

# MADAMIRA v1.0 Beta User Manual

Lead Engineer: Arfath Pasha

Team: Mohammad Al-Badrashiny, Mona Diab, Nizar Habash,  
Manoj Pooleery, and Owen Rambow

Center for Computational Learning Systems  
Columbia University

November 2013

Current Version: MADAMIRA 1.0 BETA

## 1 Introduction

MADAMIRA is the combination and refinement of two valuable tools in Arabic Natural Language Processing (NLP): MADA and AMIRA. It also includes the functionality of MADA-ARZ, a version of MADA developed specifically for the Egyptian dialect.

MADA is a system for Morphological Analysis and Disambiguation for Arabic. The primary purpose of MADA is to, given raw Arabic text, derive as much linguistic information as possible about each word in the text, thereby reducing or eliminating any ambiguity surrounding the word. MADA does this by using ALMOR (an Arabic lexeme-based morphology analyzer) to generate every possible interpretation (or *analysis*) of each input word. MADA then applies a number of language models to determine which analysis is the most probable for

Please note: When citing MADAMIRA in your own publications, please be sure to include the version number you used. This is important because different versions can produce significantly different results, and therefore the version must be considered when comparing to previous work.
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

each word, given the word's context. MADAMIRA also includes TOKAN, a general tokenizer for MADA-disambiguated text. TOKAN uses the information generated by the MADA component to tokenize each word according to a highly-customizable scheme.

AMIRA is a system for tokenization, part-of-speech tagging, Base Phrase Chunking (BPC) and Named Entity Recognition (NER). Although not ready for this release, MADAMIRA will eventually incorporate the BPC and NER components of AMIRA, providing information that can be highly valuable for tasks such as Arabic parsing.

MADAMIRA is designed with the goals of being a functional replacement for MADA and AMIRA, being platform-independent, and providing the ability to process Arabic text at a much faster rate than the older tools. In addition, MADAMIRA provides support for Server-client operation that will allow for processing of Arabic via HTTP. MADAMIRA is designed to process Modern Standard Arabic (MSA) and, in this version, most of its functionality is also available for the Egyptian (EGY) dialect as well.

This document is divided into several sections. Users interested in getting MADAMIRA operating quickly should examine Sections 2-6. These sections cover system requirements, obtaining and installing MADAMIRA, and instruction on how to run the system. Section 2 gives notes about this specific release and lists known issues. The remaining sections and appendices go into finer detail. Section 7 discusses the MADAMIRA configuration options. Section 8 provides an evaluation of MADAMIRA's time and accuracy performance for several metrics. Appendix A provides some example HTTP client code for interfacing with the MADAMIRA server. Appendix B lists the data sources used to train and develop MADAMIRA. Appendix C describes the features used by MADAMIRA and what output values they can have.

## 2 About this Release

### 2.1 Functionality Highlights

- MADAMIRA 1.0 BETA provides high-quality word-level disambiguation of Arabic text, and can conduct the following, valuable NLP tasks for Modern Standard Arabic or the Egyptian dialect words:
  - **Lemmatization** determining the lemma
  - **Diacritization** determining the fully diacritized form

- **Glossing** determining the English glossary entry
  - **Part-of-speech Tagging** determining the part-of-speech
  - **Morphological Analysis** determining every possible morphological interpretation of input words
  - **Full Morphological Disambiguation** determining a complete or partial set of morphological features (either the most likely feature values for each word given its context, or a ranked list of possible all possible analyses for each word)
  - **Stemming** the reduction of each word to its morphological stem
  - **Tokenization** segmentation of clitics with attendant spelling adjustments according a variety of schemes (the tokenization scheme specifies the tokenization separation rules and the output format)
- MADAMIRA 1.0 BETA is configurable to allow users to specify what information is required and control its output format and representation
  - MADAMIRA 1.0 BETA is based in Java and is platform-independent
  - MADAMIRA 1.0 BETA supports a Server-client operation mode that allows for Arabic processing via HTTP and interfacing with other web-based tools
  - MADAMIRA 1.0 BETA is considerably faster than the older versions of MADA for MSA and Egyptian. In terms of words processed per second, MADAMIRA 1.0 BETA in Server-client mode can achieve processing rates 25 times faster than MADA v3.2 for MSA, and 19 times faster than MADA-ARZ v0.4 for Egyptian. See Section 8 for a full comparison.

## 2.2 Known Issues

- The default memory requirements for MADAMIRA are significant (ideally 2.5 GB of Java heap space, to include both MSA and EGY resources, 1.5GB if running in MSA-only or EGY-only mode). Future versions will attempt to reduce the memory requirements.
- MADAMIRA currently does not support all the tokenization options provided by MADA 3.2; much if not all of this functionality may be added in future versions.
- Most of the MSA tokenization schemes are currently not supported for the Egyptian dialect; these are topics of active research.
- MADAMIRA 1.0 BETA does not include *CODAFy* (a component which attempts to enforce certain orthography conventions for dialectal Egyptian), which was packaged with MADA-ARZ v0.4. *CODAFy* or a similar functionality will be added in future versions.

## 3 System Requirements

MADAMIRA is built using Java and can be used on Windows, Mac or Unix/Linux machines. Using MADAMIRA requires:

- **Java 1.7 or later**
- **LDC's Standard Arabic Morphological Analyzer (SAMA) version 3.1** MADAMIRA makes use of the library files included with the LDC's SAMA tool. These files are modified for use within MADAMIRA and are included only in the standard releases. They are not included in 'unbundled' releases (see Installation section). Due to licensing restrictions any user of any release of MADAMIRA must also have a license to use the SAMA tool (version 3.1, catalog number LDC2009E73) from the Linguistic Data Consortium (LDC)<sup>1</sup> before they can legally use MADAMIRA.
- **2.5GB of RAM memory** 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 EGY-only mode. We hope to reduce the memory requirements in future releases.

## 4 Obtaining MADAMIRA

MADAMIRA can be obtained by logging into the NADIA Project Wiki:

*<https://power.ldeo.columbia.edu/twiki/bin/view/NLP/Nadia#Deliverables>*

If you do not have a user login for the NADIA Project site, please contact Manoj Pooleery (manoj@ccls.columbia.edu).

