Building an Arabic Morphological Analyzer as part of an Open Arabic NLP Platform

Lahsen Abouenour[1], Said El Hassani[2], Tawfiq Yazidy[2], Karim Bouzouba[1], Abdelfattah Hamdani[2]

Email: abouenour@yahoo.fr, s.elhassani@iera.ac.ma, elyazidy@iera.ac.ma, karim.bouzoubaa@emi.ac.ma, hamdani@iera.ac.ma

[1] Mohammadia School of Engineers, Mohamed Vth University-Agdal, Avenue Ibn Sina B.P. 765 Rabat Morocco

[2] Institute for Studies and Research on Arabization, B-P 6216, Rabat, Morocco

Abstract

For many reasons (growth of Arabic Internet, greater interest on Arabic Media, etc.), Arabic NLP is facing many challenges. To contribute in answering some of them, we present in this paper an integrated and open Arabic NLP platform. This platform is dedicated to the development of many kinds of Arabic NLP applications. In addition of its openness, this platform is aimed to respect criteria such as standardization, flexibility and reusability. As a first step for the development of the platform, we present also its morphological layer with a focus on Arabic nouns. This analyzer is mainly based on a new classification of the Arabic nouns and provides useful information for other layers (syntax and semantics). Experiments done on selected corpora are very encouraging and the sketched architecture leads to many other interesting future works.

1. Introduction

Nowadays, in the context of NLP in general and Arabic NLP in particular, the following issues can be mentioned as trends: (i) help community researchers to unify their previous and current efforts (in terms of process and data); (ii) encourage the development of open source programs; (iii) find standard protocols and information representation formalisms for a better sharing; (iv) provide conventional languages resources for benchmarking and evaluation; (v) allow the reuse of already programmed modules; (vi) guarantee the portability of programs; etc.

In Arabic NLP many programs (analysers, specific applications such as translators, QA systems, IR systems, etc.) and data (dictionaries, lexicon, and corpora) have been developed. However, most of the time the work is not done as part of an integrated context where openness, standardisation, flexibility, reusability, and so on criteria are respected. Hence, there is a need to contribute in the development of programs and platforms with respect to the current Arabic NLP challenges.

The aim of our group is to follow the trends by developing an integrated Arabic NLP platform with the aim to offer an efficient integrated framework where the above criteria are taken into account. In the context of the present article, we expose the architecture of the platform and detail its morphological module. Indeed, after sketching the architecture of our platform, the first step that has been followed so far was the development of the Arabic morphology layer since it is the basic layer for most of NLP modules.

The structure of the article is as follows. In the second section, we explain the platform architecture with its different layers and modules. Before detailing the morphological module in the fourth section in terms of data and process, we present a brief overview of the Arabic morphology related works in the third section. In section Five, we present our preliminary experiments, and we give interpretation and discussion of the obtained results. Finally, in the last section we draw some conclusions and the future works to be done.

2. The Arabic NLP platform

Our platform is a Java open source multi-layer platform and a modular integrated development environment, dedicated to the development of Arabic NLP applications. As illustrated in Figure 1, the architecture contains the three regular NLP layers (Morphology, Syntax, and Semantics).

