MADAMIRA v2.0 User Manual

Lead Engineer: Arfath Pasha
Team: Mohammad Al-Badrashiny, Mona Diab, Nizar Habash,
Manoj Pooleery, Owen Rambow and Ryan Roth

Center for Computational Learning Systems
Columbia University

April 2015
Current Version: MADAMIRA 2.1

TOC

Contents

1 Introduction **3**

2 About this Release **3**
2.1 Functionality Highlights 3
2.2 Known Issues 4

3 System Requirements **4**

4 Obtaining MADAMIRA **5**

5 Installation **5**

6 Running MADAMIRA **5**
6.1 Stand-alone mode 6
6.2 Server-client mode 7
6.2.1 Client Integration 7
6.3 API 8
6.4 Input/Output Formats 8

7 Configuration Details **9**
7.1 Customizing Tokenization Schemes 13
7.2 ALMOR Database Configuration 20

8 Evaluating MADAMIRA **21**
8.1 BPC 22
8.2 NER 23

A Appendix: Example HTTPClient code **25**

B Appendix: MADAMIRA **Model Data** **27**
B.1 Named Entity Recognition (NER) 28
B.2 Base Phrase Chunking (BPC) 28

C Appendix: MADAMIRA **Features and Values** **29**
C.1 Morphological Features 29

D Appendix: Aramorph vs SAMA/CALIMA **35**

E Appendix: MADAMIRA **Recommended Readings** **38**

F Appendix: MADAMIRA **Change Log** **40**

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.

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 each word, given the word's context. MADA 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) - components 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 2.1 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)
  - **Base Phrase Chunking** identifying sequences of adjacent words that form syntactic phrases such as noun phrases and verb phrases.
  - **Named Entity Recognition** identifying and classifying named entities in open-domain text.

- MADAMIRA 2.1 is configurable to allow users to specify what information is required and control its output format and representation
- MADAMIRA 2.1 is based in Java and is platform-independent
- MADAMIRA 2.1 supports a Server-client operation mode that allows for Arabic processing via HTTP and interfacing with other web-based tools
- MADAMIRA 2.1 is considerably faster than the older versions of MADA for MSA and Egyptian. In terms of words processed per second, MADAMIRA 2.1 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 supports all the tokenization options provided by MADA 3.2.
- Most of the MSA tokenization schemes are currently not supported for the Egyptian dialect; these are topics of active research.
- MADAMIRA 2.1 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**.

- All release types (bundled and unbundled) of MADAMIRA come packaged with a morphological database called Aramorph (almor-msa-r13.db), which is released in accordance with GNU General Public License version 2. However, MADAMIRA's prediction accuracy is best with the SAMA (almor-msa-s31.db) database which is made available only in the bundled versions of MADAMIRA (see Appendix **??**: Aramorph vs SAMA). The SAMA database is built from **LDC's Standard Arabic Morphological Analyzer (SAMA) version 3.1**. 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)[1] before they can legally use MADAMIRA with the SAMA database. All licensed users of all releases of MADAMIRA may use MADAMIRA with the Aramorph database without restriction. A similar license free database for Egyptian (almor-cra07.db) is also provided.
- **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 from:

http://innovation.columbia.edu/technologies/cu14012_arabic-language-disambiguation-for-natural-language-processing-applications

5 Installation

Installation only requires unpacking the supplied .zip archive onto your machine.

Your MADAMIRA release comes with the Aramorph morphological databases for MSA and EGY. But, it may not come with the SAMA (for MSA) and/or CALIMA (for EGY) database with which it is known to have the best prediction accuracy (see Appendix D: Aramorph vs SAMA/CALIMA). If you possess the license to LDC's resources specified in the System Requirements Section 3, you may build the SAMA and/or CALIMA database by following the instructions in the Perl script located at resources/INSTALL.pl. This process requires a Perl interpreter (with version 5.x or greater) to be present on the system MADAMIRA is being installed on.

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.

[1] http://www.ldc.upenn.edu/

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 6.4; 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 6.4; 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.[2]
- ***.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.
- ***.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.

The server is capable of responding to concurrent client requests. The maximum allowable number of concurrent client requests may be capped by the property server.buffersize in config/madamira.properties.

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[3] *resources/schema/MADAMIRA.rnc* file; the corresponding DTD and XSD schemas provided in the same directory.

For reference, Appendix A contains Java code for a simple HTTP client implementation. This implementation uses the Apache HTTP Client API[4]. 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.

[2] described here: http://www.qamus.org/transliteration.htm

[3] http://www.relaxng.org/compact-tutorial-20030326.html

[4] http://hc.apache.org/httpclient-3.x/

Attribute:Value	Description	Required
Host:localhost	Domain name or IP address of the web server.	Yes
Port:8223	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.

6.3 API

An application programming interface is included with this release in a directory called api. The start point of the API is a class called MADAMIRAWrapper. Java objects may instantiate this class in order to invoke MADAMIRA as part of another software program. The API also contains a package with classes whose structure resembles exactly that of the XML interface of MADAMIRA.

An example Java class called APIExampleUse.java and an ant build file has been provided to show how a call to MADAMIRA can be made through this API.

6.4 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[5] . 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 3 shows the different 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,

[5]http://www.relaxng.org/compact-tutorial-20030326.html

BPC tag	Description
ADJP	Adjectival phrase
ADVP	Adverbial phrase
CONJP	Conjunctive phrase
EDITED	restarts and repetition
FRAG	Fragment
INTJ	Interjective phrase
LST	List
NAC	Not a Constituent
NP	Noun phrase
PP	Prepositional phrase
PRN	Parenthetical phrase
S	Sentence marker
SBAR	Subjunctive construction
SBARQ	Subjunctive question
VP	Verb phrase
WHADVP	Adverbial phrase "Wh" question
WHNP	Noun phrase "Wh" question
WHPP	Prepositional phrase "Wh" question

Table 2: List of possible BPC tags used in the output.

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.