## 5 Installation

**For the standard version** of MADAMIRA, (version tagged with **-all.zip**) installation only requires unpacking the supplied .zip archive onto your machine.

**For the unbundled version** of MADAMIRA (version tagged with **-unbundled.zip**), the installation requires unpacking the supplied .zip onto your machine, and the creation of the SAMA and/or CALIMA databases using instructions in the Perl script located at resources/INSTALL.pl (see System Requirements section). This process requires a Perl interpreter (with version 5.x or greater) to be present on the system MADAMIRA is being installed on.

---

<sup>1</sup><http://www ldc.upenn.edu/>

**For the partially unbundled versions** of MADAMIRA (versions tagged with `-msa.zip` or `-egy.zip`), the installation only requires unpacking the supplied `.zip` onto your machine. The `-msa` version includes the SAMA database and allows the user to process MSA (Modern Standard Arabic) input only. Similarly, the `-egy` version includes the CALIMA database and allows the user to process EGY (Egyptian) input only. If the user of these versions is interested in processing, say EGY input on the `-msa` version, or MSA input on the `-egy` version, then the user must follow the instructions in the previous paragraph for the `-unbundled` version to include the appropriate database.

Additionally, users may adjust the default MADAMIRA configuration to suit their needs (see Section 7 for details).

## 6 Running MADAMIRA

MADAMIRA provides two standard modes of operation: Stand-alone, and Server-client mode.

### 6.1 Stand-alone mode

This mode is similar to the methods used in the original MADA and AMIRA systems. Here, MADAMIRA initializes and loads its models and then processes a signal input file, after which it shuts down. This mode will tend to have a longer run time than Server-client, because of the overhead involved in loading the models and initializing the system (in Server-client mode, this work is done once when the server is started). The Stand-alone mode does not involve HTTP communication.

In Stand-alone mode, there are two ways of passing data into MADAMIRA and receiving results: XML and raw. XML I/O is the default method; with it users wrap the data they wish MADAMIRA to process in a XML file (optionally including sections to override the default system configuration), and receive the results in another XML file. The output XML structure will only include information that the user (through the configuration settings) has explicitly requested. The format of the XML is described in Section 6.3; the files *exampleInputFile.xml* and *exampleOutputFile.xml* in the *resources/schema/samples/* directory are examples of the required structures. The following command uses MADAMIRA in Stand-alone mode with XML I/O:

```
java -Xmx2500m -Xms2500m -XX:NewRatio=3    \\
-jar <location of the MADAMIRA .jar file> \\
-i inputData.xml -o outputData.xml
```

Users can also include as command-line options **-msaonly** or **-egyonly**, which will instruct the system to only load MSA or EGY resources, respectively. This command will read the file *inputData.xml*, process its contents, and place the output in a file name *outputData.xml*. It also sets the Java maximum heap size to 2.5GB (recommended for the default operation; running in MSA-only or EGY-only mode only requires 1.5GB of heap space).

The second Stand-alone I/O method is to use raw data. This method simulates the input and output file formats used in the original MADA system, and allows MADAMIRA to be used without wrapping input data in XML. The following command uses MADAMIRA in Stand-alone mode with raw I/O:

```
java -Xmx2500m -Xms2500m -XX:NewRatio=3    \\
-jar <location of the MADAMIRA .jar file> \\
-rawinput inputData.txt                    \\
-rawoutdir <output directory>            \\
-rawconfig rawConfig.xml
```

The **-msaonly** or **-egyonly** option is also allowed here to restrict what resources are loaded. The **-rawconfig** argument is optional; it allows users to override the default configuration settings for the current processing run. The *rawConfig.xml* file follows a similar format as the input XML file discussed in Section 6.3; the file *resources/schema/samples/exampleConfigFile.xml* is an example of the required structure. The **-rawinput** argument specifies the input text to process; *inputData.txt* file should contain raw Arabic text, one-sentence-per-line, with no metadata or other markup tags. The **-rawoutdir** argument specifies where to place the output files; the configuration determines what files are produced, but each filename will be set equal to the input file name plus a type extension. Possible output files include:

- **\*.bw** The output of the preprocessor, in *Buckwalter*. Buckwalter is an ASCII transliteration of Arabic text that is an industry standard.<sup>2</sup>
- **\*.mada** The output of the MADA ranker, which lists all the analyses of every word, scored and sorted. The Arabic in this file is also shown in Buckwalter.

---

<sup>2</sup>described here: <http://www.qamus.org/transliteration.htm>

- **\*.tok** The output of the tokenizer; MADAMIRA will produce one of these for each requested tokenization scheme.

## 6.2 Server-client mode

The MADAMIRA server can be started with the following command:

```
java -Xmx2500m -Xms2500m -XX:NewRatio=3    \\  
-jar <location of the MADAMIRA .jar file> \\  
-s
```

The **-msaonly** or **-egyonly** option is also allowed here to restrict what resources are available to the server. On a Windows platform, it is recommended to set up the server as a service. To stop HTTP server, type "stop" and press "Enter" key. Note that it may take a few seconds for the server to shutdown.

The included MADAMIRA client can be run (once the server has started) using the command, which will process the *inputData.xml* file and place the results in *outputData.xml*:

```
java -jar <location of the MADAMIRA .jar file> \\  
-c -i inputData.xml -o outputData.xml
```

The XML in the input and output files follows the same structure described for Stand-alone mode XML I/O; using raw I/O in Server-client mode is currently not supported.

### 6.2.1 Client Integration

The integration point for MADAMIRA in a production environment is the built-in HTTP server. In order for an external program or script to integrate with MADAMIRA, a corresponding HTTP client is needed. One such client is provided with MADAMIRA. This client embeds the input XML into the HTTP message body and send a POST request to the HTTP server. The client then waits for and receives the HTTP response and parses the output XML that is embedded in the received message body. The format of the input and output XML used by the server and client is specified in the RelaxNG<sup>3</sup> *resources/schema/MADAMIRA.rnc* file; the corresponding DTD and XSD schemas provided in the same directory.