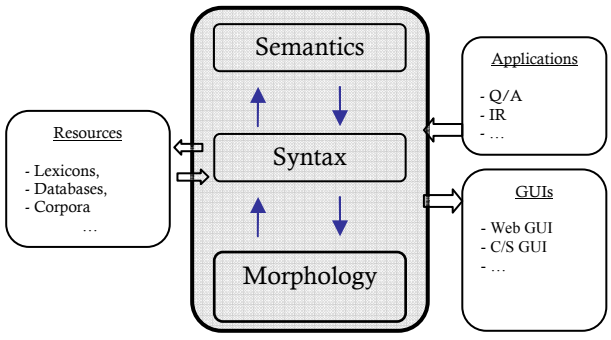

Figure 1: Architecture of the Arabic NLP Platform

Each layer is developed as a reusable Java API (library of classes) – Figure 2. The three layers make use of linguistic resources such as lexicons and corpora. In case the user needs directly to use any one of the layers, it is not necessary to call the libraries but the platform provides also appropriate graphical user interfaces (GUIs) for each layer. On another hand, it is also possible to develop NLP applications (Question/Answering systems, Information Retrieval systems, etc.) by calling one or more of the layers and linguistic resources. For example, it is known that to implement a Q/A system (Benajiba & Rosso, 2007), a light stemmer is enough. Thus, using our platform, the developer of such an application can call a subset of the functions from the morphology layer (`getNounPrefixes`, `getNounSuffixes`, `getNounRoot`, etc.). In addition, the solutions provided by each layer and the flow between them are in XML-like standardized format.

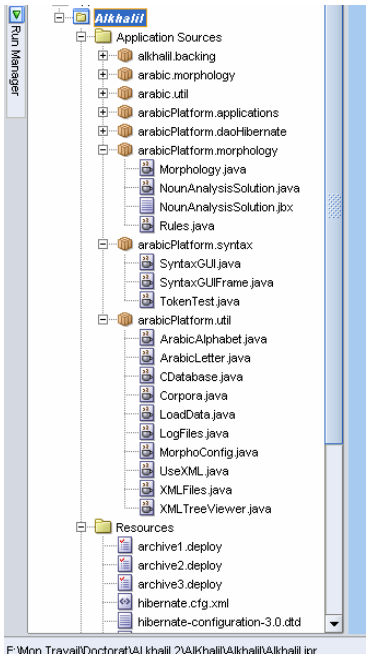

Figure 2: hierarchy of the Arabic NLP Platform APIs

The three layers form a hierarchy. Each layer is built on top of and uses the lower layers. However, a lower layer can be used by itself without the higher layers: the morphology layer (i.e. the associated APIs) can be used directly in any application without the other layers, syntactic layer (i.e. the associated APIs) can be used directly too, etc. Among the goals that have influenced the design of the platform was the goal to achieve a higher level of modularity and independence between its components.

The platform exhibits the following features:

- It guarantees the portability because it is developed with Java
- It could be used and evaluated by the community because it is open source
- It is flexible because the user can use any one of the GUIs or make call of one or more of the libraries
- Thanks to some mapping techniques and to the encapsulation of functions, resources can be downloaded in many proprietary formats (e.g. MySql, Oracle, SQLServer, Access, Ingress, etc.) or in XML standard format; This leads to the possibility of using already built resources and to a better sharing with the use of XML
- The reuse of components; i.e. some API functions can be reused in many contexts. For instance, the function that extracts the root from a word is used in the morphology GUI but could also be used in a Q/A system
- Appropriate documentation is provided to help developers efficiently use the API
- GUIs are provided as web services allowing the community to use the platform from Internet
- Ability to integrate other components

So far, we developed: (i) the structure of the whole architecture. This would allow programmers of the other layers to put their programs at the right place. (ii) the implementation of the morphological layer with its corresponding API and one specific GUI. As a first step we focused on Arabic nouns (verb analysis is to be considered next).

Let us mention also that the other layers are structured as Java interfaces (in term of software engineering), i.e. that a layer could be implemented as a plug in of an already existing tool. This could be the case for example of Buckwalter morphological analyser that can be integrated in our platform with the condition that it respects our API structure.

The analyzer is described in section 4. The next section gives a short presentation of some related works.

3. Related Works

Morphology is the most basic layer over which higher syntactic and semantic layers are built (Attia, 2000). It covers the study of the structure of words and is a term for that branch of linguistics concerned with the forms words take in their different uses and constructions (El-Sadany, 1989).

Morphological processing concerns two different tasks according to the operation type: in generation we produce correct forms using given morphemes, while in analysis we try to identify morphemes for a given word. The number of Arabic morphological analyzers and stemmers is increasing. In this part, we describe some of well known of them (a more detailed list of such analysers could be found in (Al-Sulaiti, 2004)):

- Buckwalter Arabic Morphological Analyzer (2004): Used by Linguistic Data Consortium (LDC) for POS tagging of Arabic texts, this analyzer contains

over 77800 stem entries which represent 45000 lexical items. The parser output uses a transliteration system. It has been criticised by (Attia, 2006) for excessive manual processing to state rules and because it analyses only words that appear in Arabic dictionaries.