For BPC XML output, each phrase is tagged with one of 18 different phrase types listed in Table 2. For NER XML output, the named entities are tagged with PER, LOC or ORG for person, location or organization respectively. If MADAMIRA is executed in raw mode, additional files are created for BPC and NER output. These files have extensions .bpc, .bpc-bio, .ner and .ner-bio. The .bpc and .ner files contain tags that match the tags in the XML output. The .bpc-bio and .ner-bio files contain additional boundary tags B, I and O for beginning of phrase or entity, inside of phrase or entity and outside of phrase or entity respectively.

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[6]. 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 3.

[6]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 | OFFSET	These specify what information is requested. The accompanying value attribute, if set to true, indicates that this variable is required. See Tables 14-18 for descriptions of possible values.
		BPC | NER	
tokenization: scheme	alias	D1 | D2 | D3 | S2 | 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 4 for details of each alias.

Table 3: Options that can be specified in the MADAMIRA configuration, in the XML **madamira_configuration** element.

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 6.4, and the example file file *resources/schema/samples/exampleInputFile.xml*). Table 3 shows the various configuration 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.[7] *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 unranked 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 (*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 14-18. The user can also request that the word length (*LENGTH*), word offset (*OFFSET*) and word type (*TYPE*) to be explicitly printed. These features and their values will appear as attributes in each output analysis XML element. The length of an output word is simply its character length. The offset value of an output word is the character distance from the beginning of the sentence to the first character of the corresponding input word. The word type specifies whether the word is an Arabic word, non-Arabic word or a no-analysis word which means the word has no valid analyses.

  The user may also request Named Entities (NER) and Base Phase Chunks (BPC).

[7] described here: http://www.qamus.org/transliteration.htm

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).
S2	Same as D3, but proclitic and enclitic delimiters are set to empty strings.
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 4: Tokenization Alias descriptions. Currently only the aliases containing in "BWFORM" can be used with the Egyptian (EGY) dialect. See Tables 16-18 for lists of clitics and descriptions.

- **Requested Tokenization Schemes** MADAMIRA currently provides 11 predefined ways (*schemes*) for tokenizing input text, each specified with an alias term (listed in Table 3). MADAMIRA also allows users to define entirely new tokenization schemes, or extend the predefined schemes to create slightly modified forms of these schemes. Creation of such custom schemes is described in Section 7.1.

  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, 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 4 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*.

7.1 Customizing Tokenization Schemes

MADAMIRA's configuration allows users to define their own tokenization schemes by either extending the predefined schemes listed in Table 4 or by creating entirely new schemes. The scheme language is a subset of the configuration schema and may be found in the resources/schema directory in RelaxNG Compact, XSD and DTD notations. The language was designed to closely resemble the configuration options in MADA. The RelaxNG representation of the language is presented below.

The `scheme_override` element must be specified in order to define a custom scheme. The user may override a predefined scheme by setting the `scheme` element's `alias` attribute to one of the predefined alias names, and then overriding its specification with the `scheme_override` element. The `scheme_override` element's `alias` attribute may be used to change the name of the overridden scheme. To define an entirely new scheme, the user must set the `scheme` element's `alias` attribute to a name that is not part of the predefined set of alias names in Table 4.

```
element tokenization {
    element scheme {
        attribute alias { "ATB" | "S2" | "D3" | "ATB4MT" | "D34MT" | "D1" | "D2" |
         "D3_BWFORM" | "ATB_BWFORM" | "D3_BWPOS" | "ATB_EVAL"  |
         "ATB_BWFORM_EVAL" | xsd:string },
        element scheme_override {
            attribute alias { xsd:string },
            attribute token_delimiter { xsd:string }?,  # Deprecated (use
                                                        # proclitic,stem and
                                                        # enclitic delimiters instead)
            attribute proclitic_delimiter { xsd:string }?,
            attribute stem_delimiter { xsd:string }?,
            attribute enclitic_delimiter { xsd:string }?,
            attribute form_delimiter { xsd:string }?,
            attribute mark_no_analysis { xsd:boolean }?,
            attribute include_non_arabic { xsd:boolean }?,
            attribute tokenize_from_BW { xsd:boolean }?,
            element split_term_spec {
                attribute term { "REST" | "PRC3" | "QUES" | "PRC2" |"CONJ" |
                 "PART" | "FUT" | "S+" | "PRC0" | "ART" | "DART" | "AL+" |
                 "NART" | "ENC0" | "PRON" }
            }*,
            element token_form_spec {
                attribute proclitic_mark { xsd:string }?,
                attribute enclitic_mark { xsd:string }?,
                attribute token_form_base { "WORD" | "POS_MADA" | "POS_BW" |
                 "POS_CATIB" | "POS_PENN" | "POS_ALMOR" | "LEXEME" | "GLOSS" |
                 "STEM" | "SURF" |  "BWFORM" |  "BWTAG" }?,
                attribute transliteration { "Buckwalter" | "UTF8" |
                  "Safe_Buckwalter" }?,
                element normalization {
                   attribute type { "ALEF" | "YAA" | "DIGIT" | "LEFTPAREN" |
                    "RIGHTPAREN" | "TEHMARBUTA" | "HAMZA" | "PAREN" | "LEX" |
                    "DIAC" }
                }*
            }*
        }?
    }*
}
```

The attributes of the `scheme_override` element affect the entire scheme. These attributes are described in Table 5. The `scheme_override` element has two child elements - `split_term_spec` and `token_form_spec`. `split_term_spec` controls how the input word is broken up, such

as breaking off conjunctions or particles. Table 6 provide descriptions for the possible values of the `term` attribute. `token_form_spec` controls how the different tokens are output, including their arrangement, content and encoding method. The attribute `proclitic_mark` specifies the trailing characters for the proclitics, while the attribute `enclitic_mark` specifies the leading characters for the enclitics. The default values for both `proclitic_mark` and `enclitic_mark` is no characters (""). Table 7 lists descriptions for the possible values of the `token_form_base` attribute. The transliteration attribute overrides the base setting and sets the transliteration for all forms in the scheme. The `normalization` attribute specifies the various normalization rules that may be applied to the token forms. All the normalization rules except diacritization (DIAC) are set to false by default.