---

<sup>3</sup><http://www.relaxng.org/compact-tutorial-20030326.html>

Attribute:Value	Description	Required
Host:localhost	Domain name or IP address of the web server.	Yes
Port:8082	Port the HTTP server is listening to. This number is specified in config/madamira.properties and can be changed.	Yes
Connection Timeout:1000	Connection Timeout. Number of milliseconds to wait for a connection to open.	No
Response Timeout:3000	Response Timeout. Number of milliseconds to wait for a response.	No
Method:POST	HTTP POST	Yes
Content Encoding:UTF8	Content encoding to be used for POST. This is the character encoding to be used, and is not related to the Content-Encoding of the HTTP header.	Yes
Content-Type: application/xml	The MIME type of the body of the request	No
Connection:Keep-Alive	Maintain a persistent connection	No
Transfer-Encoding:Chunked	Data is sent in a series of "chunks"	No

Table 1: Client Parameter Values: these are suggestions for any HTTP Client written to interact with the MADAMIRA server, and whether or not those parameters are required for proper operation.

For reference, Appendix A contains Java code for a simple HTTP client implementation. This implementation uses the Apache HTTP Client API<sup>4</sup>. Similar implementations may be made in other programming languages like C# by using HTTP Client APIs built for those languages. The suggested values for HTTP parameters are listed in Table 1.

### 6.3 Input/Output Formats

The XML I/O file formats for MADAMIRA are in XML. The normative specification for this XML is located in the file *resources/schema/MADAMIRA.rnc*. This specification is written in the RelaxNG Compact format<sup>5</sup>. The corresponding DTD and XSD specifications at the same directory location may also be referenced.

An input XML file can optionally contain a **madamira\_configuration** element, which can be used to override the configuration options for the processing of the that input file. The **madamira\_configuration** element can also be used, by itself, to specify a configuration to be used in Stand-alone, raw I/O mode. An example of a configuration XML is the system default configuration found under *resources/schema/samples/exampleConfigFile.xml*. Table 2 shows the dif-

<sup>4</sup><http://hc.apache.org/httpclient-3.x/>

<sup>5</sup><http://www.relaxng.org/compact-tutorial-20030326.html>



ferent configuration options, and Section 7 provides an more detailed explanation. Apart from the configuration elements, the input XML structure definition contains an element for specifying the the entire document (**in\_doc**), and an element for specifying each input sentence/segment and its ID (**in\_seg**). The file *resources/schema/samples/exampleInputFile.xml* is provided as an example of valid XML input.

The output XML format only contains structures to present processed output; only the information specified by the **requested\_output** elements of the configuration are presented. If requested, the **preprocessed** element shows the output of the MADAMIRA preprocessor component on the original sentence (in Buckwalter). For each input word, requested information is presented in a **word:analysis:morph\_feature\_set** XML element hierarchy, with specific information included as element attributes. Tokenized forms of the input words are also included under the **word** element, with one **tokenized** element included for each required tokenization method (*scheme*). The file *exampleOutputFile.xml* in the *resources/schema/samples/* directory shows an example of the output XML structure, produced from the *exampleInputFile.xml* in the same directory.

## 7 Configuration Details

MADAMIRA is configured using options organized into an XML structure. The default application configuration is located at *config/madamira.xml*. This contains the default configuration settings MADAMIRA uses when specific configuration settings are not provided by the user. Users can find the specification for this file in *resources/schema/MADAMIRA.rnc*, under the **madamira\_configuration** element description. This file is written in the RelaxNG Compact format<sup>6</sup>. The corresponding DTD and XSD specifications at the same directory location may also be referenced. The sub-elements that make up the **madamira\_configuration** element that makes up the default configuration *config/madamira.xml* are described in Table 2.

In addition to the system configuration files, users can, if desired, override all or some of the system configuration defaults by including configuration information with the data they wish to process, by including a **madamira\_configuration** element in their input XML file (see Section 6.3, and the example file file *resources/schema/samples/exampleInputFile.xml*). Table 2 shows the various con-

---

<sup>6</sup><http://www.relaxng.org/compact-tutorial-20030326.html>

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	These specify what information is requested. The accompanying value attribute, if set to true, indicates that this variable is required. See Tables 5-9 for descriptions of possible values.
tokenization: scheme	alias	D1   D2   D3   ATB   ATB4MT   D34MT   ATB_BWFORM   D3_BWFORM   D3_BWPOS   ATB_EVAL   ATB_BWFORM_EVAL	These specify what token schemes are to be used. See Table 3 for details of each alias.

Table 2: Options that can be specified in the MADAMIRA configuration, in the XML **madamira\_configuration** element.

figuration options that can be set; they fall into the following categories:

- **Sentence IDs** If set to *true*, the first word of every input sentence will be taken as the sentence ID and not processed as a word. This is mainly used for raw (non-XML) input; XML input can specify the sentence IDs as XML attributes.
- **Punctuation and Number Separation** If set to *true*, punctuation and digits/numbers will be white-space separated from words they are adjacent to, and are treated as separate words by the system. This is always recommended as it improves the internal model accuracy.
- **Input and Output Encoding/Transliteration** These settings control the form of the input and output text. *UTF8* specifies Arabic in UTF-8 encoding. *Buckwalter* is an ASCII transliteration of Arabic text that is an industry standard.<sup>7</sup> *SafeBW* is a modification of Buckwalter that uses only alphabetical characters (no punctuation marks) in its transliteration, to avoid confusion with reserved characters. Note that this variable does not control the output transliteration of the tokenized forms – there the transliteration is determined by the tokenization alias used.
- **MSA or Dialect** This setting indicates whether the input data should be handled as MSA or Egyptian (EGY).
- **Analyses Presented** If set to *TOP*, only data associated with the top-scoring morphological interpretation (*analysis*) of each word will be included in the output. A setting of *ALL* will show data associated with all the possible analyses of each word (and will result in a much larger output file). In most cases, *TOP* is the preferred setting.
- **Morphological Backoff** This setting relates to the morphological analyzer, and tell that component whether to use a secondary (backoff) method to create additional analyses for some or all of the input words. There are five possible settings. *NONE* is the preferred default and indicates that backoff analyses should not be included. *NOAN\_PROP* and *ADD\_PROP* tell the analyzer to add proper noun interpretations to words without any analyses or to all words, respectively. *NOAN\_ALL* and *ADD\_ALL* tell the analyzer to add several extra, simple interpretations to to words without any analyses or to all words, respectively.
- **Analyze Only** This setting, if true, causes MADAMIRA to construct and output an un-ranked analysis list for each input word and then stop, without applying feature models, ranking or tokenization. This setting overrides relevant **Requested Output**, **Requested Tokenization** and other settings. Using this mode is equivalent to running the ALMOR morphological analyzer on the preprocessed data and then stopping.
- **Requested Output** These variables allow the user to specify what output is produced. Users can request the original sentence after passing through the MADAMIRA preprocessor (*PREPROCESSED*, always in Buckwalter). This output is presented in its own element in the output XML file.