- Beesley's Xerox AMA and generator (2001): presented in Java Applet, it's based on finite state techniques with two level morphological analysis: level for roots and patterns and an other for affixes, prefixes, enclitics and some forms such as prepositions, conjunctions.
- Sakhr's Arabic Morphological Analyzer[1]: the analyzer of Sakhr Company covers modern and classical Arabic and it identifies the base form by removing all the affixes. It gives also the morphological pattern. However, it suffers from some problems related to disambiguation approach and heterogeneity of processing (Attia, 2000).
- Khoja's stemmer (Khoja and Garside, 1999) attempts to find roots for Arabic words. His stemmer is based on four lists: list of prefixes, list of suffixes, list of roots and list of patterns. When it removes prefixes and suffixes, then it checks a list of patterns to determine whether the remainder could be a known root with a known pattern.
- Larkey's light stemmer (Larkey and Ballesteros, 2002) removes the most frequent suffixes and prefixes (light 10) from the Arabic surface word given. The aim of this light stemmer is not to produce the root but to find the stem.

All of these approaches and others fail to deal with some particular difficulties of Arabic morphology or don't answer the current Arabic NLP challenges:

- Some of them are not available for the community
- Approaches based on a list of patterns have to do with the fact that some Arabic patterns are assigned to different categories (N, V, A).
- The availability of many types of nominal categories and several rules/processes that are not relevant or adequate to the Arabic morphology.
- The absence of a reasonable solutions for the issue of the fact that some prefixes and suffixes are similar to the basic character (in the beginning and the end) of Arabic words, or the fact that Arabic surface forms are ambiguous unless it can have many vocalized forms.

4. The Arabic Nouns Morphological Analyzer

In this part we describe the Arabic Nouns Morphological Analyzer (فريق المعالجة الآلية للغة العربية , 2007). Its approach is a little bit similar to that of (Buckwalter, 2004) in that it uses a lexicon with explicit linguistic classes. However, our analyzer provides in addition a set of information that (Buckwalter, 2004) do not provide. Moreover, Buckwalter's analyzer is based on about 20 types and misses some prefixes and suffixes (e.g. أ). Our analyzer does not contain any compatibility constraints like buckwalter's, but it uses additional rules like (Habash and Rambow, 2006). So, a lexicon of Arabic nouns has been constructed according to a new categorization. In order to guarantee high performance for our analyzer, we have classified the Arabic nouns into three linguistic classes according to their ability or not to be attached with specific list of suffixes.

Class 1 prohibits the following suffixes: ين ،ون ،ة. It includes four types of nouns: common nouns, event nouns, proper names and broken plurals.

Class 2 prevents suffixes such as: ان، يْن، ون ين. It contains common nouns (especially kind nouns) and event nouns that can carry the classifier (ة). Note that these two types of nouns are not cited in the first class.

Class 3 forbids one suffix only: ي. It includes derived nouns: present participles, past participles and the exaggerations forms.

We manually developed three dictionaries: one for Arabic nouns, one for prefixes, and one for suffixes. These dictionaries are organized as follows:

- The fields of the nouns dictionary are: the root, the stem, (vocalized and non-vocalized), the category, the type (such as broken plural, proper names, event nouns, etc.), the number and gender. Table 1 is an extraction of this dictionary for the root (ك ت ب).
- For prefixes and prefixes, we have some important morpho-syntactic features for different Arabic morphemes such as number, gender and person in possessive pronouns, or case. We have also some information about its grammatical functions such as coordination, classifier, genitive preposition, a type of the noun, etc.

