International Journal of Computational Linguistics (IJCL)

ISSN : 2180-1266

Volume 1, Issue 2

Number of issues per year: 6

International Journal of Computational Linguistics (IJCL)

Volume 1, Issue 2, 2010

Edited By
Computer Science Journals
www.cscjournals.org

International Journal of Computational Linguistics (IJCL)

Book: 2010 Volume 1, Issue 2
Publishing Date: 30-10-2010
Proceedings
ISSN (Online): 2180-1266

IJCL Journal is a part of CSC Publishers
http://www.cscjournals.org

Typesetting: Camera-ready by author, data conversation by CSC Publishing Services – CSC Journals, Malaysia

CSC Publishers

Editorial Preface

The International Journal of Computational Linguistics (IJCL) is an effective medium for interchange of high quality theoretical and applied research in Computational Linguistics from theoretical research to application development. This is the second issue of volume first of IJCL. The Journal is published bi-monthly, with papers being peer reviewed to high international standards. International Journal of Computational Linguistics (IJCL) publish papers that describe state of the art techniques, scientific research studies and results in computational linguistics in general but on theoretical linguistics, psycholinguistics, natural language processing, grammatical inference, machine learning and cognitive science computational models of linguistic theorizing: standard and enriched context free models, principles and parameters models, optimality theory and researchers working within the minimalist program, and other approaches.

IJCL give an opportunity to scientists, researchers, and vendors from different disciplines of Artificial Intelligence to share the ideas, identify problems, investigate relevant issues, share common interests, explore new approaches, and initiate possible collaborative research and system development. This journal is helpful for the researchers and R&D engineers, scientists all those persons who are involve in Computational Linguistics.

Highly professional scholars give their efforts, valuable time, expertise and motivation to IJCL as Editorial board members. All submissions are evaluated by the International Editorial Board. The International Editorial Board ensures that significant developments in image processing from around the world are reflected in the IJCL publications.

IJCL editors understand that how much it is important for authors and researchers to have their work published with a minimum delay after submission of their papers. They also strongly believe that the direct communication between the editors and authors are important for the welfare, quality and wellbeing of the Journal and its readers. Therefore, all activities from paper submission to paper publication are controlled through electronic systems that include electronic submission, editorial panel and review system that ensures rapid decision with least delays in the publication processes.

To build its international reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Scribd, CiteSeerX Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJCL. We would like to remind you that the success of our journal depends directly on the number of quality articles submitted for review. Accordingly, we would like to request your participation by submitting quality manuscripts for review and encouraging your colleagues to submit quality manuscripts for review. One of the great benefits we can provide to our prospective authors is the mentoring nature of our review process. IJCL provides authors with high quality, helpful reviews that are shaped to assist authors in improving their manuscripts.

Editorial Board Members
International Journal of Computational Linguistics (IJCL)

Table of Content

Volume 1, Issue 2, October 2010

Pages

Designing a Rule Based Stemmer for Afaan Oromo Text

Debela Tesfaye dabookoo@yahoo.com
Faculty of Informatics/Department of Information Science Addis Ababa University Jimma, 378, Ethiopia

Ermias Abebe ermiasabe@gmail.com
Faculty of Informatics/Department of Information Science Addis Ababa University Addis Abeba, Ethiopia

Abstract

Most natural language processing systems use stemmer as a separate module in their architecture. Specially, it is very significant for developing, machine translator, speech recognizer and search engines. In this work, a stemming system for Afan Oromo is presented. This system takes as input a word and removes its affixes according to a rule based algorithm. The result of the study is a prototype context sensitive iterative stemmer. Error counting technique was employed to evaluate the performance of this stemmer. The errors were analyzed and classified into two different categories: under stemming and over stemming errors. For testing purpose corpus which is collected from different public Afaan Oromo newspapers and bulletins is used. Newspapers, bulletins and public magazines are considered as consisting different issues of the community: social, economical, technological and political issues. This will reduce the probability of making the corpus biased toward some specific words that do not appear in everyday life. According to the evaluation of the experiments, it can be concluded that an overall accuracy of the stemmer is encouraging which shows stemming can be performed with low error rates in high inflected languages such as Afan Oromo.

Keywords: Afan Oromo stemmer, Rule based Stemmer, Context sensitive stemmer

1. INTRODUCTION

We can find a variety of internet search engines with advanced search parameters for retrieving documents in a document collection. During the development of search engines we can notice an ongoing specialization on the searching features. These engines are becoming more and more sophisticated trying to cover user’s demands to access specific information.

One of the attempts to make the search engines more effective in information retrieval was the usage of word stemming. For grammatical reasons, documents are going to use different forms of a word, such as organize, organizes, and organizing. Additionally, there are families of derivationally related words with similar meanings, such as democracy, democratic, and democratization. In many situations, it seems as if it would be useful for a search for one of these words to return documents that contain another word in the set. Stemming makes families of derivationally related words with similar meanings represented using single term. A stemming algorithm is a procedure that reduces all words with the same stem to a common form by stripping of its derivational and inflectional suffixes [1]. Using Stemming, many contemporary search engines associate words with prefixes and suffixes to their word stem, to make the search broader in the meaning that it can ensure that the greatest number of relevant matches is included in search results.

Afaan Oromo, one of the major languages that is widely spoken and used in Ethiopia, is morphologically very productive; derivation, reduplication and compounding are also common [2]. Obviously, these extensive inflectional and derivational features of the language are presenting various challenges for text processing and information retrieval tasks in Afaan Oromo.

This paper describes the development and evaluation of a context sensitive rule based stemmer for Afan Oromo. Most of the concepts are adopted from stemming algorithm developed by Porter [3]. We have chosen to modify the stemming algorithm developed by Porter [3] because it is well known and is frequently used in experimental IR systems.

2. STEMMING ALGORITHMS

A basic characteristic of stemming algorithm is whether it is context-free or context sensitive, which refers to any attribute of the remaining stem.

2.1 Context-free

In context-free algorithm, no restriction is placed on the removal of a suffix and thus any ending, which matches, is accepted for stripping.

2.1. Context sensitive

In context sensitive algorithms, however, various restrictions are placed on the usage of the suffix. Therefore, such kind of algorithm requires the construction of suffix dictionary and the formation of a set of rules defining the morphological context of the suffixes. The dictionary gives the exact suffix form, while the rules define conditions to be tested in order to apply the rules.

One of the widely used context sensitive rule based stemmer is that of Porter [3]. The Porter stemmer has five steps and applying rules within each step. Within each step, if a suffix rule matched to a word, then the conditions attached to that rule are tested on what would be the resulting stem, if that suffix was removed, in the way defined by the rule. For example such a condition may be, the number of vowel characters, which are followed be a consonant character in the stem (Measure), must be greater than one for the rule to be applied [3]. Within each phase there are various conventions to select rules, such as selecting the rule from each rule group that applies to the longest suffix.

In context sensitive stemmers, If the set of rules defining the correct morphological context for the suffix is satisfied, it is replaced by another string, either the null string (if the suffix is to be removed) or specified replacement string (for example, to create nominal forms to adjectival forms). Both dictionary and rules require careful analysis of vocabulary and language behavior, and are thus time-consuming to create. However, generally such techniques are rewarded by high accuracy and speed, and simplicity in implementation [4].

3. OVER VIEW OF AFAN OROMO

Afaan Oromo is one of the major African languages that is widely spoken and used in most parts of Ethiopia and some parts of other neighbor countries like Kenya and Somalia. Currently, it is an official language of Oromia state (which is the largest Regional State among the current Federal States in Ethiopia). It is used by Oromo people, who are the largest ethnic group in Ethiopia, which amounts to 34.5% of the total population [5]. With regard to the writing system, Qubee (a Latin-based alphabet) has been adopted and become the official script of Afaan Oromo since 1991.

Like a number of other African and Ethiopian languages, Afaan Oromo has a very rich morphology (Oromoo, 1995). It has the basic features of agglutinative languages where all bound forms (morphemes) are affixes. In agglutinative languages like Afaan Oromo most of the grammatical information is conveyed through affixes (prefixes, infixes and suffixes) attached to the roots or stems. Both Afaan Oromo nouns and adjectives are highly inflected for number and gender. In contrast to the English plural marker s (-es), there are more than 12 major and very common plural markers in Afaan Oromo nouns (example: -oota, -ooli, -wwan, -lee, -an, -een, -oo, etc.) [2]. Afaan Oromo verbs are also highly inflected for gender, person, number and tenses. Moreover, possessions, cases and article markers are often indicated through affixes in Afaan Oromo. Since Afaan Oromo is morphologically very productive, derivations and word formations in the language involve a number of different linguistic features including affixation, reduplication and compounding [2].

In this research, the morphological analysis of the language is organized in to 6 categories. The categories are: nouns, pronouns and determinants, case and relational concepts, functional words, verb and adverbs. Almost all Oromo nouns in a given text have person, number, gender and possession markers which are concatenated and affixed to a stem or singular noun form. Like wise, determinants have number, gender, adjectives, and quantifier markers similar to Afan Oromo nouns. Afaan Oromo verbs are also highly inflected for gender, person, number, tenses, voice and transitivity. Furthermore, prepositions, postpositions and article markers are often indicated through affixes in Afaan Oromo.

4. RELATED WORKS

Very limited works have been done in the past in the areas of stemming in relation to Afaan Oromo. One of the stemmer is developed by Kekeba, Varma and Pingali[7] and it used to develop an Oromo-English CLIR system that enable user to access and retrieve online information sources that are available in English by using Afan Oromo queries.

The stemmer used a rule based suffix-stripping algorithms focusing on very common inflectional suffixes of Oromo language. This light stemmer is designed to automatically remove frequent inflectional suffixes attached to headwords (base-word forms) of Afaan Oromo. Some of the common suffixes that have been considered in their light stemmer include gender (masculine, feminine), number (singular or plural), case (nominative, dative), possession morphemes and other related morphological features in Afaan Oromo.