Attribute	**Description**
`token_delimiter`	(**Deprecated**. Use proclitic/stem/enclitic delimiters instead). If a scheme requires tokens (or a subset) to be grouped, this is the character(s) that will join them for all forms. The default delimiter is " ".
`proclitic_delimiter`	Proclitic delimiter. If a scheme requires proclitic tokens (or a subset) to be grouped, this is the character(s) that will join them for all forms. The default delimiter is " ".
`stem_delimiter`	Stem delimiter. This is the character that will join the group of proclitic and enclitic tokens with the stem. The default delimiter is " ".
`enclitic_delimiter`	Enclitic delimiter. If a scheme requires enclitic tokens (or a subset) to be grouped, this is the character(s) that will join them for all forms. The default delimiter is " ".
`form_delimiter`	Form Delimiter. If more than one form is defined, these will be used to separate the different forms in the output. The default is the middle dot character · (Unicode #00B7).
`mark_no_analysis`	If true, this will cause no-analysis words to be marked in the tokenized output. The default value is false.
`include_non_arabic`	If true, this will cause non-Arabic words to be included in the tokenized output. The default value is true.
`tokenize_from_BW`	If true, this will cause the tokenization to be developed from the Buckwalter term. The default value is false.

Table 5: scheme_override element's attributes

term value	Description
QUES or prc3	**prc3** - The 'question' proclitic
CONJ or prc2	**prc2** - The 'conjunction' proclitic
PART or prc1	**prc1** - The 'preposition' proclitic
ART or prc0	**prc0** - The 'article' proclitic
PRON or enc0	**enc0** - Enclitics
FUT or s+	The future marker clitic only (s)
DART or Al+	The definite article only (Al)
NART	The negative articles only (lA, mA)
REST	The remainder of the word after the clitics have been removed

Table 6: term values and their description

token_form_base value	Description
WORD	This simply says that this form will be displaying some version of the token itself (original, normalized, etc.) It is the most common BASE.
LEXEME	This says that this form will be displaying lexeme information for the token.
GLOSS	This form will display the gloss term provided by ALMOR.
STEM	This form will display the Buckwalter tag provided by ALMOR.
SURF	This form uses the original word form.
POS:ALMOR, POS:CATIB, POS:MADA, POS:PENN, POS:BW	These keys indicate that the form will display part-of-speech, using one of five different POS tagsets (ALMOR, CATiB, MADA, Penn ATB, or Buckwalter).

