct (that is, all the morphological features match the gold values). This is the strictest possible metric.
- **EVALATBTOK** – Tokenization evaluation. The percentage of words that have a perfectly correct tokenization (using a common ATB scheme that tokenizes all clitics except Al determiners). Also shown are the percentage of words with correct segmentation (that is, correct number of tokens, even if not correct spelling of each token).

Table 3 shows the accuracy and speed performance of MADAMIRA on the test sets, and compares those numbers to the previous version of MADA and MADA-ARZ. The speed performance is measured in words processed per second; when measuring speed, only a single tokenization

Sub-Element	Attribute	Possible Values	Description
preprocessing	sentence_ids	false | true	If true, the first word of each segment will be considered as the sentence ID for that segment. Mainly used for raw input mode.
	separate_punct	false | true	If true, numbers and punctuation will be separated from words they are connected to, treating them as separate words.
	input_encoding	UTF8 | Buckwalter | SafeBW	Specifies the encoding/transliteration of the input.
overall_vars	output_encoding	UTF8 | Buckwalter | SafeBW	Specifies the encoding/transliteration of the output. Some raw output text is always presented in Buckwalter. Does not control output of tokenized forms.
	dialect	MSA | EGY	Specifies the dialect models used in processing.
	output_analyses	TOP | ALL	If TOP, only the top-scoring analysis will be included in the output. If ALL, all the analyses will be included.
	morph_backoff	NONE | NOAN_PROP | NOAN_ALL | ADD_PROP | ADD_ALL	Sets the morphological back-off; the "PROP" settings add proper noun analyses and "ALL" settings add a wider set. "NOAN" means back-off analyses will only be added to No-analysis words; "ALL" means they will be added to every word. "NONE" means no back-off
	analyze_only	false | true	If true, MADAMIRA will only construct an unranked analysis list for each input word and then stop without applying models, ranking or tokenizing.
requested_output: req_variable	name	PREPROCESSED | LEMMA | DIAC | GLOSS | ASP | CAS | ENC0 | ENC1 | ENC2 | GEN | MOD | NUM | PER | POS | PRC0 | PRC1 | PRC2 | PRC3 | STT | VOX | BW | STEM | SOURCE | LENGTH | OFFSET	These specify what information is requested. The accompanying value attribute, if set to true, indicates that this variable is required. The abbreviations stand for preprocessed word form, word lemma, word with diacritic markers, the English gloss, aspect, case, the word enclitic values, gender, mood, number, person, part-of-speech, the word proclitic values, state, voice, the full Buckwalter tag, the word stem, source of the word analysis, word length and word offset from start of sentence respectively.
tokenization: scheme	alias	D1 | D2 | D3 | ATB | ... ... (see Table 1)	These specify what token schemes are to be used. See Table 1 for details of each alias.

Table 2: Options that can be specified in the MADAMIRA configuration.

is requested. The speed evaluation is conducted for both Stand-alone (raw input) and Server-client modes. For the Server-client evaluations, the server was started and then primed with a small test input (2-4 words). This process ensures that all the required components are fully loaded before the speed benchmarks were calculated.

Table 3 shows respectable accuracy performance. MADAMIRA improves on the older systems for the tokenization task, and for the EGY pos-tagging task. The other accuracy metrics show that the older systems have slightly better performance, but never by more than 0.2% (absolute) for MSA and 0.6% for EGY. This minor accuracy reduction is a trade-off for the substantial speed improvement: MADAMIRA is 16-21x faster than the older system for MSA, and 14-19x faster for EGY. EGY is generally slightly slower than MSA due to the morphological analysis step, which (for EGY) is more complex and tends to generate more analyses for MADAMIRA to consider. We expect that further improvements to the internal models will be able to increase the accuracy while maintaining the word processing throughput.

MADAMIRA also compares favorably with the latest AMIRA system: in tokenizing the MSA test data, AMIRA is able to achieve a tokenization accuracy of 91.4%, and a segmentation accuracy of 99.0%. The word throughput rate for AMIRA (excluding the BPC and NER components) is 49.9 words/sec; however, if only tokenization is required AMIRA can achieve a rate of 255.3 words/sec.