In addition, the user can request the following for each analysis of each word in the input: the word lemma (*LEMMA*), the word with diacritic markers (*DIAC*), the English gloss

---

<sup>7</sup>described here: <http://www.qamus.org/transliteration.htm>

(*GLOSS*), aspect (*ASP*), case (*CAS*), gender (*GEN*), mood (*MOD*), number (*NUM*), person (*PER*), part-of-speech (*POS*), state (*STT*), voice (*VOX*), the word stem (*STEM*), the source of the word analysis (*SOURCE*), the full Buckwalter tag (*BW*), the word proclitic values (*PRC3*, *PRC2*, *PRC1*, *PRC0*), and the word enclitic values (*ENC0*, *ENC1*, *ENC2*). The possible values these features can take are described in Tables 5-9. The user can also request that the word length (*LENGTH*) be explicitly printed. These features and their values will appear as attributes in each output analysis XML element.

- **Requested Tokenization Schemes** MADAMIRA currently provides 11 different ways (*schemes*) for tokenizing input text, each specified with an alias term (listed in Table 2). 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, MADAMIRA 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 Egyptian dialect.

Table 3 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 representative character or string; the most common case is normalizing Arabic alef and yaa 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. Multiple forms of the same token are usually connected using form delimiter character (the default is the Unicode middle-dot character "u00b7"). A token scheme with more than one form is sometimes called a *multi-tier scheme*.

Alias	Description
ATB	Tokenizes all clitics except for the definite article, normalizes alefs/yaa, uses '+' as clitic markers, and normalizes '(' and ')' characters to "-LRB-" and "-RRB-". Only one form; outputs tokens in UTF-8.
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. Can be used with EGY.
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. Can be used with EGY.
ATB4MT	A large scheme consisting of 6 forms (also referred to as a "6-tier" scheme). Tokenizes the same clitics as ATB. Form 0 is the basic token form, without normalizations. Form 1 is the same, but it also normalizes alefs/yaas; Form 2 is the token/word lemma, using '+' clitic markers; Forms 3, 4, and 5 are the part-of-speech tags in the CATiB, Penn ATB and Buckwalter POS tagsets, respectively. The Arabic portions of this scheme are presented in UTF-8.
D1	Tokenizes QUES and CONJ proclitics only ; uses '+' as a clitic marker, normalizes alefs/yaas, and normalizes '(' and ')' characters to "-LRB-" and "-RRB-". Only one form; outputs tokens in UTF-8.
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. Can be used with EGY.
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 alef/yaas. The second form is the Buckwalter part-of-speech tag.
D34MT	Another large 6-form (6-tier) scheme. Effectively the same as ATB4MT, except that all the clitics are tokenized.

Table 3: Tokenization Alias descriptions. Currently only the aliases containing in "BWFORM" can be used with the Egyptian (EGY) dialect. See Tables 7-9 for lists of clitics and descriptions.

## 8 Evaluating MADAMIRA

To evaluate MADAMIRA, the **ATB3-Test** data set (described in Appendix B, about 25K words) was run through MADAMIRA, and the result was compared to a gold, annotated version derived from the PATB3 corpora. The evaluation was conducted across several accuracy metrics (all on the word level):

- **EVALDIAC** – Percentage of words where the analysis chosen by MADA 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
- **EVALFULL** – Percentage of words where the chosen analysis has 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 the `ATB_EVAL` or `ATB_BWFORM_EVAL` tokenization scheme). Also shown are the percentage of words with correct segmentation (that is, correct number of tokens, if not correct spelling) is also shown.

Table 4 shows the accuracy and speed performance of MADAMIRA on the **PATB3-Test** test set, 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 was requested. The speed evaluation is conducted for both Stand-alone (raw input) and Server-client modes (note that there is no appreciable difference between the speed of raw input Stand-alone operation and XML input Stand-alone operation in MADAMIRA).

For the Server-client evaluations, the server was started (using a Java max heap size of 2.5GB), primed with a small test input (2-4 words), and then allowed to sit for a few minutes prior to sending it its first client input. This process ensures that all the required components are fully loaded before the speed benchmarks were taken, and gives the Java Virtual Machine time to optimize its performance. We have found that priming the server in this fashion can provide better time performance without affecting accuracy.

Table 4 shows respectable accuracy performance. MADAMIRA improves on the older systems for the tokenization task, and for all EGY metrics. The other MSA metrics show that the older systems have somewhat better performance,

Evaluation Metric	MSA		EGY	
	MADA v3.2	MADAMIRA v1.0 Beta	MADA-ARZ v0.4	MADAMIRA v1.0 Beta
<b>EVALDIAC</b>	<b>86.4</b>	86.3	83.8	<b>83.8</b>
<b>EVALLEX</b>	<b>96.1</b>	96.0	86.4	<b>87.9</b>
<b>EVALPOS</b>	<b>96.1</b>	95.9	91.8	<b>92.3</b>
<b>EVALFULL</b>	<b>84.3</b>	84.1	76.8	<b>77.8</b>
<b>EVALATBTOK</b>				
Perfect Tokenization	98.8	<b>98.9</b>	96.5	<b>96.6</b>
Correct Segmentation	99.1	<b>99.2</b>	97.4	<b>97.6</b>
<b>Speed: words/sec</b>				
Stand-alone mode	48.8	399.9	44.9	344.1
Server-client mode	N/A	<b>1512.1</b>	N/A	<b>1062.4</b>

Table 4: MADAMIRA 1.0 BETA Evaluation of MADAMIRA accuracy and speed, compared to MADA for MSA and MADA-ARZ for EGY, on a blind test set. These numbers are for the MADAMIRA build created on Nov 13, 2013. The best performing system for each metric and dialect is highlighted in **bold**.

but never by more than 0.2% (absolute). This somewhat lessened accuracy is a trade-off for the substantial speed improvement MADAMIRA provides: 1512.1 words/sec for MSA (31x faster than the older system) and 1062.4 words/sec for EGY (23x faster than the older system). EGY is generally a bit slower than MSA due to the morphological analysis step, which (in addition to being more complex for EGY), tends to generate a larger number of possible EGY analyses on average, giving MADAMIRA more to consider in subsequent steps.