Table 7: token_form_base values and their description

The examples below provide reverse engineered specifications for some of the predefined schemes.

```
<tokenization>
        <!-- Same as ATB_EVAL-->
        <scheme alias="MyATB_EVAL">
            <scheme_override alias="MyATB_EVAL"
                             form_delimiter="\u00B7"
                             include_non_arabic="true"
                             mark_no_analysis="false"
                             proclitic_delimiter="_"
                             stem_delimiter="_"
                             enclitic_delimiter="_"
                             tokenize_from_BW="false">
                <split_term_spec term="PRC3"/>
                <split_term_spec term="PRC2"/>
                <split_term_spec term="PART"/>
                <split_term_spec term="NART"/>
                <split_term_spec term="REST"/>
                <split_term_spec term="ENC0"/>
                <token_form_spec enclitic_mark="+"
                                 proclitic_mark="+"
                                 token_form_base="WORD"
                                 transliteration="Buckwalter">
                    <normalization type="ALEF"/>
                    <normalization type="YAA"/>
                    <normalization type="LEFTPAREN"/>
                    <normalization type="RIGHTPAREN"/>
                    <normalization type="DIAC"/>
                </token_form_spec>
            </scheme_override>
        </scheme>
        <!-- Same as ATB_BWFORM_EVAL-->
        <scheme alias="MyATB_BWFORM_EVAL">
            <scheme_override alias="MyATB_BWFORM_EVAL"
                             form_delimiter="\u00B7"
                             include_non_arabic="true"
                             mark_no_analysis="false"
                             proclitic_delimiter="_"
                             stem_delimiter="_"
                             enclitic_delimiter="_"
                             tokenize_from_BW="true">
                <split_term_spec term="PRC3"/>
                <split_term_spec term="PRC2"/>
                <split_term_spec term="PART"/>
                <split_term_spec term="NART"/>
                <split_term_spec term="REST"/>
                <split_term_spec term="ENC0"/>
                <token_form_spec enclitic_mark="+"
                                 proclitic_mark="+"
                                 token_form_base="WORD"
                                 transliteration="Buckwalter">
                    <normalization type="ALEF"/>
                    <normalization type="YAA"/>
                    <normalization type="DIAC"/>
                    <normalization type="LEFTPAREN"/>
                    <normalization type="RIGHTPAREN"/>
                </token_form_spec>
            </scheme_override>
        </scheme>
        <scheme alias="MyD3">
            <!-- Same as D3 -->
            <scheme_override alias="MyD3"
                             form_delimiter="\u00B7"
                             include_non_arabic="true"
                             mark_no_analysis="false"
                             proclitic_delimiter=" "
```

```
                    stem_delimiter=" "
                    enclitic_delimiter=" "
                    tokenize_from_BW="false">
        <split_term_spec term="PRC3"/>
        <split_term_spec term="PRC2"/>
        <split_term_spec term="PART"/>
        <split_term_spec term="PRC0"/>
        <split_term_spec term="REST"/>
        <split_term_spec term="ENC0"/>
        <token_form_spec enclitic_mark="+"
                         proclitic_mark="+"
                         token_form_base="WORD"
                         transliteration="UTF8">
            <normalization type="ALEF"/>
            <normalization type="YAA"/>
            <normalization type="DIAC"/>
            <normalization type="LEFTPAREN"/>
            <normalization type="RIGHTPAREN"/>
        </token_form_spec>
    </scheme_override>
</scheme>
<!-- Same as ATB4MT -->
<scheme alias="MyATB4MT">
    <scheme_override alias="MyATB4MT"
                    form_delimiter="Âů"
                    include_non_arabic="true"
                    mark_no_analysis="false"
                    proclitic_delimiter=" "
                    stem_delimiter=" "
                    enclitic_delimiter=" "
                    tokenize_from_BW="false">
        <split_term_spec term="PRC3"/>
        <split_term_spec term="PRC2"/>
        <split_term_spec term="PART"/>
        <split_term_spec term="NART"/>
        <split_term_spec term="REST"/>
        <split_term_spec term="ENC0"/>
        <!-- form0 -->
        <token_form_spec enclitic_mark="+"
                         proclitic_mark="+"
                         token_form_base="WORD"
                         transliteration="UTF8">
            <normalization type="DIAC"/>
            <normalization type="LEFTPAREN"/>
            <normalization type="RIGHTPAREN"/>
        </token_form_spec>
        <!-- form1 -->
        <token_form_spec enclitic_mark="+"
                         proclitic_mark="+"
                         token_form_base="WORD"
                         transliteration="UTF8">
            <normalization type="ALEF"/>
            <normalization type="YAA"/>
            <normalization type="DIAC"/>
            <normalization type="LEFTPAREN"/>
            <normalization type="RIGHTPAREN"/>
        </token_form_spec>
        <!-- form2 -->
        <token_form_spec enclitic_mark="+"
                         proclitic_mark="+"
                         token_form_base="LEXEME"
                         transliteration="UTF8">
            <normalization type="PAREN"/>
            <normalization type="LEX"/>
        </token_form_spec>
        <!-- form3 -->
        <token_form_spec enclitic_mark="+"
                         proclitic_mark="+"
```

```
                        token_form_base="POS_CATIB"
                        transliteration="UTF8">
                <normalization type="DIAC"/>
            </token_form_spec>
            <!-- form4 -->
            <token_form_spec enclitic_mark="+"
                        proclitic_mark="+"
                        token_form_base="POS_PENN"
                        transliteration="UTF8">
                <normalization type="DIAC"/>
            </token_form_spec>
            <!-- form5 -->
            <token_form_spec enclitic_mark="+"
                        proclitic_mark="+"
                        token_form_base="POS_BW"
                        transliteration="UTF8">
                <normalization type="DIAC"/>
            </token_form_spec>
        </scheme_override>
    </scheme>
    <!-- Same as D34MT -->
    <scheme alias="MyD34MT">
        <scheme_override alias="MyD34MT"
                        form_delimiter="Âů"
                        include_non_arabic="true"
                        mark_no_analysis="false"
                        proclitic_delimiter=" "
                        stem_delimiter=" "
                        enclitic_delimiter=" "
                        tokenize_from_BW="false">
            <split_term_spec term="PRC3"/>
            <split_term_spec term="PRC2"/>
            <split_term_spec term="FUT"/>
            <split_term_spec term="PART"/>
            <split_term_spec term="S+"/>
            <split_term_spec term="PRC0"/>
            <split_term_spec term="REST"/>
            <split_term_spec term="ENC0"/>
            <!-- form0 -->
            <token_form_spec enclitic_mark="+"
                        proclitic_mark="+"
                        token_form_base="WORD"
                        transliteration="UTF8">
                <normalization type="PAREN"/>
                <normalization type="DIAC"/>
            </token_form_spec>
            <!-- form1 -->
            <token_form_spec enclitic_mark="+"
                        proclitic_mark="+"
                        token_form_base="WORD"
                        transliteration="UTF8">
                <normalization type="ALEF"/>
                <normalization type="YAA"/>
                <normalization type="PAREN"/>
                <normalization type="DIAC"/>
            </token_form_spec>
            <!-- form2 -->
            <token_form_spec enclitic_mark="+"
                        proclitic_mark="+"
                        token_form_base="LEXEME"
                        transliteration="UTF8">
                <normalization type="PAREN"/>
                <normalization type="LEX"/>
            </token_form_spec>
            <!-- form3 -->
            <token_form_spec enclitic_mark="+"
                        proclitic_mark="+"
                        token_form_base="POS_CATIB"
```

```
                        transliteration="UTF8">
                <normalization type="DIAC"/>
            </token_form_spec>
            <!-- form4 -->
            <token_form_spec enclitic_mark="+"
                        proclitic_mark="+"
                        token_form_base="POS_PENN"
                        transliteration="UTF8">
                <normalization type="DIAC"/>
            </token_form_spec>
            <!-- form5 -->
            <token_form_spec enclitic_mark="+"
                        proclitic_mark="+"
                        token_form_base="POS_BW"
                        transliteration="UTF8">
                <normalization type="DIAC"/>
            </token_form_spec>
        </scheme_override>
    </scheme>
</tokenization>
```

7.2 ALMOR Database Configuration

ALMOR may be configured to load custom (or user provided) databases instead of the databases that MADAMIRA is shipped with. To switch the databases, edit the key-value pairs in config/almor.properties and place the custom databases in the resources directory. This of course is possible only if the custom databases are in the format expected by ALMOR.

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 8 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 8 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, 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).

Note that running the same test data for MSA through MADAMIRA with BPC and NER turned on resulted in a speed of 602.2 words/sec, which is still substantially faster than the older systems.

	MSA		EGY	
Evaluation Metric	**MADA v3.2**	**MADAMIRA v1.0**	**MADA-ARZ v0.4**	**MADAMIRA v1.0**
EVALDIAC	**86.4**	86.3	83.8	**83.8**
EVALLEX	**96.1**	96.0	86.4	**87.9**
EVALPOS	**96.1**	95.9	91.8	**92.3**
EVALFULL	**84.3**	84.1	76.8	**77.8**
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	399.9	44.9	344.1
Server-client mode	N/A	**1512.1**	N/A	**1062.4**

Table 8: MADAMIRA 2.1 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**.

	Tokens	Tokens_norm	Tags	Tags_norm
Test-NewsWire	81.57	95.12	58.62	68.75
TestBroadcastNews	79.06	93.90	58.33	69.12
TestWebLogs	79.89	93.78	60.64	71.14

Table 9: MADAMIRA 2.1 Evaluation of BPC component - F-scores for Tokens, Tokens normalized and tags.