MADAMIRA makes use of several machine learning models. During operation, the required models must be loaded into memory for use. Currently, we recommend 2.5GB of Java heap space when loading all of the MADAMIRA models and resources (the default operation); 1.5GB is sufficient when running in MSA-only or

	MSA		EGY	
Evaluation Metric	**MADA v3.2**	**MADAMIRA v1.0**	**MADA-ARZ v0.4**	**MADAMIRA v1.0**
EVALDIAC	**86.4**	86.3	**83.8**	83.2
EVALLEX	**96.2**	96.0	**87.8**	**87.8**
EVALPOS	**96.1**	95.9	91.8	**92.4**
EVALFULL	**84.3**	84.1	**77.5**	77.3
EVALATBTOK				
Perfect Tokenization	98.8	**98.9**	96.5	**96.6**
Correct Segmentation	99.1	**99.2**	97.4	**97.6**
Speed: words/sec				
Stand-alone mode	48.8	420.2	44.9	389.1
Server-client mode	–	**1013.4**	–	**844.1**

Table 3: Evaluation of MADAMIRA accuracy and speed, compared to MADA for MSA and MADA-ARZ for EGY. The best performing system for each metric and dialect is highlighted in **bold**.

EGY-only mode. We hope to reduce the memory requirements in future releases.
Figures 3 and 4 present two examples of MADAMIRA's output, one for MSA and one for EGY.

6. Online Demo and Availability

Online Demo We have developed an online demo of MADAMIRA, which is located at:
`http://nlp.ldeo.columbia.edu/madamira/`.
A screenshot of this demo can be seen in Figure 2. The web server that operates this demo has a MADAMIRA server process running on it. When a user enters text, the text is bundled and sent to the server process via HTTP POST. The server processes the text and the results are extracted by the client process and displayed on the page. The total processing time for the example shown in Figure 2 was on the order of 20-100 milliseconds.
This version of the demo only displays morphological feature predictions, tokenized forms (using the common "D3" tokenization scheme, which tokenizes all clitics), diacritic forms, and lemmas. In the future, the demo will be updated to display base phrase chunks and named entities as well.

Availability A publicly available version of MADAMIRA can be found at `http://innovation.columbia.edu/technologies/cu14012_arabic-language-disambiguation-for-natural-language-processing-applications`. This version is covered by a non-commercial research-only license agreement. For commercial use, please contact `madamira@ccls.columbia.edu`.

Acknowledgment

This paper is based upon work that was partially supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-12-C-0014. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of DARPA.

7. References

Al-Sabbagh, R. and Girju, R. (2012). A supervised POS tagger for written Arabic social networking corpora. In Jancsary, J., editor, *Proceedings of KONVENS 2012*, pages 39–52. ÖGAI, September. Main track: oral presentations.

Buckwalter, T. (2004). Buckwalter Arabic Morphological Analyzer Version 2.0. LDC catalog number LDC2004L02, ISBN 1-58563-324-0.

Buckwalter, T. (2007). Issues in Arabic Morphological Analysis. In van den Bosch, A. and Soudi, A., editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Diab, M., Hacioglu, K., and Jurafsky, D. (2007). Automated Methods for Processing Arabic Text: From Tokenization to Base Phrase Chunking. In van den Bosch, A. and Soudi, A., editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Kluwer/Springer.

Diab, M., Habash, N., Rambow, O., and Roth, R. (2013). LDC Arabic Treebanks and Associated Corpora: Data Divisions Manual. Technical Report CCLS-13-02, Center for Computational Learning Systems, Columbia University.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Graff, D., Maamouri, M., Bouziri, B., Krouna, S., Kulick, S., and Buckwalter, T. (2009). Standard Arabic Morphological Analyzer (SAMA) Version 3.1. Linguistic Data Consortium LDC2009E73.

Habash, N. and Rambow, O. (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.

Habash, N. and Rambow, O. (2006). MAGEAD: A Morphological Analyzer and Generator for the Arabic Dialects. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 681–688, Sydney, Australia.

Figure 2: The MADAMIRA Online Web Demo, showing the diacritic forms of the input text. The text color codes indicate general part-of-speech class (red for verbs, green for proper nouns, black for nominals). The other tab panels show the tokenized forms, parts-of-speech without diacritics, and lemmas. Hovering over a word (as is done here) underlines it and displays a box with all the morphological features for that word.

Habash, N. and Roth, R. (2009). CATiB: The Columbia Arabic Treebank. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 221–224, Suntec, Singapore.

Habash, N., Rambow, O., and Roth, R. (2009). MADA+TOKAN: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In Choukri, K. and Maegaard, B., editors, *Proceedings of the Second International Conference on Arabic Language Resources and Tools*. The MEDAR Consortium, April.

Habash, N., Eskander, R., and Hawwari, A. (2012). A Morphological Analyzer for Egyptian Arabic. In *NAACL-HLT 2012 Workshop on Computational Morphology and Phonology (SIGMORPHON2012)*, pages 1–9.

Habash, N., Roth, R., Rambow, O., Eskander, R., and Tomeh, N. (2013). Morphological Analysis and Disambiguation for Dialectal Arabic. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Atlanta, GA.