This kind of stemming techniques can have several shortcomings when applied to heavily inflection language such as Afaan Oromo. These shortcomings can include:

- Improper removal of some affixes (part of a word might appear to be a prefix or suffix). Their stemmer simply strips of any end of a word that matches one of the affixes in a list without any condition to be tested. Afaan Oromoo suffixes are not quite different from non suffix endings. Suffixes like -aa as in the case of maqaa,mucaa;-ee,-lee as in the case of killee, eelee, itilee, mee, kee, ree; -an,-n in the case of aannan, ilkaan, Afaan, shan, kan; -tuu, -tu as in the case of utuu, hatuu, nyaatu, baatu, kaatu are part of a root word as indicated in the above words that should not be conflated. There fore most word endings can be part of a word and suffix as well. So simple removal of endings can create invalid stems. But the above stemmer conflates the words since the endings of the words matches one of the suffixes.
- The stemmer didn`t considered words formed by duplication of some characters at all. But Afaan Oromo is rich in this kind of word formation. Most of the adjectives form the plural by reduplication

of the first syllable. For example words like, jajjabaa (stron-plural), gaggabaabaa (short-plural) are formed from -jabaa and -gabaabaa by duplicating the first syllabus, respectively.

Another stemmer is developed by Wakshum[10] which use suffix table in combination with rules that strips off suffix from a given word by looking up the longest match suffix in the suffix list[10]. List of suffixes are compiled automatically by counting the most frequent endings. Other linguistically valid suffixes are also included manually. The stemmer fined the longest suffixes that matchs the end of a given word and remove. The problems he faced are similar with that of Kekeba, Varma and Pingali[7]. Some of these are: irregular formation of variants from root word, the challenge to increase the number and complexity of the rules and words formed by duplication of some characters. The lists of the suffixes are not linguistically valid. Out of 342 suffixes compiled by the stemmer only 70 are linguistically valid. This is because the lists of suffixes are compiled statistically and requires no analysis of the language. The suffixes are compiled by counting and sorting the most frequent endings. One great problem occurred with this kind of compilation of suffixes is that during conflation frequently occurring endings which are part of root word is considered as suffixes and removed.

Another problem is that, the compilation of stop words is also done statistically and frequently occurring content bearing words are also included. For example,barannoo,barattoo,barnoota are varieties of the root barat(to learn) ,duree(rich),fayyadam(to use),dhiyeess(to approach),barsiisu(to teach), barsiisa (teacher), agarsiis (to show) and the like are include as stop word. And this indicates that frequently occurring content bearing words of Afan Oromo are not considered by the stemmer. More than 96 content bearing words that occur frequently are included as stop word.

This necessitates the need for the detailed knowledge of the language to use such frameworks to generate a stemmer that handles words formed by duplication of some characters and other under/over stemming problems.

5. AFAN OROMO STEMMER

5.1. Introduction

It is not possible to apply the stemming algorithms developed for English or other languages like porter's [3] to Afan Oromo due to differences in the patterns of word formations and differences in their morphologies. Some of the concepts from Porter stemmers are however adopted to develop a stemmer for Afan Oromo. Specifically, Concepts about measure, arranging the rules in clusters ,analyzing word formation based on the nature of their endings(for example words that attaches the suffixes *–de* must end with b/g/d in Afan Oromo) are taken from porter algorithm.

Afan Oromo stemmer is based on a series of steps that each removes a certain type of affix byway of substitution rules. These rules only apply when certain conditions hold, e.g. the resulting stem must have a certain minimal length. Most rules have a condition based on the so-called measure. The measure is the number of vowel-consonant sequences (where consecutive vowels or consonants are counted as one) which are present in the resulting stem. This condition must prevent that letters which look like a suffix but are just part of the stem will be removed. Other simple conditions on the stem are:

- Does the stem end with a vowel?
- Does the stem end with a consonant?
- Does the stem end with specific character?
- Does the 1st syllabus of the stem duplicated?

5.2. Extensions to Porter Stemmer`s Implementation

Most Afan Oromo words form repetition by duplicating some of the starting characters. Because this affix exhibits certain pattern that can be recognized, the algorithm has been extended to handle them. The original Porter stemmer [3] only treats suffixes.

5.3. Definitions of Afan Oromo Stemmer

Define Afan Oromo vowel as one of

a e i o u

Define Afan Oromo consonants as one of

b c d f g h j k l m n p

q r s t v w x y z `

Define a valid de, du, di, do, dan-ending as one of

b g d

R1 is the region after the first non-vowel following a vowel. If the word starts in vowel it is the region before the next consonant.

R2 is the region after the first non-vowel following a vowel in R1

C1 is the firs character of a word.

5.4. Compilation of Stop Word List

As can be seen from the table, the stop word list consists of prepositions, conjunctions, articles, and particles. The stop word list is collected and compiled based on information in A Grammatical sketch of Written Oromo [6] and *Caasluga Afaan Oromoo, Jildi I* [2]. The list of linguistically valid Afan Oromo prepositions, conjunctions, articles, particles are available on the above mentioned books.

Number	word	English
1	kana	This
2	sun	That
3	ani	Me?
4	inni	He
5	isaan	They
6	ise	She
7	akka	Like
8	Ana	Me
9	fi	and

TABLE 1: Afan Oromo stop word list examples

There are no any content-bearing words in the stop word list.

5.5. The Test Set

A corpus is a collection of texts or speech stored in an electronic machine-readable format [11]. Balanced corpus is needed to process natural language processing tasks like stemming. Balanced corpus is a corpus that represents the words that are used in a language. As indicated in [11] texts collected from a unique source, say from scientific magazines, will probably be biased toward some specific words that do not appear in everyday life. Such types of corpuses are not balanced corpus so that they are not appropriate for many natural languages processing in general and stemming in particular except for special purposes. However, developing a balanced corpus is one of the difficult tasks in NLP research because it requires collecting data from a wide range of sources: fiction, newspapers, technical, and popular literatures. As a result it requires much time and human effort.

For this particular study, corpus was collected from different popular Afaan Oromo newspapers (Bariisaa, Bakkalcha Oromiyaa and Oromiyaa) and bulletins (Qabee and Oromiyaa) to balance the corpus. Newspapers, bulletins and public magazines are considered as consisting different issues of the community: social, economical, technological and political issues. So that they are a potential source for collecting balanced corpus for natural language processing tasks.This corpus is used for evaluating the performance of the stemmer. The corpus consists of 159 sentences (the total of 1621 tokens).

5.6. Afan Oromo Stemmer Rule Clusters

Based on the information written on A Grammatical sketch of Written Oromo [6] and Caasluga Afaan Oromoo, Jildi I [2] six rule clusters were created for Afan Oromo stemmer.

The rule-clusters are defined by similarity of their pattern in word formation, the level at which the affixes occur in the word formation process (the most common order/sequence of Afaan Oromo suffixes (within a given word) is: <stem> <derivational suffixes> <inflectional suffixes> <attached suffixes> [7]) and the length of the affixes.

Thus, this stemmer removes (from the right end of a given word) first all the possible attached suffixes, then inflectional suffixes and finally derivational suffixes step by step. This is done to reduce computational time. Morphemes that are formed out of this sequence can also be removed though it takes additional computational time. Complex affixes are thus removed in consecutive steps. For example, *baratootarratti* (on the students) has four suffixes*: -itti, -rra, -oota* and *-at*. Therefore firs *-tti*, then *-rra*, then *-oota* and finally *-at* is removed to get the root *"bar-".*

In addition to the affix-rules, context sensitive conditions are also designed to cover some specific phenomena. Examples of these conditions are, *Endswith V/C*, i.e. when remaining stem ends in a vowel or consonant; *Ends with B/G/D*, i.e. when remaining stem ends in the characters *B* or *G* or *D*; *Endswith CC*, i.e. when remaining stem ends in double consonant.

The affix-rules have the following general form:

Affix $\longrightarrow$ ***substitution measure-condition <additional conditions>***

Where:

Affix is a valid Afan Oromo prefix or suffix

In Afan Oromo repetition (plural) is formed by duplicating the first syllabus and it is also considered as prefix.

Substitution is a string which is substituted with a given affix to produce valid stem.
Measure-condition is the number of vowel-consonant sequences (where consecutive vowels or consonants are counted as one) which are present in the resulting stem.
Additional conditions- additional conditions are also designed to cover some specific phenomena. Examples of these conditions are, *Endswith V/C,*

The sixth cluster contains derivational suffixes whose measure is greater or equal to 1 and ends with consonant.
e.g.

“eenya”				
“offaa”	$\longrightarrow$	0	measure > =1	EndsWith c
“annoo”,				
“ummaa”				

Explanation for the sixth rule cluster:
If a word ends with one of the suffixes: “eenya”, “offaa” , “annoo”, “ ummaa” and measure> =1 and the remaining stem ends with consonant, delete the suffix.
E.g. qabeenya(asset) ends with the suffix “eenya”,measure>=1 and the remaining stem *qab-* ends with the consonant *b.* Therefore it is correctly stemmed to *qab-*.

The fifth cluster contains suffixes that are removed if measure is greater or equal to 1 or substituted with the suffix -` if measure equal zero.
e.g.

“`aa”			
“’e”,			
“’u”	$\longrightarrow$	0	measure > =1
“’ee”,			
“suu”			
“sa”,			
“sse”			
“nya”	$\longrightarrow$	`	measure > =0

Explanation for the fifth rule cluster:
If a word ends with one of the suffixes: “`aa”, “’e”, “’u”, “’ee”, “suu”, “sa”, “sse” ,“nya” and measure> =1, delete the suffix. If measure =0, substitute the suffix with `(*glottal*).

The fourth cluster covers a special case
e.g.

“du”				
“di”	$\longrightarrow$	0	measure > =1	EndsWith B/G/D
“dan”	$\longrightarrow$	d	measure =0	
“wwan”	$\longrightarrow$	0	measure > =1	EndsWith VV

Explanation for the fourth rule cluster:

If a word ends with one of the suffixes: “du”, “di”, “dan” and measure> =1 and the remaining stem ends with B/G/D, delete the suffix. If measure =0, substitute the suffix with d.
If a word ends with the suffix “wwan” and measure> =1 and the remaining stem ends with double vowel delete the suffix.