8.1 BPC

The evaluation results for the BPC component are documented in Table 9 and Table 10. The tables contain F-scores obtained from comparing the output of the BPC component to the ATB gold data. Since there are many tokenization mismatches between the BPC output and the ATB gold data, the evaluation only takes into consideration the counts for exact tokens. For example, the Chunks result is obtained from counting chunks that have exact tokens and exact chunk label matches. If any token contains a mismatch or a different label, the chunk is not counted as a correct match. While comparing the BPC output with the ATB gold, we found that there are some cases where the clitic marker "+" did not match while the respective tokenizations matched. We also found some mismatches due to different forms of Alef or Yaa only; hence the reason for

	Chunks	Chunks_boundary	Chunks_norm	Chunks_boundary_norm
Test-NewsWire	43.25	46.37	55.93	59.66
TestBroadcastNews	43.82	45.60	56.10	59.55
TestWebLogs	44.56	46.81	58.01	61.04

Table 10: MADAMIRA 2.1 Evaluation of BPC component - F-scores for Chunks, Chunks normalized and Chunks boundary normalized.

	Precision	Recall	F1
LOC	83.45	87.78	85.56
ORG	57.80	72.41	64.29
PER	64.89	77.98	70.83
OVERALL	73.47	82.62	77.78

Table 11: MADAMIRA 2.1 Evaluation of NER component using conllEval script.

	Precision	Recall	F1
LOC	78.51	86.06	82.11
ORG	83.38	92.26	87.60
PER	78.03	78.49	78.26
OVERALL	80.65	86.96	83.68

Table 12: MADAMIRA 2.1 Evaluation of NER component after ignoring order of entity types.

also reporting the normalized results which are relaxed comparisons that neglect these mismatches while comparing the tokens.

The description for each of the column headers in Table 9 and Table 10 are listed below.

- Tokens: Tokenization matching score.
- BPCTags: BPC BIO Tags matching score. This compares BIO tags (e.g. B_NP, I_VP, O etc) per individual token. (both the token and the BIO tag should match, O tag is also counted)
- BPCChunks: This compares a complete chunk (i.e. a group of tokens) for both tokens and the overall chunk tag matching (like NP, VP, PP ...etc. Note: O tags are not counted as they are out_of_chunk tags).
- BPCChunks_boundary: This compares the chunk boundaries i.e. it matches the grouping of tokens without paying attention to the group/chunk label.

  All "*_norm" scores are calculated the same way but after removing clitics "+" markers and normalizing Alf and Yaa forms (for both MADAMIRA output and Golden ATB).

8.2 NER

MADAMIRA NER component yields accuracy of 96.2%. Table 12 depicts MADAMIRA evaluation results using Precision, Recall, and harmonic F-score (F1) metrics on PERSON (PER), LOCATION (LOC), and ORGANIZATION (ORG) named entity types, in addition to the overall macro-average. We used conllEval script to measure MADAMIRA prediction performance. For more details on IOB tagging and conllEval, please visit http://www.clips.uantwerpen.be/conll2003/ner/

A separate evaluation was performed in which the system is rewarded for recognizing the word entity type regardless of its order. Results for this evaluation are shown in Table **??**. This form of evaluation can be further explained with the following example.

Word	Gold Tag	MADAMIRA Predicted Tag
ErDt	O	O
qnAp	ORG	O
Aljzyrp	ORG	ORG
xTAb	O	O
>wbAmA	PER	PER
En	O	O
qAnwn	O	O
Alhjrp	O	O

Table 13: MADAMIRA 2.1 Example of evaluation of NER component after ignoring order of entity types. O is a boundary tag that shows words that are not entity words.

Sentence: ErDt qnAp Aljzyrp xTAb >wbAmA En qAnwn Alhjrp

Gold Annotation: ErDt [ORG qnAp Aljzyrp] xTAb [PER >wbAmA] En qAnwn Alhjrp

MADAMIRA prediction: ErDt qnAp [ORG Aljzyrp] xTAb [PER >wbAmA] En qAnwn Alhjrp

The system will be penalized for not recognizing qnAp as ORG, but fully rewarded for recognizing Aljzyrp as ORG. Table 13 shows an example of the tags used during evaluation.

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 = 8223;

    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(HttpHostConnectException 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

B.1 Named Entity Recognition (NER)

The NER component currently supports MSA only. The component uses data taken from ACE 2004 and 2005 (LDC2005T09 and LDC2005E18). The data is split into train, tune, dev and test sets in a manner similar to the splits described in Appendix B for MADAMIRA MSA.

B.2 Base Phrase Chunking (BPC)

The BPC component currently supports MSA only. The component uses data taken from the following list of corpora. The data is split into train, tune, dev and test sets in a manner similar to the splits described in Appendix B for MADAMIRA MSA.

- **MSA-NewsWire Treebanks**: ATB1 4.1 LDC2010T13, ATB2 3.1 LDC2011T09, ATB3 3.2 LDC2010T08
- **MSA-BroadcastNews Treebanks**: ATB5 1.0 LDC2009E72, ATB7 1.0 LDC2009E114, ATB10 1.0 LDC2010E22, ATB12 1.0 LDC2011E17
- **MSA-Weblogs Treebanks**: ATB6 1.0 LDC2009E108, ATB11 1.0 LDC2011E16
- **Egyptian Treebanks**: ARZ1 2.0 LDC2012E93, ARZ2 2.0 LDC2012E98, ARZ3 2.0 LDC2012E89, ARZ4 2.0 LDC2012E99, ARZ5 2.0 LDC2012E107, ARZ6 2.0 LDC2012E125, ARZ7 2.0 LDC2013E12, ARZ8 2.0 LDC2013E21

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 14-18. Table 14 shows the basic morphology features; these are identical for both MSA and EGY.