Habash, N. (2010). *Introduction to Arabic Natural Language Processing*. Morgan & Claypool Publishers.

Kulick, S., Gabbard, R., and Marcus, M. (2006). Parsing the arabic treebank: Analysis and improvements. In *Proceedings of the 5th Conference on Treebanks and Linguistics Theories*, pages 31–32.

Maamouri, M., Bies, A., and Kulick, S. (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.

Maamouri, M., Bies, A., Kulick, S., Krouna, S., Tabassi, D., and Ciul, M. (2012). Egyptian Arabic Treebank DF Parts 1 through 6 V2.0.

Mohamed, E., Mohit, B., and Oflazer, K. (2012). Annotating and Learning Morphological Segmentation of Egyptian Colloquial Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Istanbul.

Roth, R., Rambow, O., Habash, N., Diab, M., and Rudin, C. (2008). Arabic morphological tagging, diacritization, and lemmatization using lexeme models and feature ranking. In *ACL 2008: The Conference of the Association for Computational Linguistics; Companion Volume, Short Papers*, Columbus, Ohio, June. Association for Computational Linguistics.

Waldvogel, B. (2008). Liblinear-java: http://liblinear.bwaldvogel.de.

```
<madamira_output xmlns="urn:edu.columbia.ccls.madamira.configuration:0.1">
  <out_doc id="ExampleDocument">
    <out_seg id="SENT1">
      <segment_info> <preprocessed>الأطفال أمل أمتنا ومستقبلها</preprocessed> </segment_info>
      <word_info>
        <word id="0" word="الأطفال">
          <analysis rank="0" score="0.8847645546831359">
            <morph_feature_set diac="الأَطْفالُ" lemma="طِفْل_1"    bw="Al/DET+&gt;aTofAl/NOUN+u/CASE_DEF_NOM"
                gloss="children;infants" pos="noun" prc3="0" prc2="0" prc1="0" prc0="Al_det" per="na" asp="na" vox="na"
                mod="na" gen="m" num="s" stt="d" cas="n" enc0="0" source="lex" stem="أَطْفال"/>
          </analysis>
          <tokenized scheme="ATB"> <tok id="0" form0="الأطفال"/> </tokenized>
          <tokenized scheme="D3"> <tok id="0" form0="ال+"/> <tok id="1" form0="أطفال"/> </tokenized>
        </word>
        <word id="2" word="أمل">
          <analysis rank="0" score="0.8794376636395095">
            <morph_feature_set diac="أَمَلُ" lemma="أَمَل_1" bw="&gt;amal/NOUN+u/CASE_DEF_NOM" gloss="hope;wish"
                pos="noun" prc3="0" prc2="0" prc1="0" prc0="0" per="na" asp="na" vox="na" mod="na" gen="m" num="s"
                stt="c" cas="n" enc0="0" source="lex" stem="أَمَل"/>
          </analysis>
          <tokenized scheme="ATB"> <tok id="0" form0="أمل"/> </tokenized>
          <tokenized scheme="D3"> <tok id="0" form0="أمل"/> </tokenized>
        </word>
        <word id="3" word="أمتنا">
         <analysis rank="0" score="0.8939132838679384">
            <morph_feature_set diac="أُمَّتِنا" lemma="أُمَّة_1"
                bw="&gt;um~/NOUN+at/NSUFF_FEM_SG+i/CASE_DEF_GEN+nA/POSS_PRON_1P"
                gloss="nation;people" pos="noun" prc3="0" prc2="0" prc1="0" prc0="0" per="na" asp="na" vox="na"
                mod="na" gen="f" num="s" stt="c" cas="g" enc0="1p_poss" source="lex" stem="أُمّ"/>
          </analysis>
          <tokenized scheme="ATB"> <tok id="0" form0="أمة"/> <tok id="1" form0="+نا"/> </tokenized>
          <tokenized scheme="D3"> <tok id="0" form0="أمة"/> <tok id="1" form0="+نا"/> </tokenized>
        </word>
        <word id="4" word="ومستقبلها">
          <analysis rank="0" score="0.8939507088636945">
            <morph_feature_set diac="وَمُسْتَقْبَلِها" lemma="مُسْتَقْبَل_1"
                bw="wa/CONJ+musotaqobal/NOUN+i/CASE_DEF_GEN+hA/POSS_PRON_3FS" gloss="future"
                pos="noun" prc3="0" prc2="wa_conj" prc1="0" prc0="0" per="na" asp="na" vox="na" mod="na" gen="m"
                num="s" stt="c" cas="g" enc0="3fs_poss" source="lex" stem="مُسْتَقْبَل"/>
          </analysis>
          <tokenized scheme="ATB">
            <tok id="0" form0="و+"/> <tok id="1" form0="مستقبل"/> <tok id="2" form0="+ها"/> </tokenized>
          <tokenized scheme="D3">
            <tok id="0" form0="و+"/> <tok id="1" form0="مستقبل"/> <tok id="2" form0="+ها"/> </tokenized>
        </word>
      </word_info>
    </out_seg>
  </out_doc>
</madamira_output>
```