The seventh cluster contain rules that conflate words formed by duplication of the first syllabus

R1	$\longrightarrow$	0	measure > =0	Startswith C	R1=R2
	$\longrightarrow$	0-1	measure > =0	Startswith C	C1+R1 =R2
C1+`	$\longrightarrow$	0	measure > =0	Startswith V	`+R1=R2

Explanation for the seventh rule cluster:
If R1 and R2 are the same delete R1.
If C1+R1 and R2 are the same delete R2.
If the word starts with vowel and if glota (`)+R1 equals R2 delete R2.
E.g. gaggabaaba(plural form of short),R1 is ga , R2 is gga and C1 is g. Therefore gaggabaaba is correctly stemmed to gabaaba by removing R2, since C1+R1 equals R2.

To stem a word the, the stemmer first checks if a given word is in the stop list or not. If found in the list, the word is excluded from further processing and nothing returned to the calling routine; stop and process the next word if any. If the word is not in the stop list, the word is checked for any match in the rule clusters. If a match is found, the respective action for that rule will be taken. As described earlier the rule has conditions like measure (the number of vowel consonant sequence), ending of the remaining stem with specific character, ending of the remaining stem with consonant, ending of the remaining stem with short or long vowel, matching of the ending of the word with one of the suffixes and the detail for each rule cluster is described in the privies section. The actions taken includes removing the suffixes, substituting the suffix with another one, removing of the reduplicated characters in the case of words formed by reduplication of some of the characters as described in the 7th rule cluster.

5.7. Evaluation of the Stemmer

In this report, error counting approach is adapted to evaluate the algorithm in terms of the number of accurately conflated results. The number of correctly conflated words and incorrectly conflated ones are counted for analysis. The output from the stemmer was then checked against the respective expected valid stem. These errors were then described in terms of under stemming and over stemming:

- Over stemming
  Over stemming occurs when too much of the term is removed.
- Under stemming
  Under stemming occurs when too little of the term removed.

Evaluation of the effectiveness of stemming algorithms is necessary to reveal specific error patterns. This information can subsequently be used to improve the algorithm where possible. Some error types, however, are inherent to the suffix-stripping method and without the additional information provided by, for instance, a dictionary, these errors cannot be avoided [8].

This stemmer is run on the test set of 5000 words which is assumed to be balanced. The literature from which the rule of the stemmer developed is totally different from the test set. This was done deliberately in order to predict the performance of the stemmer in the real world data.

The output from the stemmer indicates, Out of 5000 words 38 words (0.77%) were under stemmed and 220 words (4.39 %) were over stemmed. Totally this stemmer generates 258 words (5.16 %) stemming error. As a result, the accuracy of the stemmer becomes 94.84%.

In terms of compression, i.e., reduction of dictionary size, percentage of compression is calculated using the formula [9]:

$C = 100 * (W - S)/W$

Where,

C is the compression value (in percentage)

W is the number of the total words

S is a distinct stem after conflation

Accordingly,

Size of the data = 5000

Number of distinct stem after conflation = 1654

Hence, the percentage of compression for Afan Oromo text based on the evaluation text for this stemmer becomes 100 * (5000- 3100) / 5000 = 38%.

Reasons for the under stemming and over stemming problems are:

1. It was difficult to come up with the complete rule because of the complexity of the language. More conditions/rules are required based on more study of the morphology of the language.

2. Homographs are words which are spelled identically but nevertheless have a different meaning. The algorithm does not have access to information about, for instance, word categories; the different senses of these types of words are not distinguished.

3. The rule designed for the suffix *-s* is not general and conflates some terms incorrectly. Designing general rule for Words ending in *-s* is challenging.

4. Some Compound words are not conflated correctly. This stemmer didn`t include any rule that handles compound words. Even though there is no rule included to conflate compound words, rules that are designed for non compound words can be applied and produce correct result for most compound words. Examples of compound words that are conflated correctly are: *karadeemaa(*passenger) is correctly conflated to *"karadeem-*", *biyyalafaa*(world) is correctly conflated to "*biyyalaf-*". But *manabaate*(married(f)) is incorrectly conflated to "*manab-*".

5. *"ni"* and *"hin"* are considered as prefixes in some literatures and independent terms in other literatures. There fore, this stemmer didn`t include the rule that remove them when they appear as prefix. Both cases are possible in the language.

6. CONSLUSION & FUTURE WORK

Stemming is important for highly inflected languages such as Afan Oromo for many applications that require the stem of a word. In this work, a context sensitive rule based stemmer was developed that attempts to determine the stem of a word according to linguistic rules. According to the evaluation of the experiments, it can be concluded that an overall accuracy of about 94.84% is an encouraging figure. The proposed method generates some errors. Indeed, it is possible to anticipate such considerable contributions and positive effects of the stemmer since Afaan Oromo is one of the morphologically rich and complex languages. These errors were analyzed and classified into two different categories (under stemmed words and over stems). The error rate is about 4.27%. This shows that rule based stemmer can be performed with low error rates in high inflected languages such as Afan Oromo.

A big step in the future improvement of the Afan Oromo stemmer can be a study on how the word compounding and suffixes affect Afan Oromo words and their stems, and how one can include new rules that do not affect the effectiveness of the stemming process.

Besides, further study is required to increase the effectiveness of the rule based stemmer with no or little decrease in efficiency. Extracting other additional context sensitive conditions which is based on the study of the morphology of the language can increase the accuracy of the stemmer.

Moreover, the stemmer has to be tested with large amount of texts to prove its real performance. To succeed this we need to apply Afan Oromo stemmer in a web search engine, which retrieves information from Afan Oromo texts. Then we can have a complete view of the stemming system and the returned results after every search request. In this case we can do extended evaluation tests, we can measure the precision and recall in various texts and we can estimate the errors distribution in the stemming results.

All the rules described in this work can be a base for this further research and it can support extended stemming rules covering most of the terms in the Afan Oromo.

7. REFERENCES

[1] L. JB. "*Development of a stemming algorithm"*. Mechanical Translation and Computational Linguistics, 11: 22-31,(1968)

[2] G. Q. A. Oromoo. "*Caasluga Afaan Oromoo* Jildi I", Komishinii Aadaaf Turizmii Oromiyaa, Finfinnee, Ethiopia, pp. 105-220 (1995)

[3] M. F Porter. "*An algorithm for suffix stripping*". Program, 14(3):130–137,(1980)

[4] S. Jacques. "*Stemming of French Words Based on Grammatical Categories*." Journal of American Society for Information Science 44(1): 1-9, (1993)

[5] Census Report. "*Ethiopia's population now 76 million*". (2008) available at: http://ethiopolitics.com/news

[6] C. G. Mewis." *A Grammatical sketch of Written Oromo"*, Germany: Koln,pp. 25-99 (2001)

[7] K. K. Tune, V. Varma and P. Pingali. "*Evaluation of Oromo-English Cross-Language Information Retrieval*", Language Technologies Research Centre IIIT, Hyderabad India, (2007)

[8] W. Kraaij, R. Pohlmann."*Porter's stemming algorithm for Dutch*". Bioinformatics, 25: 1412–1418, (1997)

[9] L. Lessa. "*development of stemming algorithm for wolaytta text*", Masters Thesis Addis Ababa University, Fuculty of Informatics, Department of Information Science, (2003)

[10] W. Mekonen. "*Development of stemming algorithm for Affan Oromo anguage text*", MSc thesis faculty of informatics, Addis Ababa University, Addis Ababa,(2000)

[11] S. Dandapat, S. Sarkar and A. Basu. "*A Hybrid Model for Part-f-Speech Tagging and its Application to Bengali"*, Journal of world information society, 43(6):384–390, (2004)

XMODEL: An XML-based Morphological Analyzer for Arabic Language

Mourad Gridach mourad_i4@yahoo.fr
Faculty of Sciences of Fez, Mathematics and
Informatics Department Sidi Mohamed
Ben Abdellah University
Fez, 30000, Morocco

Noureddine Chenfour chenfour@yahoo.fr
Faculty of Sciences of Fez, Mathematics and
Informatics Department Sidi Mohamed
Ben Abdellah University
Fez, 30000, Morocco

Abstract

Morphological analysis is an essential stage in language engineering applications. For the Arabic language, this stage is not easy to develop because the Arabic language has some particularities such as the phenomena of agglutination and a lot of morphological ambiguity phenomenon. These reasons make the design of the morphological analyzer for Arabic somewhat difficult and require lots of other tools and treatments. The volume of the lexicon is another big problem of the morphological analysis of the Arabic Language which affects directly the process of the analyzing. In this paper we present a Morphological Analyzer for Modern Standard Arabic based on Arabic Morphological Automaton technique and using a new and innovative language (XMODEL) to represent the Arabic morphological knowledge in an optimal way. Both the Arabic Morphological Analyzer and Arabic Morphological Automaton are implemented in Java language and used XML technology. Buckwalter Arabic Morphological Analyzer and Xerox Arabic Finite State Morphology are two of the best known morphological analyzers for Modern Standard Arabic and they are also available and documented. Our Morphological Analyzer can be exploited by Natural Language Processing (NLP) applications such as machine translation, orthographical correction, information retrieval and both syntactic and semantic analyzers. At the end, an evaluation of Xerox and our system is done.

Key words: NLP, Morphology, Arabic Morphological Analyzer, Morphological Automaton, XMODEL language.

1. INTRODUCTION

Nowadays, Arabic language faces many challenges due to a lot of reasons such as the increase of the Arabic web sites, Arabic media, and Arabic companies around the world using the Arabic language, etc as [5]. For these reasons, a lot of research in the domain has been developed to satisfy the increasing demand of the applications using Arabic language. Arabic morphology is one of the essential needs in this domain and lots of morphological analyzers are available now, some of them have a commercial purpose and the others are available for research and evaluation as [4]. Buckwalter Arabic Morphological Analyzer and Xerox Arabic Finite State Morphology are two of the best known morphological analyzers for Modern Standard Arabic and they are also available and well documented.