Table 15 describes the part-of-speach (**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 16, 17 and 18 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 2.1
Aspect	LABEL	**asp**
	Command	c
	Imperfective	i
	Perfective	p
	Not applicable	na
Case	LABEL	**cas**
	Nominative	n
	Accusative	a
	Genitive	g
	Not applicable	na
	Undefined	u
Gender	LABEL	**gen**
	Feminine	f
	Masculine	m
	Not applicable	na
Mood	LABEL	**mod**
	Indicative	i
	Jussive	j
	Subjunctive	s
	Not applicable	na
	Undefined	u
Number	LABEL	**num**
	Singular	s
	Plural	p
	Dual	d
	Not applicable	na
	Undefined	u
Person	LABEL	**per**
	1st	1
	2nd	2
	3rd	3
	Not applicable	na
State	LABEL	**stt**
	Indefinite	i
	Definitie	d
	Construct/Poss/Idafa	c
	Not applicable	na
	Undefined	u
Voice	LABEL	**vox**
	Active	a
	Passive	p
	Not applicable	na
	Undefined	u

Table 14: MADAMIRA feature and value definitions, with the labels used to represent them under MADAMIRA 2.1. "LABEL" indicates the identifying tag used for that feature in MADAMIRA output files. These values are used in both MSA and the EGY dialect.

	POS Definition	MADAMIRA 2.1	Bies Equivalent	CATiB Equivalent	Buckwalter Stem Tag Equivalent
Part-of-speech	LABEL	**pos**	—	—	—
	Nouns	noun	NN ǀ NNS	NOM	FOREIGN, FUNC_WORD, NOUN
	Number Words	noun_num	NN ǀ NNS	NOM	NOUN_NUM
		noun_quant	NN ǀ NNS	NOM	NOUN_QUANT
	Proper Nouns	noun_prop	NNP ǀ NNPS	PROP	FOREIGN, FUNC_WORD, NOUN_PROP
	Adjectives	adj	JJ	NOM	ADJ
		adj_comp	JJ	NOM	ADJ_COMP
		adj_num	JJ	NOM	ADJ_NUM
	Adverbs	adv	RB	NOM	ADV
		adv_interrog	RP	NOM	INTERROG_ADV
		adv_rel	WP	NOM	REL_ADV
	Pronouns	pron	PRP	NOM	PRON_[1P ǀ 1S ǀ 2D ǀ 2FP ǀ 2FS ǀ 2MP ǀ 2MS ǀ 3D ǀ 3FP ǀ 3FS ǀ 3MP ǀ 3MS]
		pron_dem	DT	NOM	DEM_PRON_[D ǀ F ǀ FD ǀ FP ǀ FS ǀ MD ǀ MP ǀ MS ǀ P]
		pron_exclam	PRP	NOM	EXCLAM_PRON
		pron_interrog	RP	NOM	INTERROG_PRON
		pron_rel	WP	NOM	REL_PRON
	Verbs	verb	VBN ǀ VBP ǀ VBD	VRB ǀ VRB-PASS	VERB ǀ CV ǀ IV ǀ IV_PASS ǀ PV ǀ PV_PASS
		verb_pseudo	VBN ǀ VBP ǀ VBD	PRT	PSEUDO_VERB
	Particles	part	IN	PRT	PART
		part_dem	DT	PRT	—
		part_det	DT	PRT	DET
		part_focus	IN	PRT	FOCUS_PART
		part_fut	IN	PRT	FUT_PART
		part_interrog	IN	PRT	INTERROG_PART
		part_neg	RP	PRT	NEG_PART
		part_restrict	IN	PRT	RESTRIC_PART
		part_verb	IN	PRT	VERB_PART
		part_voc	IN	PRT	VOC_PART
	Prepositions	prep	IN	PRT	PREP
	Abbreviations	abbrev	NN	PROP	ABBREV
	Punctuation	punc	PUNC	PNX	NUMERIC_COMMA
	Conjunctions	conj	CC	PRT	CONJ
		conj_sub	CC	PRT	SUB_CONJ
	Interjections	interj	UH	PRT	INTERJ
	Digital Numbers	digit	CD	NOM	—
	Foreign/Latin	latin	IN	NOM	—

Table 15: MADAMIRA part-of-speech definitions and the labels used to represent them under MADAMIRA 2.1, 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 2.1
Proclitic 3 (AKA question proclitic or QUES)	LABEL	**prc3**
	No proclitic	0
	Not applicable	na
	Interrogative Particle *>a*	>a_ques
Proclitic 2 (AKA conjunction proclitic or CONJ)	LABEL	**prc2**
	No proclitic	0
	Not applicable	na
	Conjunction *fa*	fa_conj
	Connective particle *fa*	fa_conn
	Response conditional *fa*	fa_rc
	Subordinating conjunction *fa*	fa_sub
	Conjunction *wa*	wa_conj
	Particle *wa*	wa_part
	Subordinating conjunction *wa*	wa_sub

Table 16: MADAMIRA proclitic definitions for **prc3** and **prc2** (for MSA and EGY) and the labels used to represent them under MADAMIRA 2.1. 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 2.1
Proclitic 1 (AKA preposition proclitic or PART)	LABEL	**prc1**
	No proclitic	0
	Not applicable	na
	Interrogative *ish*	<i$_interrog
	Particle *bi*	bi_part
	Preposition *bi*	bi_prep
	Progressive verb particle *bi*	bi_prog
	Preposition *Ea*	Ea_prep
	Preposition *EalaY*	EalaY_prep
	Preposition *fy*	fiy_prep
	Demonstrative *hA*	hA_dem
	Future marker *Ha*	Ha_fut
	Preposition *ka*	ka_prep
	Emphatic Particle *la*	la_emph
	Preposition *la*	la_prep
	Response conditional *la*	la_rc
	Preposition *li* + preposition *bi*	libi_prep
	Empathic *la* + future marker *Ha*	laHa_emphfut
	Response conditional *la* + future marker *Ha*	laHa_rcfut
	Jussive *li*	li_jus
	Preposition *li*	li_prep
	Preposition *min*	min_prep
	Future marker *sa*	sa_fut
	Preposition *ta*	ta_prep
	Particle *wa*	wa_part
	Preposition *wa*	wa_prep
	Vocative *wA*	wA_voc
	Vocative *yA*	yA_voc
Proclitic 0 (AKA article proclitic or ART)	LABEL	**prc0**
	No proclitic	0
	Not applicable	na
	Demonstrative particle *Aa*	Aa_prondem
	Determiner	Al_det
	Determiner *Al* + negative particle *mA*	AlmA_detneg
	Negative particle *lA*	lA_neg
	Negative particle *mA*	mA_neg
	Negative particle *ma*	ma_neg
	Particle *mA*	mA_part
	Relative pronoun *mA*	mA_rel

