Volume 5 ▪ Issue 2 ▪ May / June 2014

INTERNATIONAL JOURNAL OF COMPUTATIONAL LINGUISTICS (IJCL)

Publication Frequency: 6 Issues / Year

ISSN : 2180-1266

CSC PUBLISHERS
http://www.cscjournals.org

INTERNATIONAL JOURNAL OF COMPUTATIONAL LINGUISTICS (IJCL)

VOLUME 5, ISSUE 2, 2014

EDITED BY
DR. NABEEL TAHIR

ISSN (Online): 2180 - 1266
International Journal of Computational Linguistics (IJCL) is published both in traditional paper form and in Internet. This journal is published at the website http://www.cscjournals.org, maintained by Computer Science Journals (CSC Journals), Malaysia.

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

INTERNATIONAL JOURNAL OF COMPUTATIONAL LINGUISTICS (IJCL)

Book: Volume 5, Issue 2, May / June 2014
Publishing Date: 01-06-2014
ISSN (Online): 2180-1266

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

© IJCL Journal
Published in Malaysia

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

CSC Publishers, 2014

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 *Five* 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)

EDITORIAL BOARD

TABLE OF CONTENTS

Volume 5, Issue 2, May / June 2014

Design of A Spell Corrector For Hausa Language

Lawaly Salifou *salifoumma@yahoo.fr*
Département de Mathématiques et Informatique
Faculté des Sciences et Techniques
Université Abdou Moumouni
Niamey, BP 10662, NIGER

Harouna Naroua *hnaroua@yahoo.com*
Département de Mathématiques et Informatique
Faculté des Sciences et Techniques
Université Abdou Moumouni
Niamey, BP 10662, NIGER

Abstract

In this article, a spell corrector has been designed for the Hausa language which is the second most spoken language in Africa and do not yet have processing tools. This study is a contribution to the automatic processing of the Hausa language. We used existing techniques for other languages and adapted them to the special case of the Hausa language. The corrector designed operates essentially on Mijinguini's dictionary and characteristics of the Hausa alphabet. After a brief review on spell checking and spell correcting techniques and the state of art in the Hausa language processing, we opted for the data structures trie and hash table to represent the dictionary. The edit distance and the specificities of the Hausa alphabet have been used to detect and correct spelling errors. The implementation of the spell corrector has been made on a special editor developed for that purpose (LyTexEditor) but also as an extension (add-on) for OpenOffice.org. A comparison was made on the performance of the two data structures used.

Keywords: Natural Language Processing, Spell Checker, Spell Corrector, Computerization of Hausa, African Languages.

1. INTRODUCTION

Automatic natural language processing (NLP) has many industrial applications including, among others, verification and correction of spelling and grammar, text indexing and retrieval of information from the Internet, voice recognition and synthesis, vocal control of domestic robots, automated response systems and machine translation [1 - 2]. The most commonly used application is of course spell checking. Indeed, it is integrated into computer tools used every day by millions of people worldwide. Computer programs in this area are of two kinds: spell checkers and spell correctors. A spell checker detects spelling errors in a given text whereas a spell corrector both detects spelling errors and seeks for the most likely correct words [3]. The correction can be automatic (in the case of a speech synthesizer for example) or interactive allowing the user to select the desired word from several suggestions [1]. This second approach is the one used by most of word processing softwares. Such programs are generally designed for a given language. Spell checking is nowadays present in almost all computer applications where text is expected to be entered by the user. Wrong words are generally underlined in red to alert the user. Examples of such applications are word processors, email clients, source code editors, programming environments and search engines. The causes of errors are of several orders and there is more than one way to classify them [4]. The most important causes are ignorance of the author, typographical errors and errors in transmission and storage [3]. A spell corrector performs two main functions, one after another: first detecting and then correcting spelling errors. In one of his articles, Kukich [1] said that the methods of detection and correction use three approaches:

- non-word error detection, eg. 'grafe' written instead of 'giraffe';

- isolated-word error correction;
- Context error detection and correction: each word is considered taking into account the context, thereby correcting spelling errors even when they consist of real words.