The morphological analysis of Arabic is interested, as of other languages, in the structure of the word. But being given the wealth of the Arabic word's structure and the problem of agglutination, the operation becomes more complex than in the other languages as [13]. Another source of the difficulty is that sentences are longer and more complex compared to other languages. The average length of a sentence is 20 to 30 words, and it often exceeds 100 words. We also note that diacritics are particularities for our language and it is also considered as another source of difficulty for morphological analysis. For all these reasons seen so far, Arabic language is conceived as one of the languages that present a big problem in the morphological analysis and make this process very complicated.

In this article we'll be presenting our Arabic Morphological Analyzer based on morphological automaton developed using Java language. The use of this language has some advantages like openness, standardisation, flexibility, and reusability. To develop this morphological analyzer, we used the particularities of the Arabic language that is concretized on multilevel: verbs and nouns are also characterized by a specific representation named the matrix "root – scheme". This representation will help us construct a morphological automaton for the Arabic language. We note that we have used a new and innovative language (XMODEL) to represent the Arabic morphological knowledge. The use of this new language helps us to reduce the number of the entries in the lexicon. It also makes our system very flexible and one of the best existing morphological analyzers for the Arabic language.

The structure of the article is as follows. First, in this introduction we discuss some challenges of Arabic language and the importance of morphological analyzers in Natural Language Processing. After that, we present some morphological analyzers for Arabic language related to our work in the second section. Then in the third section, we explain our approach for the choice of the linguistic resource. In section four, we present our Arabic Morphological Analyzer. In section five, we evaluate our morphological analyzer. In section six, we discuss the obtained results. Finally, in the last section, we draw some conclusions and future works to be done.

2. WORKS IN THE DOMAIN

Morphological processing involves two different tasks according to the operation type: generation and analysis. In generation we produce correct forms using given morphemes, while in analysis we try to identify morphemes for a given word. A lot of research has been done in the development of morphological analyzers for Arabic; some of them are available for research and evaluation, while the rest have a commercial purpose.

2.1. Buckwalter Arabic Morphological Analyzer (2004)

This analyzer is considered as one of the most referenced in the literature, well documented and available for evaluation. It is also used by Linguistic Data Consortium (LDC) for POS tagging of Arabic texts, Penn Arabic Treebank, and the Prague Arabic Dependency Treebank. It can be freely download from this address: http://www.nongnu.org/aramorph/french/. It takes the stem as the base form and root information is provided. This analyzer contains over 77800 stem entries which represent 45000 lexical items. However, the number of lexical items and stems makes the lexicon voluminous and as result the process of analyzing an Arabic text becomes long.

2.2. Xerox Arabic Morphological Analysis and Generation

Xerox Arabic morphological Analyzer is well known in the literature and available for evaluation and well documented. This analyzer is constructed using Finite State Technology (FST). It adopts the root and pattern approach. Besides this, it includes 4930 roots and 400 patterns, effectively generating 90000 stems. The advantages of this analyzer are, on the one hand, the ability of a large coverage. On the other hand, it is based on rules and also provides an English glossary for each word. But the system fails because of some problems such as the overgeneration in word derivation, production words that do not exist in the traditional Arabic dictionaries and we can consider the volume of the lexicon as another disadvantage of this analyzer which could affect the process of analyzing.

2.3. Darwish's Sebawai Morphological Analyzer

Sebawai is an Arabic Morphological Analyzer developed by (Darwish, 2003) in one day. The author affirms that his morphological analyzer was built in less than 12 hours using about 200 lines of Perl code. The analyzer is based on automatically derived rules and statistics. The aim of this analyzer is the generation of the possible roots of any given Arabic word with a rate of success that reaches 84%. The advantage of Sebawai is the rapidity of the generation of roots. A disadvantage of this analyzer, however, is it's can't give information about the word analyzed which make it very limited for the most applications in the domain.

2.4. Attia's Morphological Analyzer

This morphological analyzer is built using Finite State Technology and it is suitable for both analysis and generation. It is based on contemporary data (a corpus of news articles of 4.5 million words), and takes the stem as the base form. It contains 9741 lemmas and 2826 multiword expressions. The advantage of this system is the treatment of multiword expressions (MWEs). The system can efficiently handle compound names of people, places, and organizations. A disadvantage of the system, however, is its limited coverage. It does not handle diacritized texts and targets a particular application (syntactic parser).

2.5. ElixirFM: an Arabic Morphological Analyzer by Otakar Smrz

ElixirFM is an online Arabic Morphological Analyzer for Modern Written Arabic developed by Otakar Smrz available for evaluation and well documented. This morphological analyzer is written in Haskell, while the interfaces in Perl. ElixirFM is inspired by the methodology of Functional Morphology (Forsberg and Ranta, 2004) and initially relied on the re-processed Buckwalter lexicon as [9]. It contains two main components: a multi- purpose programming library and a linguistically morphological lexicon as [14]. The advantage of this analyzer is that it gives to the user four different modes of operation (Resolve, Inflect, Derive and Lookup) for analyzing an Arabic word or text. But the system is limited coverage because it analyzes only words in the Modern Written Arabic.

3. LINGUISTIC RESOURCE REPRESENTATION: THE XMODEL LANGUAGE

So as to develop a morphological analyzer of the Arabic language, representing the morphological knowledge becomes very crucial. Besides this, it's viewed as one of the central problems of the automatic processing of the Arabic morphology.

According to some works, in order to represent the morphological knowledge of the Arabic language, they have chosen to use the database concept as a basic support to store the morphological information as [3], [9] and [12]. To seek any information, they make use of requests consulting. Unfortunately, this method remains very limited to this type of challenges. Consequently, they do not give good results.

A second method of representation, which is widely used, is offered by the artificial intelligence. Accordingly, a morphological analyzer serves as an intelligent system able to infer the morphological nature of the analysed sentence from a certain knowledge-base which constitutes of data and morphological rules as [15]. However, the artificial intelligence language is criticized for its being general and sequential search of information. The choice of the Lisp or Prolog language as a support of representing the morphological knowledge may not probably be the right option. This is due to the fact that the interpreter is not well adapted to this kind of problems.

We can also mention a third method known as the semantic networks. This method is also used in the artificial intelligence as a support of knowledge representation. The semantic networks, already developed in relation with psychology, correspond to a graphic representation composed of a set of items called nodes linked with arcs: the nodes stand for concepts, while the arcs stand for the relationship between the concepts.

To achieve a better representation of the morphological knowledge of Arabic, we conceived an innovative language adapted for this specific situation: it's the XMODEL language (XML-based MOrphological DEfinition Language). XMODEL is based on the XML language setting profits of its advantages and particularities. As a result, all morphological entries are gathered in an XMODEL files. Using the new language helps direct search for information and determinism. It also enables us to represent the whole components, properties and morphological rules with a very optimal way. To clarify the last point, we note that our morphological database contains 960 lexicon items (morphological components) and 455 morphological rules to be applied to these morphological components which present a remarkable reduction in the number of entries in the lexicon compared to the existing systems (Xerox and Buckwalter). This representation helps us achieve the following goals:

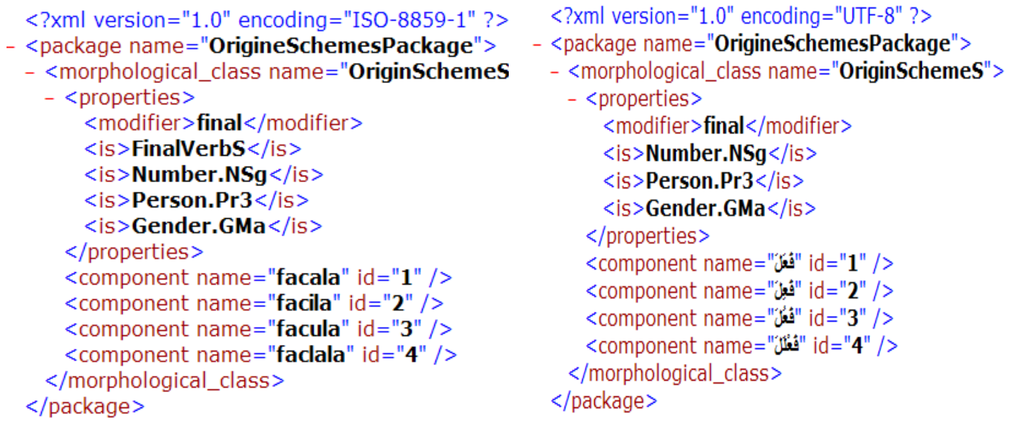
- ✓ A symbolic definition, declarative and therefore progressive of the Arabic morphology.
- ✓ A morphological database independent of processing that will be applied (see later).
- ✓ A considerable reduction of the number of morphological entries.
- ✓ The notion of scheme enables us to define the maximum morphological components by means of XMODEL language.

Our language makes it possible for us to represent the Arabic morphology as morphological classes and rules. Accordingly, our Arabic morphological database will be composed of three main parties: morphological classes, morphological properties and morphological rules.

Now let us first introduce the XMODEL language which permits to represent the morphological knowledge of Arabic and consists of three main parties. The first of which is:

3.1 Morphological components class

It enables us represent all morphological components of the Arabic language. It also permits to gather a set of morphological components having the same nature, the same morphological characteristics and the same semantic actions. Relying on the notion of scheme /*ealwazn*/ (الوزن), this class allows us a better optimization hence, a considerable reduction of morphological entries. By so doing, we needn't represent all the language items, but only their schemes. We note that our lexicon contains 960 items (morphological components) which is a remarkable reduction in the number of the items compared to the other dictionaries.