Note also that, as the input is sent to the server, results of some internal computations are cached; this means that as the server is used its time performance can improve (both due to caching and to the optimizations performed by the Java Virtual Machine). These benchmarks represent performance just after the server is started and primed (that is, when the server is slowest).

## A Appendix: Example HTTPClient code

The implementation below may be used as a reference for building a HTTP client for the MADAMIRA application server.

```
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.InputStreamEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.util.EntityUtils;
import org.apache.http.conn.HttpHostConnectException;
import java.io.*;

/** A HTTP Client that reads an input XML file, sends a request to the
 * server and writes the response to an output XML file.
 */
public class HTTPClient {
    private static String url = "http://localhost:";
    private org.apache.http.client.HttpClient httpClient;
    private File iFile, oFile;
    private int PORT = 8082;

    public HTTPClient(File iFile, File oFile) {
        this.iFile = iFile;
        this.oFile = oFile;
        httpClient = new DefaultHttpClient();
    }

    /**
     * @return true if successful execution, else false.
     */
    public boolean run() {

        try {
            HttpPost httpPost = new HttpPost(url+Integer.toString(PORT));

            InputStreamEntity reqEntity =
                new InputStreamEntity(new FileInputStream(iFile), -1);
            reqEntity.setContentType("application/xml");
            reqEntity.setChunked(true);
            // It may be more appropriate to use FileEntity
            // class in this particular instance but we are using
            // a more generic InputStreamEntity to demonstrate
            // the capability to stream out data from any
            // arbitrary source

            // FileEntity entity =
            //     new FileEntity(file, "binary/octet-stream");

            httpPost.setEntity(reqEntity);
            HttpResponse response=null;
            HttpEntity resEntity=null;

            try {
                response = httpClient.execute(httpPost);
```



```

        resEntity = response.getEntity();
    } catch (HttpException ex) {
        System.out.println(ex.getMessage());
        return false;
    }

    System.out.println(response.getStatusLine());

    if (resEntity != null) {
        InputStream responseBody = response.getEntity().getContent();
        BufferedReader reader = new BufferedReader(
            new InputStreamReader(responseBody, "utf8"));
        BufferedWriter writer = new BufferedWriter(
            new OutputStreamWriter(new FileOutputStream(oFile), "utf8"));

        String line = null;
        StringBuilder sbr = new StringBuilder();
        while ((line = reader.readLine()) != null) {
            sbr.append(line+"\n");
        }
        reader.close();
        writer.write(sbr.toString());
        writer.flush();
        writer.close();
    }
    EntityUtils.consume(resEntity);

} catch (IOException ioe) {
    ioe.printStackTrace();
} finally {
    // When HttpClient instance is no longer needed,
    // shut down the connection manager to ensure
    // immediate deallocation of all system resources
    httpClient.getConnectionManager().shutdown();
}

return true;
}

// runs client standalone
public static void main(String [] args) {
    HTTPClient client =
        new HTTPClient(new File(args[0]), new File(args[1]));
    client.run();
}
}

```

## B Appendix: MADAMIRA Model Data

The current MSA models that are included with the MADAMIRA release were created using data from the Linguistic Data Consortium (LDC) Penn Arabic Treebank (PATB) corpora, parts 1 (v4.1), 2 (v3.1) and 3 (v3.2). The corresponding EGY dialectal models were created using the dialectal ARZ-ATB corpora, parts 1, 2, 3, 4 and 5 (all version 2.0).

Each corpus is divided into a development set, a training set, and a test set, consisting of about 10%, 80%, and 10% of the total corpus word volume, respectively. The corpora development divisions were then concatenated into a single development set; the train sets and test sets were similarly concatenated. In addition, tuning of the individual models was conducted by randomly selecting documents (about 10% of the total word volume) from the training set to be used as a tuning set; after tuning these documents were folded back into the main training set and the models were trained again using all the train data.

Note that, in order to compare with the previous version of MADA, for MSA it was necessary to evaluate MADAMIRA on the same, older test set that MADA was evaluated on. The documents making up this older test set (**ATB3-Test**, about 25K words) are listed below. See Section 8 for the evaluation performance numbers.

The individual document ranges that correspond to the development, training, and testing sets are listed below, along with the LDC corpora catalog numbers.

- **MSA DEV** (~63K words):  
PATB1 v4.1 (LDC2010T13): 20000715\_AFP\_ARB.0001 - 20000715\_AFP\_ARB.0073  
PATB2 v3.1 (LDC2011T09): UMAAH\_UM.ARB\_20010721-e.0018 - UMAAH\_UM.ARB\_20020217-a.0019  
PATB3 v3.2 (LDC2010T08): ANN20020115.0001 - ANN20020115.0110
- **MSA TRAIN** (~500K words):  
PATB1 v4.1 (LDC2010T13): 20000715\_AFP\_ARB.0074 - 20001115\_AFP\_ARB.0128  
PATB2 v3.1 (LDC2011T09): UMAAH\_UM.ARB\_20020224-a.0005 - UMAAH\_UM.ARB\_backissue\_34-a.0013  
PATB3 v3.2 (LDC2010T08): ANN20020215.0001 - ANN20021115.0033
- **MSA TEST** (~63K words):  
PATB1 v4.1 (LDC2010T13): 20001115\_AFP\_ARB.0129 - 20001115\_AFP\_ARB.0236  
PATB2 v3.1 (LDC2011T09):  
    UMAAH\_UM.ARB\_backissue\_34-a.0014 - UMAAH\_UM.ARB\_backissue\_40-e.0025  
PATB3 v3.2 (LDC2010T08): ANN20021115.0034 - ANN20021215.0045
- **MSA ATB3-Test** (~25K words):  
PATB3 v3.1 (LDC2008E22): ANN20021115.0068 - ANN20021215.0045
- **EGY DEV** (~21K words):  
ARZ ATB1 v2.0 (LDC2012E93): bolt-arz-NG-169-181081-14577.arz.su - bolt-arz-NG-169-181081-19026.arz.su  
ARZ ATB2 v2.0 (LDC2012E98): bolt-arz-NG-169-181081-16222.arz.su - bolt-arz-NG-169-181081-68225.arz.su  
ARZ ATB3 v2.0 (LDC2012E89): bolt-arz-DF-175-182187-572764.arz.su - bolt-arz-DF-175-182187-577973.arz.su