2. TECHNIQUES AND ALGORITHMS FOR ERROR DETECTION

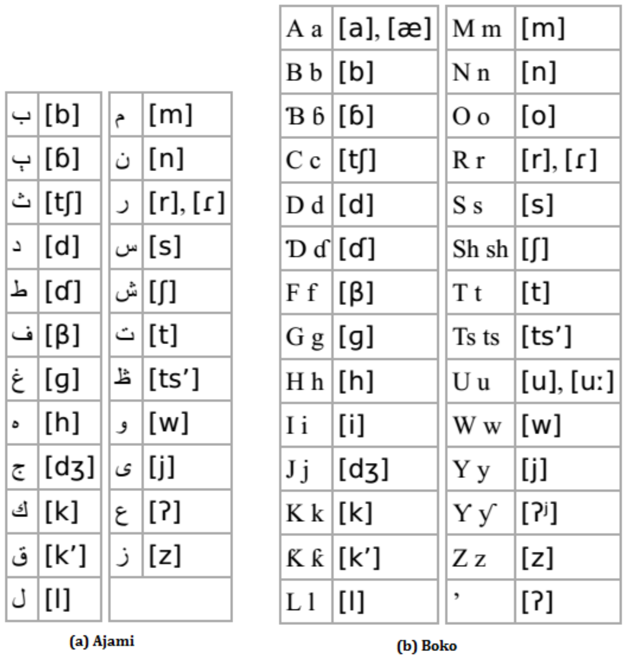
The search for solutions to spelling errors correction problem has been a challenge for many years. Efforts in that area led to the emergence of various techniques and algorithms. Error detection is to find incorrect words in a text. A wrong word is then marked by the application in charge of spell checking. If the word is really wrong - because it is not always the case - an error is said to be detected. Research in this area has been done by many authors [2 - 10]. The main techniques used for non-word error detection in a text are either based on analysis of n-grams, or dictionary lookup [1]. The techniques based on n-grams are to analyze each n-gram of a given input word and check its validity in a precompiled table. These techniques usually require a dictionary or corpus that’s large enough to determine the statistics table of n-grams [1]. A dictionary is a collection of correct or acceptable words. Some authors use the term lexicon or the phrase "word list" instead of dictionary. The techniques based on the use of a dictionary or lexicon involve taking a word as input and verifying its existence in the dictionary. Any word that is not in the dictionary is then considered wrong [1]. A detection algorithm based on dictionary lookup is given by Peterson [3]. The hash table is one of the most commonly used data structures to reduce response time when searching in a dictionary [1]. The idea of hash table was introduced for the first time in 1953 [11]. With the hash code, a hash allows a selective access to the searched word and, therefore, significantly reduces the response time. But the major drawback is finding a hash function which admits very few collisions and provides uniformly distributed indices in the considered interval. UNIX spellchecker *Spell* illustrates the use of a hash table for fast search in a dictionary. Binary search trees are especially useful to check whether a given word belongs to a larger set of words. Several variations of binary search trees were used to accelerate dictionary search. Among them is the Median Split Tree, a modified frequency-ordered binary search tree that allows faster access to most frequently used words. Finite automata were also used in some search algorithms in a dictionary or a text. One of the famous algorithms in this area is that of Aho - Corasick [12]. The algorithm is to move through an abstract data structure called dictionary that contains the words to search by reading the text characters one by one. The data structure is implemented efficiently, which ensures that each character of the text is read only once. Generally, the dictionary is represented using a trie. A trie may be seen as a representation of the transition function of a deterministic finite automaton. The algorithm has a linear complexity in the size of the text and search strings. Comparatively, techniques using n-grams derived from a dictionary provides less accuracy than those using all the information in the dictionary. But, the latter ones are time consuming depending on the data structure used to represent the dictionary. A comparative study showed that the hash table provides better performance than the AVL tree, the Red-Black tree and Skip list [9]. A comparison of five data structures was performed for the Punjabi dictionary [13]. It concerned binary search tree, trie, ternary search tree, multi-way tree and reduced memory method tree. As a result, the binary search tree was found to be the most suitable data structure in terms of memory usage and time. But it is limited when it comes to suggest a list of candidates for the correction or find all words that differ by one or two characters. This limitation may be avoided by the use of a trie which offers almost the same time complexity with a binary search tree. Hash table and trie are shown to be the most suitable data structures for dictionary representation.