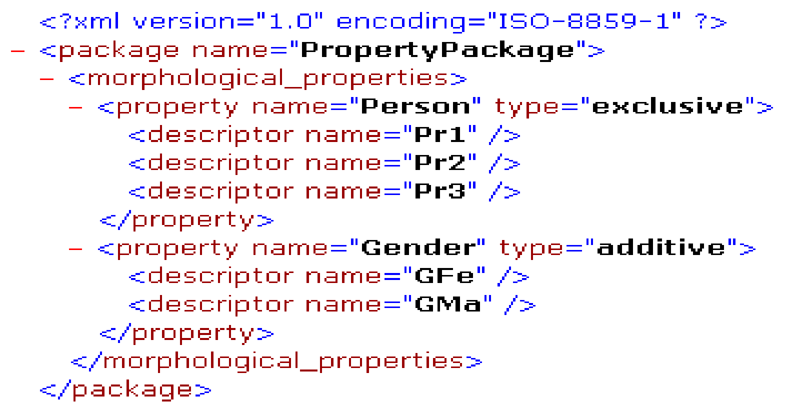
```
  <?xml version="1.0" encoding="ISO-8859-1" ?>
- <package name="OrigineSchemesPackage">
  - <morphological_class name="OriginSchemeS
    - <properties>
        <modifier>final</modifier>
        <is>FinalVerbS</is>
        <is>Number.NSg</is>
        <is>Person.Pr3</is>
        <is>Gender.GMa</is>
      </properties>
      <component name="facala" id="1" />
      <component name="facila" id="2" />
      <component name="facula" id="3" />
      <component name="faclala" id="4" />
    </morphological_class>
  </package>
```

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <package name="OrigineSchemesPackage">
  - <morphological_class name="OriginSchemeS">
    - <properties>
        <modifier>final</modifier>
        <is>Number.NSg</is>
        <is>Person.Pr3</is>
        <is>Gender.GMa</is>
      </properties>
      <component name="فَعَلَ" id="1" />
      <component name="فَعِلَ" id="2" />
      <component name="فَعُلَ" id="3" />
      <component name="فَعْلَلَ" id="4" />
    </morphological_class>
  </package>
```

FIGURE 1: Representation of some verbs schemes using XMODEL language

3.2 Morphological properties class

It permits to characterize the different morphological components represented by the morphological class: a morphological property class contains a set of morphological descriptors or morphological values of properties that would be assigned to the different morphological components. We mention, for example, the property "*Gender*" which will distinguish between masculine and feminine components. The morphological properties are not related to a specific morphological class which makes it necessary to define them outside the morphological classes.



```
  <?xml version="1.0" encoding="ISO-8859-1" ?>
- <package name="PropertyPackage">
  - <morphological_properties>
    - <property name="Person" type="exclusive">
        <descriptor name="Pr1" />
        <descriptor name="Pr2" />
        <descriptor name="Pr3" />
      </property>
    - <property name="Gender" type="additive">
        <descriptor name="GFe" />
        <descriptor name="GMa" />
      </property>
    </morphological_properties>
  </package>
```

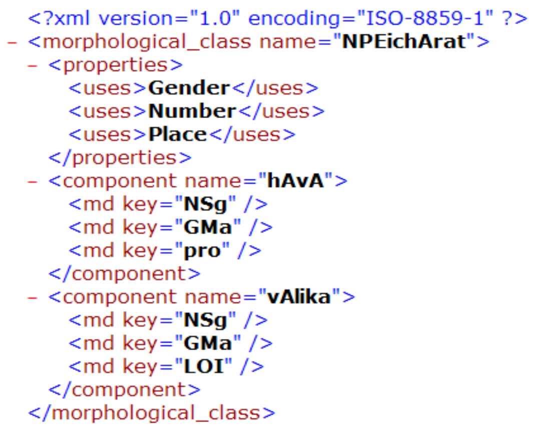
FIGURE 2: Representation of morphological properties "*Person*" and "*Gender*"

We have added the attribute "*type*" to work out the problem of the semantic of the morphological descriptors that might be **exclusive** (the morphological component can not be characterized by the morphological descriptors of the same property as in the case of the "*Person*" property) or **additive** (the morphological component can be characterized by the morphological descriptors of the same property as it is the case in the "*Gender*" property).

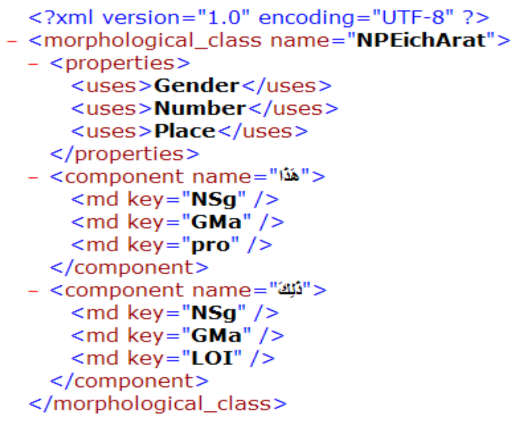
There are two strategies to characterize the morphological components using the properties:

3.2.1 Property of components

A morphological class can use a list of morphological descriptors to define its components generally speaking; each morphological component can have its own morphological descriptors. As for the "*Gender*" property, some components of this class can be masculine while the others can be feminine. This type of properties is named the **property of components**. In order to put them into practice, we have introduced the "*uses*" tag. This means that the different morphological descriptors defined by the property of components can be used by the different morphological components of the morphological class.



```
  <?xml version="1.0" encoding="ISO-8859-1" ?>
- <morphological_class name="NPEichArat">
  - <properties>
      <uses>Gender</uses>
      <uses>Number</uses>
      <uses>Place</uses>
    </properties>
  - <component name="hAvA">
      <md key="NSg" />
      <md key="GMa" />
      <md key="pro" />
    </component>
  - <component name="vAlika">
      <md key="NSg" />
      <md key="GMa" />
      <md key="LOI" />
    </component>
  </morphological_class>
```

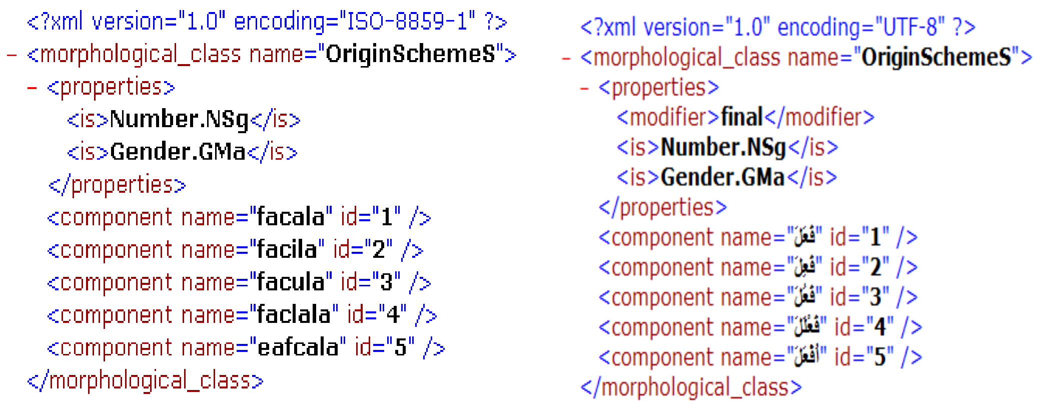


```
  <?xml version="1.0" encoding="UTF-8" ?>
- <morphological_class name="NPEichArat">
  - <properties>
      <uses>Gender</uses>
      <uses>Number</uses>
      <uses>Place</uses>
    </properties>
  - <component name="هذا">
      <md key="NSg" />
      <md key="GMa" />
      <md key="pro" />
    </component>
  - <component name="ذلك">
      <md key="NSg" />
      <md key="GMa" />
      <md key="LOI" />
    </component>
  </morphological_class>
```

FIGURE 3: The property of components (« *Gender* » « *Number* » and « *Place* ») characterizing the components "*hAvA*" and "*vAlika*".

3.2.2 Property of classes

This one requires assigning a set of morphological components the commons morphological properties. For example, all components are masculine names. This type of property is known as **property of classes**. To realize this, we introduce the "*is*" tag.



```
  <?xml version="1.0" encoding="ISO-8859-1" ?>
- <morphological_class name="OriginSchemeS">
  - <properties>
      <is>Number.NSg</is>
      <is>Gender.GMa</is>
    </properties>
    <component name="facala" id="1" />
    <component name="facila" id="2" />
    <component name="facula" id="3" />
    <component name="faclala" id="4" />
    <component name="eafcala" id="5" />
  </morphological_class>
```

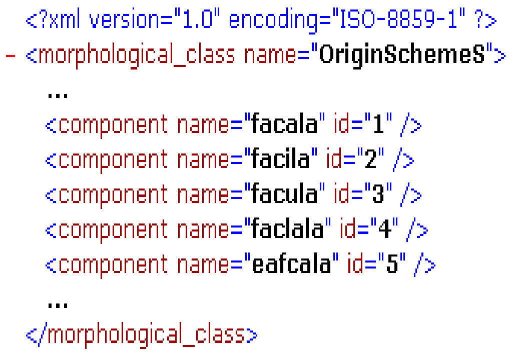
```
  <?xml version="1.0" encoding="UTF-8" ?>
- <morphological_class name="OriginSchemeS">
  - <properties>
      <modifier>final</modifier>
      <is>Number.NSg</is>
      <is>Gender.GMa</is>
    </properties>
    <component name="فَعَلَ" id="1" />
    <component name="فَعِلَ" id="2" />
    <component name="فَعُلَ" id="3" />
    <component name="فَعْلَلَ" id="4" />
    <component name="أَفْعَلَ" id="5" />
  </morphological_class>
```

FIGURE 4: Example of the property of classes

In the above example, all the schemes are singular components and masculine gender. It becomes evident to mention that the same class of the morphological components can use one combination of the tags "*uses*" and "*is*".

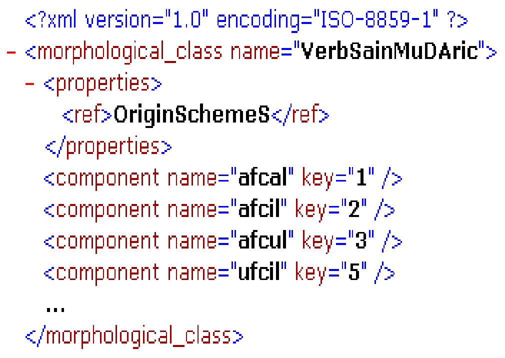
3.2.3 Property of reference

Another strong point of the XMODEL language is the introducing of the notion of **property of reference** which has an important role to benefit from the specificities of the Arabic morphology. As for the Arabic language some morphological components might be conjugated forms of other components which we call original components. An example of this is the case of the following components "*afcalu*", "*afcilu* ", "*afculu*". These components are all conjugated forms of the component "*facala*". We have specified this link of reference between the components using the "*ref*" tag.