ARZ ARB4 v2.0 (LDC2012E99): bolt-arz-DF-175-182187-575959.arz.su - bolt-arz-DF-175-182187-581488.arz.su  
ARZ ARB5 v2.0 (LDC2012E107): bolt-arz-DF-169-181089-15751715.arz.su - bolt-arz-DF-169-181091-8751442.arz.su

- **EGY TRAIN** (~134K words):  
ARZ ATB1 v2.0 (LDC2012E93): bolt-arz-NG-169-181081-21390.arz.su - bolt-arz-NG-169-181090-38942.arz.su  
ARZ ATB2 v2.0 (LDC2012E98): bolt-arz-NG-169-181081-68287.arz.su - bolt-arz-NG-169-181090-39607.arz.su  
ARZ ATB3 v2.0 (LDC2012E89): bolt-arz-DF-175-182187-578399.arz.su - bolt-arz-NG-169-181090-40341.arz.su  
ARZ ARB4 v2.0 (LDC2012E99): bolt-arz-DF-175-182187-581658.arz.su - bolt-arz-DF-175-182192-10963633.arz.su  
ARZ ARB5 v2.0 (LDC2012E107): bolt-arz-DF-175-182185-10963619.arz.su - bolt-arz-DF-204-185979-1392879.arz.su
- **EGY TEST** (~20K words):  
ARZ ATB1 v2.0 (LDC2012E93): bolt-arz-NG-169-181090-38993.arz.su - bolt-arz-NG-169-181090-40037.arz.su  
ARZ ATB2 v2.0 (LDC2012E98): bolt-arz-NG-169-181090-39695.arz.su - bolt-arz-NG-169-181090-40322.arz.su  
ARZ ATB3 v2.0 (LDC2012E89): bolt-arz-NG-169-181090-40504.arz.su - bolt-arz-NG-169-181092-26920.arz.su  
ARZ ARB4 v2.0 (LDC2012E99): bolt-arz-DF-175-182258-1245345.arz.su - bolt-arz-NG-169-181090-40249.arz.su  
ARZ ARB5 v2.0 (LDC2012E107): bolt-arz-DF-204-185979-1393182.arz.su - bolt-arz-NG-169-181081-72955.arz.su

## C Appendix: MADAMIRA Features and Values

In this appendix the details of MADAMIRA operation are discussed: what features are used and what values they can take.

### C.1 Morphological Features

Morphological information in MADAMIRA is represented by a combination of open-class features (e.g., the word lemma, its fully diacritized form, its English gloss) and a set of 16 closed-class features (e.g., part-of-speech, gender, case). By definition, the closed-class features each have a finite set of possible values and are meant to represent a specific aspect of the word morphology.

The 16 closed-class morphological features and their possible values are summarized in Tables 5-9. Table 5 shows the basic morphology features; these are identical for both MSA and EGY.

Table 6 describes the part-of-speech (**pos**) feature. For reference, we supply the equivalent POS tag in the Penn Arabic Treebank (PATB) POS tagset and Columbia Arabic Treebank (CATiB) POS tagset.

Table 7, 8 and 9 show the MADAMIRA proclitic and enclitic features. These features are organized according to the possible location of the clitic in the word and a consideration of what clitics can co-occur, rather than the exact clitic type (such as particles). Note that **enc1** and **enc2** are only valid for the EGY dialect. The pattern these clitics can follow is:

```
[ prc3 prc2 prc1 prc0 BASEWORD enc0 enc1 enc2 ]
```

Feature	Feature Value Definition	MADAMIRA 1.0 BETA
Aspect	LABEL	<b>asp</b>
	Command	c
	Imperfective	i
	Perfective	p
	Not applicable	na
Case	LABEL	<b>cas</b>
	Nominative	n
	Accusative	a
	Genitive	g
	Not applicable	na
	Undefined	u
Gender	LABEL	<b>gen</b>
	Feminine	f
	Masculine	m
	Not applicable	na
Mood	LABEL	<b>mod</b>
	Indicative	i
	Jussive	j
	Subjunctive	s
	Not applicable	na
	Undefined	u
Number	LABEL	<b>num</b>
	Singular	s
	Plural	p
	Dual	d
	Not applicable	na
	Undefined	u
Person	LABEL	<b>per</b>
	1st	1
	2nd	2
	3rd	3
	Not applicable	na
State	LABEL	<b>stt</b>
	Indefinite	i
	Definitie	d
	Construct/Poss/Idafa	c
	Not applicable	na
	Undefined	u
Voice	LABEL	<b>vox</b>
	Active	a
	Passive	p
	Not applicable	na
	Undefined	u

Table 5: MADAMIRA feature and value definitions, with the labels used to represent them under MADAMIRA 1.0 BETA. "LABEL" indicates the identifying tag used for that feature in MADAMIRA output files. These values are used in both MSA and the EGY dialect.