Figure 3: MADAMIRA output for the MSA input الأطفال أمل أمتنا ومستقبلها *Al>TfAl >ml >mtnA wmstqblhA* ‘Children are the hope of our nation and its future’.

```xml
<madamira_output xmlns="urn:edu.columbia.ccls.madamira.configuration:0.1">
  <out_doc id="ExampleDocument">
    <out_seg id="SENT1">
      <segment_info> <preprocessed>انا مابذاكرش اليومين دول</preprocessed> </segment_info>
      <word_info>
        <word id="0" word="انا">
          <analysis rank="0" score="0.8321365890115302">
            <morph_feature_set diac="انا" lemma="انا_1" bw="AnA/PRON_1S" gloss="I;me" pos="pron"
              prc3="0" prc2="0" prc1="0" prc0="0" per="1" asp="na" vox="na" mod="na" gen="m" num="s" stt="i" cas="n"
              enc0="0" enc1="0" enc2="0" source="lex" stem="انا"/>
          </analysis>
          <tokenized scheme="ATB_BWFORM"> <tok id="0" form0="انا"/> </tokenized>
          <tokenized scheme="D3_BWFORM"> <tok id="0" form0="انا"/> </tokenized>
        </word>
        <word id="1" word="مابذاكرش">
          <analysis rank="0" score="0.8780958428941577">
             <morph_feature_set diac="ما بأَذاكِرش" lemma="ذاكِر_1"
              bw="mA/NEG_PART+bi/PROG_PART+Aa/IV1S+*Akir/IV+$/NEG_PART" gloss="review;revise;study"
              pos="verb" prc3="0" prc2="0" prc1="bi_prog" prc0="mA_neg" per="1" asp="i" vox="a" mod="i" gen="m"
              num="s" stt="na" cas="na" enc0="0" enc1="0" enc2="part_neg" source="spvar" stem="ذاكِر"/>
          </analysis>
          <tokenized scheme="ATB_BWFORM">
             <tok id="0" form0="ما+"/> <tok id="1" form0="ب+"/> <tok id="2" form0="اذاكر"/> <tok id="3" form0="+ش"/>
          </tokenized>
          <tokenized scheme="D3_BWFORM">
          <tok id="0" form0="ما+"/> <tok id="1" form0="ب+"/> <tok id="2" form0="اذاكر"/> <tok id="3" form0="+ش"/>
          </tokenized>
        </word>
        <word id="2" word="اليومين">
          <analysis rank="0" score="0.9107984495293612">
             <morph_feature_set diac="اليُومَين" lemma="يَوم_1" bw="Al/DET+yuwm/NOUN+ayn/NSUFF_MASC_DU"
              gloss="today;day;some_day;ever;days" pos="noun"  prc3="0" prc2="0" prc1="0" prc0="Al_det" per="na"
              asp="na" vox="na" mod="na" gen="m" num="d" stt="d" cas="u" enc0="0" enc1="0" enc2="0" source="lex"
              stem="يُوم"/>
          </analysis>
          <tokenized scheme="ATB_BWFORM"> <tok id="0" form0="اليومين"/> </tokenized>
          <tokenized scheme="D3_BWFORM"> <tok id="0" form0="ال+"/> <tok id="1" form0="يومين"/> </tokenized>
        </word>
        <word id="3" word="دول">
          <analysis rank="0" score="0.9108938228388679">
             <morph_feature_set diac="دَوْل" lemma="دَوْل_1" bw="dawol/DEM_PRON_P" gloss="those;these"
              pos="pron_dem" prc3="0" prc2="0" prc1="0" prc0="0" per="na" asp="na" vox="na" mod="na" gen="m"
              num="p" stt="i" cas="u" enc0="0" enc1="0" enc2="0" source="lex" stem="دَوْل"/>
          </analysis>
          <tokenized scheme="ATB_BWFORM"> <tok id="0" form0="دول"/> </tokenized>
          <tokenized scheme="D3_BWFORM"> <tok id="0" form0="دول"/> </tokenized>
        </word>
      </word_info>
    </out_seg>
  </out_doc>
</madamira_output>
```


Figure 4: MADAMIRA output for the EGY input انا مابذاكرش اليومين دول *AnA mAb*Akr$ Alywmyn dwl* ‘I do not study these days’.