Table 17: MADAMIRA proclitic definitions for **prc1** and **prc0** (for MSA and EGY) and the labels used to represent them under MADAMIRA 2.1. 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 2.1
Enclitics 2 (EGY only)	LABEL	**enc0**
	No enclitic	0
	Not applicable	na
	Negative particle suffix	part_neg
Enclitics 1 (EGY only)	LABEL	**enc0**
	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	**enc0**
	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 *lA*	lA_neg
	Interrogative pronoun *ma*	ma_interrog
	Interrogative pronoun *mA*	mA_interrog
	Interrogative pronoun *man*	man_interrog
	Relative pronoun *man*	man_rel
	Relative pronoun *ma*	ma_rel
	Relative pronoun *mA*	mA_rel
	Subordinating conjunction *ma*	ma_sub
	Subordinating conjunction *mA*	mA_sub

Table 18: MADAMIRA enclitic definitions and the labels used to represent them under MADAMIRA 2.1. 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** in 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.

D Appendix: Aramorph vs SAMA/CALIMA

Figure 1 below shows a comparison of prediction accuracy of MADAMIRA when used with SAMA (almor-msa-s31) and Aramorph (almor-msa-r13) morphological databases on various evaluation metrics for MSA. It is found that SAMA outperforms Aramorph on all metrics by a small margin.

Figure **??** below shows a comparison of prediction accuracy of MADAMIRA when used with CALIMA (almor-egy-c044) and Aramorph (almor-cra01) morphological databases on various evaluation metrics for MSA and EGY. It is found that CALIMA outperforms Aramorph on all metrics by a small margin.

Brief descriptions of each of the evaluation metrics follows:

- **EVALBW** – Percentage of words where the analysis chosen by MADAMIRA has the correct Buckwalter tag.
- **EVALBWPENN** – Percentage of words where the analysis chosen by MADAMIRA has the correct BWPenn tag.
- **EVALDIAC** – See Section 8: Evaluating MADAMIRA.
- **EVALLEX** – See Section 8: Evaluating MADAMIRA.
- **EVALPARTDIAC** – As EVALDIAC, but the diacritcs in the tail of the word (which carry information like case and which are relatively harder to predict correctly) are first discarded.
- **EVALPENNSTEM** – Percentage of words where the analysis chosen by MADAMIRA has the correct Penn part-of-speech tag.
- **EVALPOS** – See Section 8: Evaluating MADAMIRA.
- **EVALSTAR** – Percentage of words where the chosen analysis has is perfectly correct (that is, all the features match the gold values). This is the strictest metric used.
- **EVALSTD** – Percentage of words where the chosen analysis has the correct pos, num, gen, asp, vox, per, enc0, prc0, prc1, and prc2. This feature set is roughly equivalent to the evaluation metric that was used when MADA was first developed.
- **EVALSTRICT** – As EVALSTD, except that the chosen analysis must have the correct cas, mod and stt as well, and thus is a stricter metric.

D APPENDIX: ARAMORPH VS SAMA/CALIMA

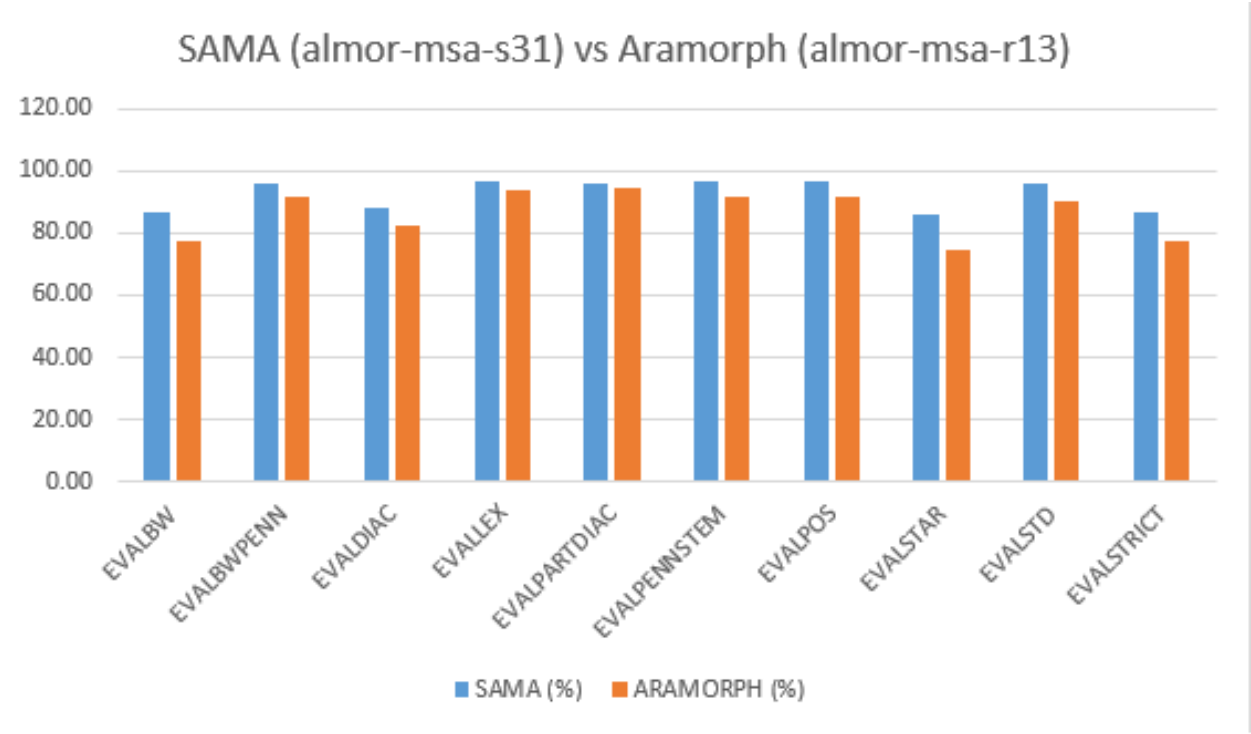

Figure 1: Comparison of MADAMIRA's prediction accuracy on various metrics when used with SAMA and Aramorph.