	<b>POS Definition</b>	<b>MADAMIRA 1.0 BETA</b>	<b>PATB Equivalent</b>	<b>CATiB Equivalent</b>
Part-of-speech	<b>LABEL</b>	<b>pos</b>	—	—
	Nouns	noun	NN / NNS	NOM
	Number Words	noun_num	NN / NNS	NOM
		noun_quant	NN / NNS	NOM
	Proper Nouns	noun_prop	NNP / NNPS	PROP
	Adjectives	adj	JJ	NOM
		adj_comp	JJ	NOM
		adj_num	JJ	NOM
	Adverbs	adv	RB	NOM
		adv_interrog	RP	NOM
		adv_rel	WP	NOM
	Pronouns	pron	PRP	NOM
		pron_dem	DT	NOM
		pron_exclam	PRP	NOM
		pron_interrog	RP	NOM
		pron_rel	WP	NOM
	Verbs	verb	VBN / VBP / VBD	VRB or VRB-PASS
		verb_pseudo	VBN/ VBP / VBD	PRT
	Particles	part	IN	PRT
		part_dem	DT	PRT
		part_det	DT	PRT
		part_focus	IN	PRT
		part_fut	IN	PRT
		part_interrog	IN	PRT
		part_neg	RP	PRT
		part_restrict	IN	PRT
		part_verb	IN	PRT
		part_voc	IN	PRT
	Prepositions	prep	IN	PRT
	Abbreviations	abbrev	NN	PROP
	Punctuation	punc	PUNC	PNX
	Conjunctions	conj	CC	PRT
		conj_sub	CC	PRT
	Interjections	interj	UH	PRT
	Digital Numbers	digit	CD	NOM
	Foreign/Latin	latin	IN	NOM

Table 6: MADAMIRA part-of-speech definitions and the labels used to represent them under MADAMIRA 1.0 BETA, with the equivalent Penn ATB and CATiB POS tags given as reference. These tags are used in both MSA and the EGY dialect.

	Proclitic Value Definition	MADAMIRA 1.0 BETA
Proclitic 3 (AKA question proclitic or QUES)	LABEL	<b>prc3</b>
	No proclitic	0
	Not applicable	na
	Interrogative Particle > <i>a</i>	>a_ques
Proclitic 2 (AKA conjunction proclitic or CONJ)	LABEL	<b>prc2</b>
	No proclitic	0
	Not applicable	na
	Conjunction <i>fa</i>	fa_conj
	Connective particle <i>fa</i>	fa_conn
	Response conditional <i>fa</i>	fa_rc
	Subordinating conjunction <i>fa</i>	fa_sub
	Conjunction <i>wa</i>	wa_conj
	Particle <i>wa</i>	wa_part
	Subordinating conjunction <i>wa</i>	wa_sub

Table 7: MADAMIRA proclitic definitions for **prc3** and **prc2** (for MSA and EGY) and the labels used to represent them under MADAMIRA 1.0 BETA. The proclitic number refers to the location of the clitic, according to [ PRC3 PRC2 PRC1 PRO0 BASEWORD ENC0 ENC1 ENC2 ]. **enc1** and **enc2** are only used in the EGY dialect.

	Proclitic Value Definition	MADAMIRA 1.0 BETA
Proclitic 1 (AKA preposition proclitic or PART)	LABEL	<b>prc1</b>
	No proclitic	0
	Not applicable	na
	Interrogative <i>ish</i>	<i\$_interrog
	Particle <i>bi</i>	bi_part
	Preposition <i>bi</i>	bi_prep
	Progressive verb particle <i>bi</i>	bi_prog
	Preposition <i>Ea</i>	Ea_prep
	Preposition <i>EalaY</i>	EalaY_prep
	Preposition <i>fy</i>	fy_prep
	Demonstrative <i>hA</i>	hA_dem
	Future marker <i>Ha</i>	Ha_fut
	Preposition <i>ka</i>	ka_prep
	Emphatic Particle <i>la</i>	la_emph
	Preposition <i>la</i>	la_prep
	Response conditional <i>la</i>	la_rc
	Preposition <i>li</i> + preposition <i>bi</i>	libi_prep
	Empathic <i>la</i> + future marker <i>Ha</i>	laHa_emphfut
	Response conditional <i>la</i> + future marker <i>Ha</i>	laHa_refut
	Jussive <i>li</i>	li_jus
	Preposition <i>li</i>	li_prep
	Preposition <i>min</i>	min_prep
	Future marker <i>sa</i>	sa_fut
Preposition <i>ta</i>	ta_prep	
Particle <i>wa</i>	wa_part	
Preposition <i>wa</i>	wa_prep	
Vocative <i>wA</i>	wA_voc	
Vocative <i>yA</i>	yA_voc	
Proclitic 0 (AKA article proclitic or ART)	LABEL	<b>prc0</b>
	No proclitic	0
	Not applicable	na
	Demonstrative particle <i>Aa</i>	Aa_prondem
	Determiner	Al_det
	Determiner <i>Al</i> + negative particle <i>mA</i>	AlmA_detneg
	Negative particle <i>lA</i>	lA_neg
	Negative particle <i>mA</i>	mA_neg
	Negative particle <i>ma</i>	ma_neg
	Particle <i>mA</i>	mA_part
	Relative pronoun <i>mA</i>	mA_rel

Table 8: MADAMIRA proclitic definitions for **prc1** and **prc0** (for MSA and EGY) and the labels used to represent them under MADAMIRA 1.0 BETA. The proclitic number refers to the location of the clitic, according to [ PRC3 PRC2 PRC1 PRO0 BASEWORD ENC0 ENC1 ENC2 ]. **enc1** and **enc2** are only used in the EGY dialect.



	Enclitic Value Definition	MADAMIRA 1.0 BETA
Enclitics 2 (EGY only)	LABEL	<b>enc0</b>
	No enclitic	0
	Not applicable	na
	Negative particle suffix	part_neg
Enclitics 1 (EGY only)	LABEL	<b>enc0</b>
	No enclitic	0
	Not applicable	na
	1st person singular	1s_iobj
	1st person plural	1p_iobj
	2nd person plural	2p_iobj
	2nd person masculine singular	2ms_iobj
	2nd person feminine singular	2fs_iobj
	3rd person plural	3p_iobj
	3rd person masculine singular	3ms_iobj
3rd person masculine plural	3fs_iobj	
Enclitics 0 (AKA pronominals or PRON)	LABEL	<b>enc0</b>
	No enclitic	0
	Not applicable	na
	1st person plural	1p_dobj, 1p_poss, or 1p_pron
	1st person singular	1s_dobj, 1s_poss, or 1s_pron
	2nd person dual	2d_dobj, 2d_poss, or 2d_pron
	2nd person plural	2p_dobj, 2p_poss, or 2p_pron
	2nd person feminine plural	2fp_dobj, 2fp_poss, or 2fp_pron
	2nd person feminine singular	2fs_dobj, 2fs_poss, or 2fs_pron
	2nd person masculine plural	2mp_dobj, 2mp_poss, or 2mp_pron
	2nd person masculine singular	2ms_dobj, 2ms_poss, or 2ms_pron
	3rd person dual	3d_dobj, 3d_poss, or 3d_pron
	3rd person plural	3p_dobj, 3p_poss, or 3p_pron
	3rd person feminine plural	3fp_dobj, 3fp_poss, or 3fp_pron
	3rd person feminine singular	3fs_dobj, 3fs_poss, or 3fs_pron
	3rd person masculine plural	3mp_dobj, 3mp_poss, or 3mp_pron
	3rd person masculine singular	3ms_dobj, 3ms_poss, or 3ms_pron
	Vocative particle	Ah_voc
	Negative particle <i>la</i>	lA_neg
	Interrogative pronoun <i>ma</i>	ma_interrog
	Interrogative pronoun <i>mA</i>	mA_interrog
	Interrogative pronoun <i>man</i>	man_interrog
	Relative pronoun <i>man</i>	man_rel
Relative pronoun <i>ma</i>	ma_rel	
Relative pronoun <i>mA</i>	mA_rel	
Subordinating conjunction <i>ma</i>	ma_sub	
Subordinating conjunction <i>mA</i>	mA_sub	