```
  <?xml version="1.0" encoding="ISO-8859-1" ?>
- <morphological_class name="OriginSchemeS">
    ...
    <component name="facala" id="1" />
    <component name="facila" id="2" />
    <component name="facula" id="3" />
    <component name="faclala" id="4" />
    <component name="eafcala" id="5" />
    ...
  </morphological_class>
```

FIGURE 5 : Example of some verbs schemes



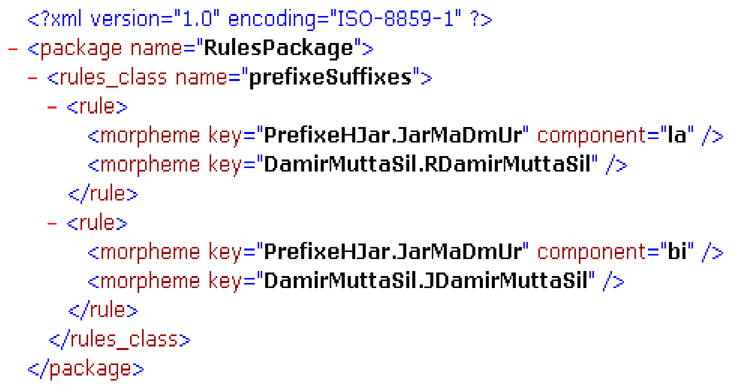
```
  <?xml version="1.0" encoding="ISO-8859-1" ?>
- <morphological_class name="VerbSainMuDAric">
  - <properties>
      <ref>OriginSchemeS</ref>
    </properties>
    <component name="afcal" key="1" />
    <component name="afcil" key="2" />
    <component name="afcul" key="3" />
    <component name="ufcil" key="5" />
    ...
  </morphological_class>
```

FIGURE 6 : The conjugated forms of some verbs

In order to concretize this reference between the components, we have opted the attribute "*id*" to the original component. This attribute is specified in the "*component*" tag. The components that are conjugated forms will use this code as an attribute of that tag (the "*key*" attribute) to indicate this reference.

3.3 Morphological rules class

Firstly, it should be noted that we developed 455 morphological rules for the Arabic language. They help us combine some morphological components (morphemes) together to generate correct language words. They use the different morphological components classes as well as the morphological properties classes. The morphological rules classes allow us the possibility to add new morphological descriptors which do not belong to the union of morphological descriptors of components of rules. As a result, they are considered as a generator of language words. The implementation of the morphological rules class permits to put into practice all the possible concatenations between components.



```
  <?xml version="1.0" encoding="ISO-8859-1" ?>
- <package name="RulesPackage">
  - <rules_class name="prefixeSuffixes">
    - <rule>
        <morpheme key="PrefixeHJar.JarMaDmUr" component="la" />
        <morpheme key="DamirMuttaSil.RDamirMuttaSil" />
      </rule>
    - <rule>
        <morpheme key="PrefixeHJar.JarMaDmUr" component="bi" />
        <morpheme key="DamirMuttaSil.JDamirMuttaSil" />
      </rule>
    </rules_class>
  </package>
```

FIGURE 7: Class of rules which represent the components prefixed by the prefix "*la*" and "*bi*"

In the above example, this rule permits to generate the components which begin by the prefix "*la*" and the prefix "*bi*".

In a morphological rule, morphemes can have properties of different natures. Therefore, it is necessary to define with the rule of concatenation the strategy to manipulate the morphological descriptors of different morphemes. There are three possible strategies:

- *additive*: the gathering of the different morphological descriptors is assigned to the result word.
- *exclusive*: the concatenation doesn't take in any account the morphological properties of the components. Only the morphological descriptors of the rule are taken in account.
- *inclusive*: in this case, the same descriptors of the different morphemes are assigned to the concatenation.

The structuring of our morphological database using XMODEL language allows us to generate the morphological automaton of the Arabic language. In the next section, we will be dealing with the notion of morphological automaton.

4. SYSTEM DESCRIPTION

In this part, we describe the Arabic morphological analyzer. This latter is based on using morphological automaton technology. The implementation of each morphological analyzer for any language needs a main resource. This resource is the morphological database, so the first task is the conception and the realization of a morphological database. We also used a new language, XMODEL language, to create this database. After that, the second task is the development of a set of morphological automatons for Arabic language each of which represents a very definite category of morphology.

4.1. Arabic Morphological Automaton

To implement a morphological analyzer for any language, especially Arabic language, the use of morphological automaton is considered among the most efficient methods. It can be used for both analysis and generation. This latter is based on the notion of Finite State Automaton (DFA). A word is accepted by the morphological automaton if it belongs to a correct word in Arabic and rejected in the contrary case. Generally, the Arabic morphological automaton will have the following features:

- Q is a finite set of states of the control unit which represents the states of a morphological automaton.
- ∑ is a finite input tape alphabet symbols. Concerning morphological automaton for Arabic, it is constituted of the alphabets of Arabic language (Refer to figure1 in the Appendix to find all the Arabic alphabets).
- q_0 is the start state of the morphological automaton. It is constituted of only one start state in the case of a morphological automaton.
- F is a subset of Q. It also represents the accepting states of the morphological automaton for Arabic. This latter has a very important role because it permits to give us a set of information of the Arabic words analyzed. This information is called the morphological descriptors and they also characterize these words.
- The set τ also represents the transition function of the morphological automaton.

Consequently, the building of the morphological automaton of the Arabic language needs to use the XMODEL database discussed before. We have to extract all the morphological rules from this database and construct a morphological automaton of each rule. So to realize that constructing, we have to use some automaton operations such as concatenation and union operation. Let us clarify how we can use these two operations to generate a morphological automaton for a definite morphological rule. So, we consider the following rule:

```
  <?xml version="1.0" encoding="ISO-8859-1" ?>
- <package name="RulesPackage">
  - <rules_class name="cardNbCRules">
    - <rule id="rule_1">
        <morpheme key="CardNumber.CNAccepteSCID.CNAccepteSC" />
        <morpheme key="CasSuffixe.SCD" component="u" />
        <idp name="CNADefMarfUc" />
      </rule>
      .....
    </rules_class>
  </package>
```

So to generate the morphological automaton which represent this rule, we have to use the operation of concatenation to concatenate the first morpheme (key = "*CardNumber.CNAccepteSCID.CNAccepteSC*") with the second one (key = "*CasSuffixe.SCD*" component = "*u*"). Therefore, the morphological automaton that represents this morphological rule is:

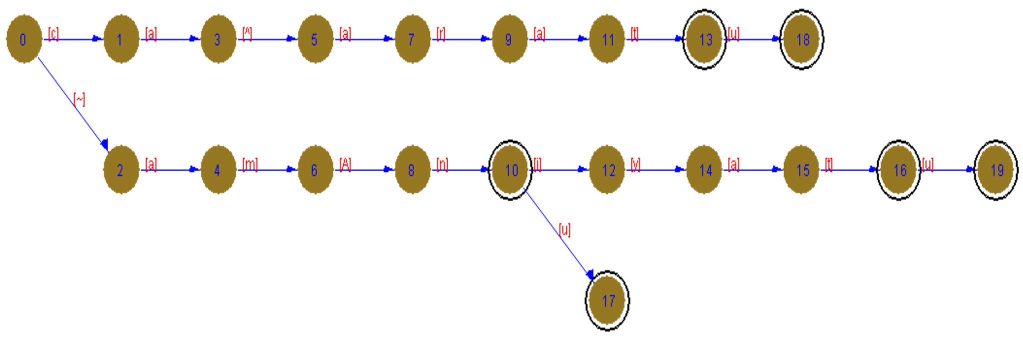


FIGURE 8: A morphological automaton representing the above morphological rule

In addition to the operation of concatenation used to concatenate morphemes or morphological automatons together, we used the union operation to associate two or several morphological automatons generated by the first operation, each one represent a definite morphological rule. To concretize the use of this second operation, let us consider the following class of morphological rules:

```
    <?xml version="1.0" encoding="ISO-8859-1" ?>
  - <package name="RulesPackage">
    - <rules_class name="cardNbCRules">
      - <rule id="rule_1">
          <morpheme key="CardNumber.CNAccepteSCID.CNAccepteSC" />
          <morpheme key="CasSuffixe.SCD" component="u" />
          <idp name="CNADefMarfUc" />
        </rule>
      - <rule id="rule_2">
          <morpheme key="CardNumber.CNAccepteSCID.CNAccepteSC" />
          <morpheme key="CasSuffixe.SCD" component="a" />
          <idp name="CNADefManSUb" />
        </rule>
        .....
      </rules_class>
    </package>
```

In the above example, we have two morphological rules; each one generates a morphological automaton. We used the union operation to associate the first automaton which represents the rule identified by "*rule_1*" with the second automaton which represents the rule identified by "*rule_2*". The result morphological automaton is:

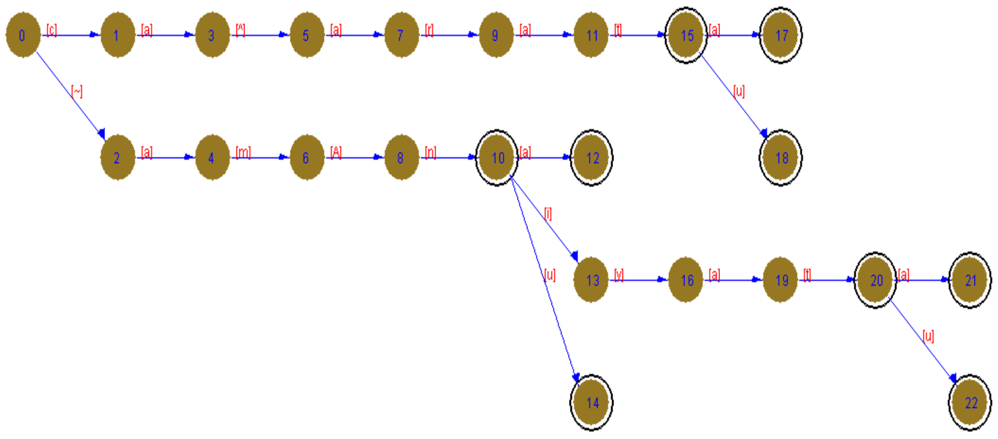


FIGURE 9: A morphological automaton representing the above rule

In the following paragraphs, we present a detail of how to construct all the morphological automatons of the Arabic language and the technical used in this constructing.

So as to build a morphological automaton of the Arabic language, we have classified words of the Arabic language in to two categories: the first category is that which submits to the derivation process, while the second one doesn't. This process of derivation is generated by a set of morphological rules known in the Arabic grammar under the name "*qawAcidu eaSSarfi*" /قواعد الصرف/. They repose on the manipulation of a set of very determined schemes named "*ealeawzAn*" /الأوزان/.

A scheme "*ealwazn*" /الوزن/ is an abstract linguistics term that represents a family of varied derived words; these words might be verbs or derived nouns which share the same linguistic features as [16]. At the graphical level, a scheme generally constitutes of:

- Three main consonants that are represented by the letters "f" /ف/, "c" /ع/ and "l" /ل/ with a possibility to duplicate the last letter "l" as in the case of schemes that correspond to a four letters root like "*faclala*" /فعلل/.
- Some consonants that serve as tools to extend the root like "*stafcala*" /استفعل/ and "*tafAcala*" /تفاعل/.
- A group of adequate vowels.

We have grouped in the first category of words the following items:

- Derived nouns "*ealaSmAe ealmu^taqqa*" /الأسماء المشتقة/.
- Strong verbs "*ealeafcAl eaSSaHiyHa*" /الأفعال الصحيحة/: these are the verbs that contain no weak letters. In the Arabic language, there are three weak letters: "w" /و/, "A" /ا/ and "y" /ي/.
- Weak verbs "*ealeafcAl ealmuctalla*" /الأفعال المعتلة/: these are the verbs that contain a weak letter. Weak verbs are also classified into three categories (ATTIA, 2005):

  a. Assimilated "*ealmi~al*" /المثال/: a verb that contains an initial weak letter.
  b. Hollow "*ealajwaf*" /الأجوف/: a verb that contains a middle weak letter.
  c. Defective "*eannAqiS*" /الناقص/: a verb that contains a final weak letter.

While the second category of words contains three families of words:

- The particular nouns "*ealasmAe ealxASSa*" /الأسماء الخاصة/: these nouns comprise proper nouns, names of cities, names of countries, etc. It also regroups the exclusive nouns "*easmAe ealeisti~nAe*" /أسماء الاستثناء/, the interrogative nouns "*easmAe ealeistifhAm*" /أسماء الاستفهام/, the demonstrative nouns "*easmAe ealei^Ara*" /أسماء الإشارة/, the conditional nouns "*easmAe ea^^art*" /أسماء الشرط/, etc.
- The particles "*ealHurUf*" /الحروف/ likes for example "*HurUfu ealjarri*" /حروف الجر/, "*HurUfu ealjazmi*" /حروف الجزم/, "*HurUfu ealcaTfi*" /حروف العطف/, etc.
- The incomplete verbs "*ealeafcAl eannAqiSa*" /الأفعال الناقصة/: this family of verbs contains the family of verb "*kAda*" /كاد/, the family of verb "*kAna*" /كان/ and the family of verb "*Zanna*" /ظن/.

Finally, after the constructing of our morphological automaton which contains the Arabic vocalized, its size is about 120 MB. We note that developing the morphological automatons of the Arabic language is the main idea of constructing a morphological analyzer for our language. So, in the following paragraph we will present our morphological analyzer for the Arabic language.

4.2. The Arabic Morphological Analyzer

First of all, because our morphological analyzer is based on the morphological automaton which is the main idea of this work, so this part it will be short than the part of the Arabic morphological automaton. So, in this section we describe our morphological analyzer for Arabic language. This analyzer is developed using three principal components:

- A morphological database constructed using the XMODEL language based on XML language integrating all the data suitable for Arabic language. Its regroups three packages: package of morphological components that contains verbs, nouns, particles and affixes. The second package includes the morphological rules and the last package is concerned with the morphological properties.
- A set of morphological automatons for the Arabic language each of which represents a very specific morphological category.
- A program handling the morphological database and the morphological automaton. It is developed through the use of Java language.

In addition, our morphological analyzer is meant to give a set of information about any Arabic word given to it. This set of information is about:

- The gender of the word: masculine or feminine.
- The person of the word: first, second or third person.
- The number of the word: singular, dual or plural.
- The case of the word: "*marfUc*" (مرفوع), "*manSUb*" (منصوب), "*majrUr*" (مجرور), "*majzUm*" (مجزوم).
- The type of the word: verb, noun or particle.

- If the word is a verb, we give its tense: present (“*ealmuDAric*”: المضارع), past (“*ealmADI*”: الماضي) or imperative (“*ealeamr*”: الأمر). We also give its voice: active or passive.
- The origin scheme of the word is given if available.

We note that this set of information has an important role especially in future works like for example the building of a syntactic analyzer, a semantic analyzer, machine translation, etc.

Finally, the development of our morphological analyzer for Arabic language has many advantages such as:

- The separation between the task of the linguist and the developer.
- We can also reuse our programs in future works.
- Development standardization means in our application that we have build all the applications with the same standards, technologies (Java language, XML technologies, SAX, DOM, etc).
- Our morphological analyzer is developed using Java language. Therefore, our analyzer can be run in any platform such as Windows, Linux, UNIX and Mac OS.
- The facility of maintenance.

5. EVALUATION

In this section, we are going to evaluate Xerox Arabic Morphological Analysis and Generation and our Morphological Analyzer for the Arabic Language. We note that a standard annotated corpus for Arabic language is not yet available, for this reason the process of evaluation will be difficult. So, we have chosen Xerox Morphological Analysis and Generation because it is one of the best known morphological analyzers for Modern Standard Arabic and it is also available and well documented.

On the one hand, the first remark when we compare the two morphological analyzers is about the information giving by each one. Used an innovative language (XMODEL) for representing the morphological knowledge and the notion of the Morphological Automaton for Arabic Language, our morphological analyzer gives more information about each word analyzed and more precision compared to Xerox Arabic Morphological Analyzer. To clarify this point, let us take some Arabic words and try to analyze them using the two morphological analyzers:

The word	**Morphological Analysis using Xerox Arabic Morphological Analyzer**
صِفْرٌ [Sifrun]	CiCoC Noun +N Indef Nom
خَارِجُونَ [xArijUna]	CACiC participle Active +U3na Masc Plur Nom
مُرْتَدِّي [murtaddI]	muCtaCaC Participle Passive +I3 Ma Plur Acc/Gen Possessive
فُصِلْتُ [fuSiltu]	+tu 1stPer Masc/Fem Sing CuCiC Verb
أُخْرِجْتُمَا [euxrijtumA]	uCoCiC Verb +tumA 2ndPer Masc/Fem Dual
مَعَ [maca]	maEa Funcwa
أَمَامَ [eamAma]	CaCAC Noun +a Def Acc
العَاشِرَ [ealcA^ira]	al Article CACiC Noun +a Def Acc
بِهِمَا [bihimA]	bi +himA Funcwa
يُجَادِلُونَ [yujAdilUna]	yu Imperfect Prefix CACiC Verb +Una Indicative 3rdPer Masc Plur

TABLE 1: Words analyzed using Xerox Arabic Morphological Analyzer

The word	Morphological Analysis using our Arabic Morphological Analyzer
صِفْرٌ [Sifrun]	Gma Noun V0 Ind Raf [un]
خَارِجُونَ [xArijUna]	facala facila facula fAcil Gma NPl Noun efc Raf [Una]
مُرْتَدِّي [murtaddI]	eifcalla mufcallI Gma Pr1 NDl NSg Noun emf mmi8 KaS [I]
فُصِلْتُ [fuSiltu]	facula facala facila Gfe Gma Pr1 NSg Verb [tu]
أُخْرِجْتُمَا [euxrijtumA]	eafcala Gfe Gma Pr2 NDl Verb [tumA]
مَعَ [maca]	Particle zam mak Def NaS [a]
أَمَامَ [eamAma]	Particle mak Def NaS [a]
العَاشِرَ [ealcA^ira]	Noun V10 Def NaS [eal] [a]
بِهِمَا [bihimA]	Gfe Gma Pr3 NPl KaS [bi] [himA]
يُجَادِلُونَ [yujAdilUna]	fAcala Gma Pr3 NPl Verb Raf [yu] [Una]

TABLE 2: Words analyzed using our Morphological Analyzer

Related to the two tables above which represented the results of ten different Arabic words analyzed using the two morphological analyzers, we note that our morphological analyzer provides more information and more precision about the word analyzed compared to Xerox Morphological Analyzer. Thanks to our new innovative language (XMODEL) which permit to represent the morphological knowledge in an optimal way and the power of the morphological automaton for Arabic. This advantage will be very useful especially in the future works which will be done later. It should be noted that our system could provide more information about the word analyzed according to the user needs.

On the other hands, let us see the evaluation process from another view. So, we have selected a corpus of 975 words containing different type of the word in Arabic (verbs, nouns and particles). Then, we tested them on each morphological analyzer, and after that we draw a detailed analysis for the two analyzers. Our corpus contains 975 words divided into 481 nouns, 362 verbs and 132 particles. The table below shows the number of words which are not found when they are analyzed using the two morphological analyzers:

Type of the word	The number	Xerox Morphological Analyzer	Our System
Nouns	481	39	21
Verbs	362	16	-
Particles	132	29	-
Total	975	84	21

TABLE 3: Results of the evaluation process

To conclude this part of evaluation, using a new innovative language (XMODEL) and the notion of morphological automaton, our morphological analyzer can reach an average of performance around 95% which will make it one of the best existing morphological analyzers for the Arabic language and it will be very useful for the next future works to be done in NLP. We note that an update of our morphological database could resolve these errors seen in the table above. Represented as a set of XMODEL files, the process of updating the morphological database becomes very easy which make our innovative language one of the best languages to represent the Arabic morphological knowledge.

6. DISCUSSION

To compare our morphological analyzer for the Arabic language to the other existing systems, the task is difficult to do because there is no standard to make this comparison and every system has its own target. For this reason, each analyzer has some advantages and disadvantages comparing to the others.

Our morphological analyzer for the Arabic language has some advantages comparing to the others analyzers. These advantages are:

- Our morphological analyzer can be used in both analysis and generation
- It handles diacritized texts which permit to reduce the rate of ambiguity
- Our new and innovative language (XMODEL) used for the representation of the morphological knowledge and the use of morphological automaton for the Arabic Language permit to avoid a huge problems of ambiguity in the Arabic language which the most analyzers can't resolve
- The use of XMODEL language permit to reduce the number of the entries in the morphological database which present a big problem of the other morphological analyzers
- Represented as a set of XMODEL files, the process of updating the Arabic morphological database is very easy to develop. This advantage makes our system very flexible and one of the best existing morphological analyzers
- The major advantage of our system is that it permits, on the one hand, to give the affixes, the stem and the scheme (if the word is a noun or a verb and if it has a scheme) for any word given. On the other hands, it gives the information about the word analyzed using a list of morphological descriptors which permit to characterize every Arabic word

Our system has some disadvantages compared to some other systems. Firstly, it can't handle undiacritized texts. Secondly, it handles words which not exist in the Arabic language and finally, it doesn't provide an English glossary.

7. CONCLUSION & FUTURE WORKS

In this paper, we presented a Morphological Analyzer for Arabic language which is developed using a morphological database realized using XMODEL language and a set of morphological automatons for Arabic. The use of these automatons makes the system very efficient and fast. Another strong point of our morphological analyzer is the portability and the reusability because we have used Java for the development and the XML technology.

To extend our platform, we can also think to develop some works in the future:

- Develop the syntactic analyzer.
- Develop the semantic analyzer.
- Develop a system for Arabic learning.
- The help of the correction and the generation of texts.
- Automatic understanding of the texts: classification of texts, automatic summary and automatic extraction of the key words.
- Realize some specific applications such as machine translation, Q/R systems, Information Retrieval systems, etc.

Appendix

ا	:	A		س	:	s		ك	:	k
ب	:	B		ش	:	^		ل	:	l
ت	:	T		ص	:	S		م	:	m
ث	:	~		ض	:	D		ن	:	n
ج	:	J		ط	:	T		هـ	:	h
ح	:	H		ظ	:	Z		و	:	w
خ	:	X		ع	:	c		ي	:	y
د	:	D		غ	:	g		ى	:	A
ذ	:	V		ف	:	f		ة	:	t
ر	:	r		ق	:	q		ء	:	e
ز	:	z								

Figure 1: Letter mappings

The short vowels	The long vowels
a : indicate the "*fatHa*" u : indicate the "*Damma*" i : indicate the "*kasra*"	A : represents the long vowel "ا" U : represents the long vowel "و" I : represents the long vowel "ي"

REFERENCES

1. Abouenour L., EL Hassani S., Yazidy T., Bouzouba K., Hamdani A. *"Building an Arabic Morphological Analyzer as part of an Open Arabic NLP Platform"*. In the Language Resources and Evaluation Conference (LREC), Marrakech, Morocco, 31st May, 2008

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

3. Attia M. (2005). *"Developing a Robust Arabic Morphological Transducer Using Finite State Technology"*. 8th Annual CLUK Research Colloquium. Manchester, UK

4. 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

5. Atwell E., Al-Sulaiti L., Al-Osaimi S., Abu Shawar B.. (2004). *"Un Examen d'Outils pour l'Analyse de Corpus Arabes"*. JEP-TALN 04, Arabic Language Processing, Fès, 19-22 April 2004

6. Beesley KR (1996). *"Arabic Finite-State Morphological Analysis and Generation"*. Proceedings of the 16th conference on Computational linguistics, Vol 1. Copenhagen, Denmark: Association for Computational Linguistics, pp 89-94

7. Beesley KR. 1998b. "*Arabic morphology using only finite-state operations*". In Michael Rosner, editor, Computational Approaches to Semitic Languages: Proceedings of the Workshop, pages 50–57, Montreal, Quebec, August 16. Université de Montreal

8. Beesley KR (2000). *"Finite-State Non-Concatenative Morphotactics"*. SIGPHON-2000, Proceedings of the Fifth Workshop of the ACL Special Interest Group in Computational Phonology, p. 1-12, August 6, 2000, Luxembourg

9. Buckwalter T. (2002). *"Buckwalter Arabic Morphological Analyzer Version 1.0. Linguistic Data Consortium"*. University of Pennsylvania, LDC Catalog No.: LDC2002L49

10. Darwish K (2002). *"Building a Shallow Morphological Analyzer in One Day"*. Proceedings of the workshop on Computational Approaches to Semitic Languages in the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02). Philadelphia, PA, USA

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

12. Mars M., Belgacem M. (2006). *"Developed of a morphological analyser for Arabic language, tool for creation of educational activities of training of Arabic"*. Workshop "TEL in working context", 13-15 November 2006, Grenoble, France. 2006

13. Mars M., Belgacem M., Zrigui M., Antoniadis G., (2007). *"Analyseur morphologique de l'arabe"*. CITALA2007, 18-19 juin 2007, Rabat, Maroc

14. Otakar Smrz. ElixirFM. *"Implementation of Functional Arabic Morphology"*. In ACL 2007 Proceedings of the Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources, pages 1–8, Prague, Czech Republic, 2007.

15. Shaalan K. *"Extending Prolog for Better Natural Language Analysis"*. In Proceeding of the 1st Conference on Language Engineering, Egyptian Society of Language Engineering (ELSE), PP. 225-236, Egypt, March 14-15, 1998.

16. Tahir Y., Chenfour N., Harti M., *"Modélisation à objets d'une base de données morphologique pour la langue arabe"*. JEP-TALN 2004, Traitement Automatique de l'Arabe, Fès, 20 avril 2004

CALL FOR PAPERS

Journal: International Journal of Computational Linguistics (IJCL)
Volume: 1 **Issue:** 3
ISSN: 2180-1266
URL: http://www.cscjournals.org/csc/description.php?JCode=IJCL

About IJCL

Computational linguistics is an interdisciplinary field dealing with the statistical and/or rule-based modeling of natural language from a computational perspective. Today, computational language acquisition stands as one of the most fundamental, beguiling, and surprisingly open questions for computer science. With the aims to provide a scientific forum where computer scientists, experts in artificial intelligence, mathematicians, logicians, cognitive scientists, cognitive psychologists, psycholinguists, anthropologists and neuroscientists can present research studies, International Journal of Computational Linguistics (IJCL) publish papers that describe state of the art techniques, scientific research studies and results in computational linguistics in general but on theoretical linguistics, psycholinguistics, natural language processing, grammatical inference, machine learning and cognitive science computational models of linguistic theorizing: standard and enriched context free models, principles and parameters models, optimality theory and researchers working within the minimalist program, and other approaches. IJCL is a peer review journal and a bi-monthly journal.

To build its International reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJCL.

IJCL List of Topics

The realm of International Journal of Computational Linguistics (IJCL) extends, but not limited, to the following:

- Computational Linguistics
- Computational Models
- Computational Theories
- Corpus Linguistics
- Formal Linguistics-Theoretic and Grammar Induction
- Information Retrieval and Extraction
- Language Generation
- Language Learning
- Linguistics Modeling Techniques
- Linguistics Theories
- Machine Translation
- Models of Language Change and its Effect on Lingui
- Models that Address the
- Models that Combine Linguistics

Acquisition of Word-order
- Models that Employ Statistical/probabilistic Gramm
- Natural Language Processing
- Speech Analysis/Synthesis
- Spoken Dialog Systems

Parsing
- Models that Employ Techniques from machine learnin
- Quantitative Linguistics
- Speech Recognition/Understanding
- Web Information Extraction/Mining

Important Dates

Volume: 1
Issue: 3
Paper Submission: September 30 2010
Author Notification: November 01, 2010
Issue Publication: November / December 2010

Contact Information

Computer Science Journals Sdn BhD
M-3-19, Plaza Damas Sri Hartamas
50480, Kuala Lumpur MALAYSIA

Phone: +603 6207 1607
+603 2782 6991
Fax: +603 6207 1697

BRANCH OFFICE 1
Suite 5.04 Level 5, 365 Little Collins Street,
MELBOURNE 3000, Victoria, AUSTRALIA

Fax: +613 8677 1132

BRANCH OFFICE 2
Office no. 8, Saad Arcad, DHA Main Bulevard
Lahore, PAKISTAN

EMAIL SUPPORT
Head CSC Press: coordinator@cscjournals.org
CSC Press: cscpress@cscjournals.org
Info: info@cscjournals.org

COMPUTER SCIENCE JOURNALS SDN BHD

M-3-19, PLAZA DAMAS

SRI HARTAMAS

50480, KUALA LUMPUR

MALAYSIA