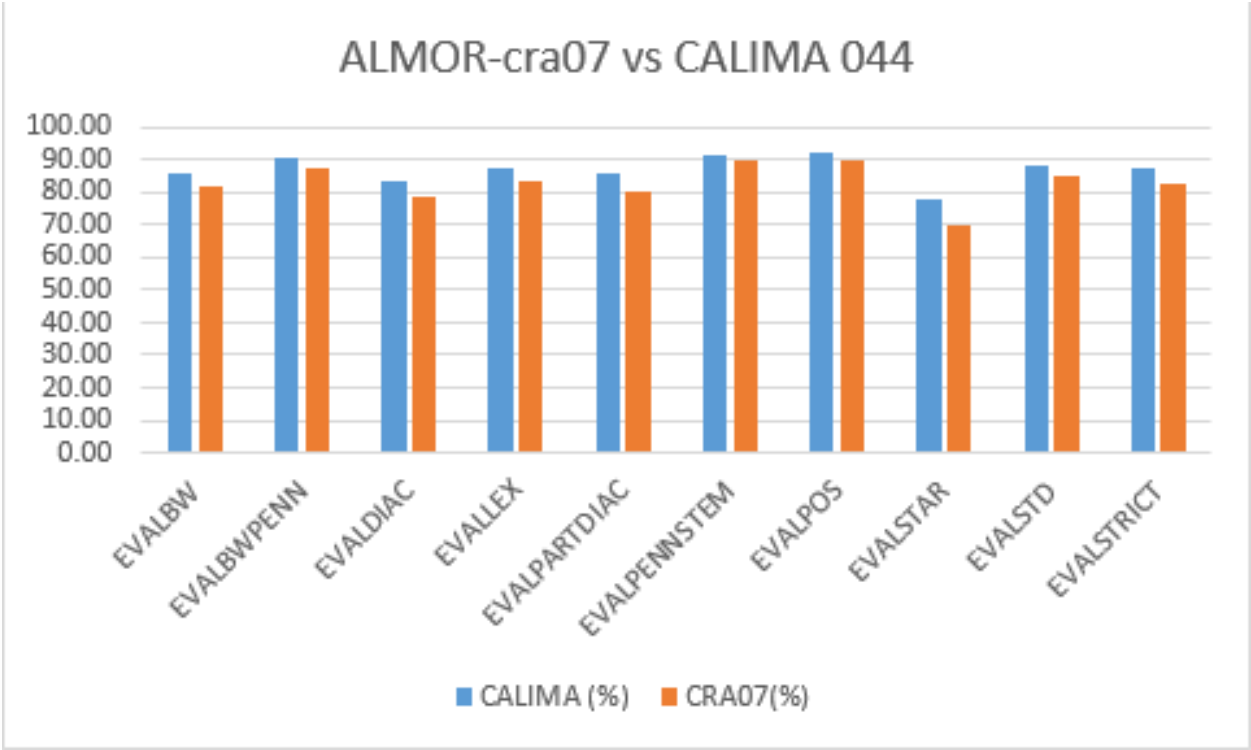

Figure 2: Comparison of MADAMIRA's prediction accuracy on various metrics when used with CALIMA and Aramorph.

D APPENDIX: ARAMORPH VS SAMA/CALIMA

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.

E Appendix: MADAMIRA Recommended Readings

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

2014

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

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 Soudi. 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.

F Appendix: MADAMIRA Change Log

MADAMIRA-release-01292014-1.0-beta-*
- Added a QuickStart README.txt file and sample input/output files.

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).

MADAMIRA-release-02182014-1.0-beta-*
- Added word character offsets to the output of each word. See Section 7.
- Server is capable of handling concurrent client requests. See Section 6.2.

MADAMIRA-release-03142014-1.0-beta-*
- Added Aramorph (aramorph3.1c.db) database as a replacement for SAMA3.1 MSA database.
- Fixed a bug in preprocessor that was not separating latin characters from utf8 Arabic characters that are touching.
- Added word type to XML output. See Section 7.

MADAMIRA-release-20140505-1.0*
- Added Aramorph morphological analyser
- Added to the user manual a comparison of MADAMIRA's accuracy with Aramorph vs SAMA on various metrics
- Added svm predictions to MADAMIRA XML output
- Made MADAMIRA wrapper thread-safe
- Added @@LAT@@ to Buckwalter output in raw mode
- Fixed a bug in the preprocessor that was not allowing the preprocessor from being switched off via config setting

MADAMIRA-release-20140825-1.0*
- Replaced aramorph3.1c.db with almor-msa-a12.db for overall increased accuracy. almor-msa-a12.db is an improved and renamed version of Aramorph3.1c.db
- Added capability for tokenization scheme customization. See Section 7.1

MADAMIRA-release-20141021-1.0*
- Add an API and API documentation.

MADAMIRA-release-20150108-1.0*
- DEPRECATED token_delimiter and replaced it with three new delimiters - proclitic, stem and enclitic delimiters see Section 7.1

MADAMIRA-release-20150401-2.0*
- Added map from MADAMIRA v1.0 pos tags to Buckwalter Stem Tag equivalent in Table 15.
- Replaced Aramorph v1.2 (almor-msa-a12.db) with Aramorph v1.3 (almor-msa-r13.db)
- Added Named Entity Recognition (NER) and Base Phase Chunking (BPC) components

MADAMIRA-release-20150421-2.1*
- Added Aramorph morphological analyzer for EGY (almor-cra07.db)