Table 9: MADAMIRA enclitic definitions and the labels used to represent them under MADAMIRA 1.0 BETA. The enclitics associated with (person, gender, number) combinations each have three variants specifying their object relation. **\_dobj** indicates that the word is the direct object of a verb, **\_poss** indicates that the word is a possessive pronoun (typically of a nominal), and **\_pron** indicates that the word is a regular pronoun, often the object of a particle. In the EGY dialect, **\_iobj** indicates an indirect object of a verb. The clitic number refers to the location of the clitic: [ PRC3 PRC2 PRC1 PRO0 BASEWORD ENC0 ENC1 ENC2 ]. **enc1** and **enc2** are only used in the EGY dialect.

Please note: When citing MADAMIRA in your own publications, please be sure to include the version number you used. This is important because different versions can produce significantly different results, and therefore the version must be considered when comparing to previous work.

## Recommended Readings

The following is an incomplete list of publications that relate to past and current versions of MADA, AMIRA, and MADA-ARZ.

### 2012

- Nizar Habash, Ramy Eskander, Abdelati Hawwari. *A morphological analyzer for egyptian arabic*. Proceedings of the Twelfth Meeting of the Special Interest Group on Computational Morphology and Phonology, June 2012.
- Nizar Habash, Mona Diab, Owen Rambow. *Conventional Orthography for Dialectal Arabic*. Proceedings of the Language Resources and Evaluation Conference (LREC), Istanbul, 2012.

### 2011

- Nizar Y Habash, Mona T Diab, Owen C Rambow. *Conventional Orthography for Dialectal Arabic (CODA) Version 0.1–July 2011*. Center for Computational Learning Systems, Columbia University, 2011.

### 2010

- Yassine Benajiba, Imed Zitouni, Mona Diab, Paolo Rosso. *Arabic Named Entity Recognition: Using Features Extracted from Noisy Data*. Proceedings of ACL 2010, Uppsala, Sweden, July 2010.

### 2009

- Yassine Benajiba, Mona Diab, Paolo Rosso. *Arabic Named Entity Recognition: A Feature-driven Study*. Accepted In IEEE Transactions on Audio, Speech and Language Processing, 2009.
- Mona Diab. *Second Generation Tools (AMIRA 2.0): Fast and Robust Tokenization, POS tagging, and Base Phrase Chunking*. MEDAR 2nd International Conference on Arabic Language Resources and Tools, April, Cairo, Egypt, 2009.
- Nizar Habash, Ryan M Roth. *Catib: The columbia arabic treebank*. Proceedings of the ACL-IJCNLP 2009 Conference Short Papers. August, 2009.

### 2008

- Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab and Cynthia Rudin. *Arabic Morphological Tagging, Diacritization, and Lemmatization Using Lexeme Models and Feature*

*Ranking*. In Proceedings of the Conference of American Association for Computational Linguistics (ACL08), 2008.

#### 2007

- Mona Diab. *Improved Arabic Base Phrase Chunking with a new enriched POS tag set*. Proceedings of ACL workshop on Computational Approaches to Semitic languages (CASL), Prague, Czech Republic, 2007.
- Mona Diab, Mahmoud Ghoneim, Nizar Habash. *Arabic Diacritization in the Context of Statistical Machine Translation*. In Proceedings of the Machine Translation Summit (MT-Summit), Copenhagen, Denmark, 2007.
- Jakob Elming, Nizar Habash. *Combination of Statistical Word Alignments Based on Multiple Preprocessing Schemes*. In Proceedings of the North American chapter of the Association for Computational Linguistics (NAACL), Rochester, New York, 2007.
- Nizar Habash. *Arabic Morphological Representations for Machine Translation*. Book Chapter. In Arabic Computational Morphology: Knowledge-based and Empirical Methods. Editors Antal van den Bosch and Abdelhadi Souidi. Kluwer/Springer Publications, 2007.
- Nizar Habash, Owen Rambow. *Arabic Diacritization through Full Morphological Tagging*. In Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2007); Companion Volume, Short Papers. 2007.

#### 2006

- Nizar Habash, Fatiha Sadat. *Arabic Preprocessing Schemes for Statistical Machine Translation*. In Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL), New York, 2006.
- Fatiha Sadat, Nizar Habash. *Combination of Preprocessing Schemes for Statistical MT*. In Proceedings of COLING-ACL, Sydney, Australia, 2006.

#### 2005

- Nizar Habash, Owen Rambow. *Arabic Tokenization, Morphological Analysis, and Part-of-Speech Tagging in One Fell Swoop*. In Proceedings of the Conference of American Association for Computational Linguistics (ACL05), 2005.

#### 2004

- Nizar Habash. *Large Scale Lexeme Based Arabic Morphological Generation*. In Proceedings of Traitement Automatique du Langage Naturel (TALN-04). Fez, Morocco, 2004.

## **Change Log**

### **MADAMIRA-release-01162014-1.0-beta-\***

- Included the install script resources/INSTALL.pl to allow users to build new morphological databases from LDC corpora resources (See section on Installation).