gender	*number*	*type*	*category*	*Stem vocalized*	*Stem non vocalized*	*root*
						ك ت ب
1	1	2	1	كَتْب	كتب	
2	2	4	1	كُتُب	كتب	
1	1	1	1	كِتَاب	كتاب	
1	2	4	1	كُتَاب	كتاب	
1	1	1	1	كُتَاب	كتاب	
2	1	2	1	كِتَابَة	كتابة	
1	1	1	1	كُتَيِّب	كتيب	
1	1	1	1	مَكْتَب	مكتب	

Table 1: Extraction from our lexicon of the entry كتب

[1] www.sakhr.com

The goal of our analyzer is not only to remove all prefixes and suffixes from the surface Arabic words to find roots or stems, but also to:

- introduce many special information about stems and its prefixes and suffixes
- find all possible results existing in the Arabic lexicon.

Finally, there are a set of processes (rules) that apply in particular cases. For example, we have to predict that the form of ء (hamza) can be ـئـ or ؤ, the omission of ة when we move from singular forms to plurals ones in words like (معلمة-معلمات), and so on.

Figure 3 below describes how the morphological analyzer processes.

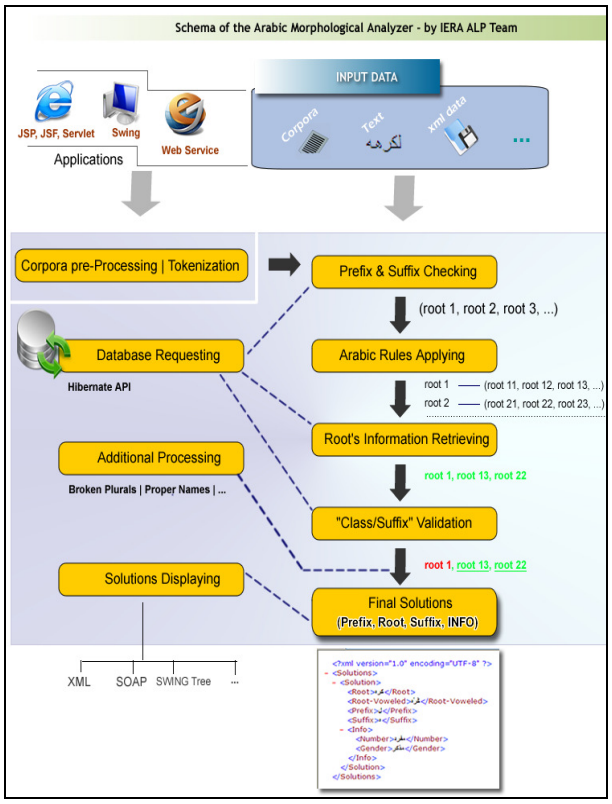

Figure 3: Schema of our Arabic Nouns Analyzer

Given an Arabic word, text or corpora, our analyzer begins by pre-processing it applying text segmentation. Next, for each word, it tries to look up a list of possible solutions. They are composed of a prefix, a root and a suffix. For each possible solution we apply a set of specific Arabic rules according to its elements (e.g. if prefix is ل then apply rule 1). After this step new possible solutions are generated. All those possible solutions are checked from the lexical database. After the category validation, the analyzer returns the final solutions under different formats (XML, GUI, etc.).

Additional processing can be done before displaying final solutions in order to resolve some particular problems like broken plurals and proper names, or to disambiguate with vocalization.
Figure 4 below shows a snapshot of one GUI using our analyzer API:

Figure 4 below show a snapshot of two GUI using our analyzer API:

Figure 4: GUI for analyzing a given Arabic text

In the first figure we introduce an Arabic text to be processed. The second figure shows an example of our analyzer processing report where indicated the results of each step (getting prefixes, getting suffixes, applying rules, etc.). The analysis of the word لكرمه reports two possible prefixes (ل, NULL) and two suffixes (ه,NULL). After applying linguistic rules to the possible solutions above, the system validate the only solution given as result (root : ك-ر-م stem : كرم prefix : ل suffix : ه).

As mentioned before, the solutions are also stored under XML format.