3. TECHNIQUES AND ALGORITHMS FOR ERROR CORRECTION

Error correction refers to the fact of equipping spell checkers with the ability to correct detected errors. This is to find words in the dictionary (or lexicon) that are similar in some ways to the misspelled word. The task of a spell corrector is thus composed of three sub-tasks: detecting errors, generating possible corrections, and ranking suggested corrections. To achieve this, various techniques were invented. Each is related either to non-word error correction, real-word error correction, or both. Spelling errors may be typographical, cognitive or phonetic. Typographical errors occur when the keys are pressed in the wrong order (eg ahnd instead of hand). Cognitive errors arise from ignorance of the correct spelling of the word (eg sicretary instead of secretary). Phonetic errors are special cases of cognitive errors. A phonetic error refers to a wrong word that is pronounced the same way as the correct word (eg speack / speak). In typed texts, 1% to 3% of the errors are spelling errors [14]. Damerau [6] stated that 80% of these errors are related to insertion, deletion, substitution, or transposition. The minimum edit distance or simply edit distance is until now the most widely used technique in the spelling errors correction. It has been applied in almost all spell checking functions in text editors and command language interfaces. The first spelling correction algorithm based on this technique was proposed by Damerau [6]. Almost at the same time, Levenshtein also developed a similar algorithm. Several other algorithms on edit distance were born thereafter. The edit distance is defined as the minimum number of edit operations required to transform a word to another [1]. These operations are insertion, deletion, substitution and transposition. In most cases, correcting a spelling error requires the insertion, deletion or substitution of a single character, or the transposition of two characters. When a wrong word can be transformed into a dictionary word by inverting one of these operations, the dictionary word is considered a plausible correction. Damerau's algorithm [6] for edit distance detects spelling errors by comparing words of four to six characters with a list of most frequently used words. When there are multiple candidate words for a given edit distance on a detected word, the first word in the dictionary appearing in alphabetical order is chosen. Levenshtein's algorithm is in the field of dynamic programming and seems to be the most widely used in edit distance computing. Each edit operation is assigned a cost, usually 1 for deletion and insertion and 2 for substitution and transposition. Given a dictionary of n words, the correction algorithms based on edit distance generally require n comparisons for each wrong word. To reduce the search time, reversed edit distance technique is used. Another approach used to reduce the number of comparisons involves sorting or partitioning the dictionary according to certain criteria (alphabetical order, word length, words occurrences). Many other techniques are also used in spelling errors correction like: similarity keys, rules system, n-grams, probabilistic techniques and neural networks. However, the most widely used technique in errors correction remains edit distance [7]. It has a time complexity of O(nm), with n and m the respective sizes of the two compared words. A technique developed by Horst [15] combining automata and edit distance was used to quickly find the closest correct word to a wrong word. It has a linear complexity in time relative to the length of the wrong word, regardless of the dictionary size. But the space complexity of the method is exponential ($O\left(3.\exp\left(\sum_{i=1}^{N}|A_i|\right)\right)$, where A_i are the words of the dictionary).

4. REVIEW OF THE HAUSA LANGUAGE PROCESSING

Hausa is part of the family of Afro-Asiatic languages. It belongs to the group of Chadic languages (sub-group of West Chadic languages) [16]. Compared to other African languages, Hausa is remarkably unitary. Standard Hausa (Kano dialect) is to be distinguished from West dialect (Sokoto) and Nigerien dialects (Tibiri, Dogondoutchi, Filingué) [17]. From a vocal point of view, the Hausa words have high tones and low tones and one can observe a flexion of gender and number [18]. Geographically, Hausa is the second most spoken language in Africa. It is the most widely spoken language in sub-Saharan Africa with about a hundred million speakers worldwide. Hausa is now used by major radio stations of the world such as VOA (USA), BBC (UK), CRI (China), RFI (France), IRIB (Iran), Deutsche Welle (Germany) and Radio Moscow (Russia). In Niger, Hausa and other national languages are used by regional and local, public and private media [19]. Cinematographically, the Hausa language video industry has made a remarkable progress. Indeed, over 1000 Hausa films are produced each year, mainly from Nigeria. The presence of Hausa on Internet is very precarious. It is unfortunately the same case with all African languages even though they represent 30% of the languages of the world [20]. According to Van Der and Gilles-Maurice [20], Hausa texts can be divided into three main categories: popular culture (47%), newspapers (35%) and religion (17%). The percentage of texts produced on forums by Internet users is tiny (0.3% of total words). Today's famous search engines like Google and Mozilla Firefox as well as other softwares and electronic gadgets (including mobile phones) have a graphical user interface in Hausa. The ISO identifier for Hausa language is *ha* or *hau* (ISO 639-3 and ISO 639-1). On the academic side, the first poems written in Hausa with Arabic alphabet adapted to the notation of African languages (`Ajami), date from the early nineteenth century. To this tradition was added in the 1930s, as a result of British colonization, a literary production in Latin alphabet (dramas, tales, stories, poetry) [21]. Hausa language is now being taught in African and Western universities (Niger, Nigeria, Libya, Inalco (Paris) , Boston University, UCLA) . The written Hausa is essentially based on the dialect of Kano and there are two writing systems, one based on the Arabic script (Ajami) and the other using the Latin alphabet (Boko) as shown in Figure 1. We note, in the case of Boko, the presence of four additional special characters like consonants (ɓ , ɗ , ƙ and ƴ) and glottal stop (').