5. Results and Evaluation

The experiments were done with a 1.66 GHZ core 2 duo processor with 1 Go RAM and 1 Mo cache. Table 1 provides the detailed results of the three corpora used for the evaluation:

- Corpus 1 is a culture content extracted from a cultural web site[2].
- Corpus 2 is a one topic article extracted from el akhbar online newspaper[3].

[2] http://fatiha.sosblog.fr/Premier-blog-b1/Day7-b1-m20080207.htm
[3] http://www.elakhbar.org.eg/akhbarelyom/Issues/

- Corpus 3 is multi topics article extracted from the ecoworld magazine[4].

	Corpus 1	Corpus 2	Corpus 3
Average solutions per word	2	6	1
Response time (ms)	4300	11329	7857
Performance (%)	87,07	80,53	78,81

Table 1: Results of experiments

In the experiments done we process words (nouns and verbs) in a corpus, then, we calculate performance as: P=number of words with solutions/number of nouns. Note that we do not verify whether the solution is correct or not, and the number of nouns is calculated manually.

The analyzer reaches an average performance of 82.14 %. The remainder percentage is due to the lack in the dictionary of some proper names (persons, places, etc.) and to the lack of some specific rules that converts a singular noun to its plural (e.g in the lexicon we have سماء and in corpora we can find سماوات).

6. Conclusion and Future Work

We presented in this paper an open platform that answers some of the current and future Arabic NLP challenges such as openness, portability, reusability, and flexibility. After explaining the different components of the platform, we detailed its morphological layer as the module that has been developed so far, supported by some preliminary experiments.

In the next future, we plan to:

- Extend the dictionary with proper names and integrate new morphological rules to improve the performance of the analyzer;
- Confirm the results by further experiments on bigger and conventional corpora (e.g. LDC or Elda corpora);
- Develop the morphological analysis of the Arabic verbs;
- Develop the other modules and layers of the platform;
- Develop some specific applications (e.g. QA systems, IR systems) in the context of the platform.

7. References

Al-Sulaiti, L. (2004). Developing a corpus of Contemporary Arabic. Submitted in accordance with the requirements for the degree of Master of science, The University of Leeds.

Attia, M., (2000). A large-scale computational processor of the Arabic Morphology and applications. thesis submitted to the faculty of engineering, Cairo University.

Attia, M. (2006). An Ambiguity-Controlled Morphological Analyzer for Modern Standard Arabic Modelling Finite State Networks. The Challenge of Arabic for NLP/MT Conference, the British Computer Society, London.

Beesley, K., (2001). Finite-State Morphological Analysis and Generation of Arabic at Xerox Research: Status and Plans in 2001. Proceedings of the Arabic Language Processing: Status and Prospect, 39th Annual Meeting of the Association for Computational Linguistics. Toulouse, France.

Benajiba, Y., Rosso, P. (2007). Towards a measure for Arabic corpora quality. In Proceedings of the seventh International Conference CITALA 07. Rabat, Morocco. pp. 213--220.

Buckwalter, T. (2004). Issues in Arabic Orthography and Morphology Analysis. In Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages, COLING 2004, Geneva.

El-Sadany, T. A., Hashish, M. A. (1989). An Arabic Morphological System. IBM SYSTEM JOURNAL vol 28-no 4.

Habash, N., Rambow, O. (2006). Morphological Analysis for Arabic Dialects. In Proceedings of COLING-ACL, Sydney, Australia.

Khoja, S., Garside, R. 1999. Stemming Arabic text. Computing Department, Lancaster University, Lancaster.http://www.comp.lancs.ac.uk/computing/users/khoja/stemmer.ps.

Larkey, L. S., Ballesteros, L., 2002. Improving stemming for Arabic information retrieval: light stemming and co-occurrence analysis Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, Tampere, Finland.

فريق المعالجة الآلية للغة العربية، 2007 .التحليل الصرفي للأسماء العربية .ندوة دولية حول المعالجة الآلية للغة العربية، معهد الدراسات و الأبحاث للتعريب، الرباط، المغرب.

[4] http://www.ecoworldmag.com