ب	[b]	م	[m]
ٻ	[ɓ]	ن	[n]
ث	[tʃ]	ر	[r], [ɾ]
د	[d]	س	[s]
ط	[ɗ]	ش	[ʃ]
ف	[β]	ت	[t]
غ	[g]	ظ	[ts’]
ه	[h]	و	[w]
ج	[dʒ]	ى	[j]
ك	[k]	ع	[ʔ]
ق	[k’]	ز	[z]
ل	[l]		

(a) Ajami

A a	[a], [æ]	M m	[m]
B b	[b]	N n	[n]
Ɓ ɓ	[ɓ]	O o	[o]
C c	[tʃ]	R r	[r], [ɾ]
D d	[d]	S s	[s]
Ɗ ɗ	[ɗ]	Sh sh	[ʃ]
F f	[β]	T t	[t]
G g	[g]	Ts ts	[ts’]
H h	[h]	U u	[u], [uː]
I i	[i]	W w	[w]
J j	[dʒ]	Y y	[j]
K k	[k]	Ƴ ƴ	[ʔʲ]
Ƙ ƙ	[k’]	Z z	[z]
L l	[l]	’	[ʔ]

(b) Boko

FIGURE 1: Writing Systems for Hausa Language.

The Latin transcription, introduced by the British in Nigeria in the early 20th century, has emerged in 1930 as the official spelling [17]. In Niger, it was only in 1981 that the Latin alphabet was used as an official Hausa spelling. This alphabet was completed in 1999 by a decree [22]. This is the same alphabet as that of Figure 1 (b) to which are added digraphs fy, gw, kw, ky, ƙw, and ƙy representing specific and considered consonant sounds. The same decree defined the symbols in Table 1 as punctuation symbols.

Symbol name	Symbol
Full stop	.
Comma	,
Semi colon	;
Colon	:
Interrogation mark	?
Exclamation mark	!
Parentheses	()
Quotes	"
Union mark	-
Suspension marks	…
Dash, next line	-
Asterix	*
Dashes or parentheses	…-…-
Dash	-

TABLE 1: Hausa Official Punctuation Symbols In Niger.

The Boko became the dominant writing convention for scientific and educational materials, mass media, information and general communication since the second half of the 20th century [23]. The first step in the computerization of a language is the existence of language resources [24]. It is the only possible way to design computer tools (editors, spelling and grammar checkers, electronic dictionaries ...) adapted to the language and ensure its presence in the cyberspace. But these resources are scarce for African languages. Various projects and studies are carried out which aim is the constitution or usage of these linguistic resources for total computerization of African languages. For example, PAL is a project aimed at adapting information technology to African languages to make them more accessible to indigenous peoples [25]. The work on modern Hausa lexicography started in 1843 with Schön's "Hausa vocabulary". Schön compiled, in 1876, the first work that can be called Hausa-English bilingual dictionary with 3800 entries. A few years later appeared the first Hausa - French dictionary written by Le Roux in 1886. In the early 20th century, three other bilingual dictionaries were published, including Landeroin and Tilho's (1909) Hausa-French dictionary with 6000 entries. The dictionary of Bargery [26] seems to be the most important and the largest (with 39 000 words) Hausa dictionary. In their Hausa lexicography genesis, Roxana and Paul [27] mentioned several other dictionaries before discussing the Hausa - French bilingual dictionary written by Mijinguini [28], a Nigerien Hausa native linguist. This dictionary is, according to them, "the latest scientific reference in Hausa lexicography". It includes 10,000 well illustrated entries and is largely based on the standard Hausa of Niger, consisting essentially of the Damagaram dialect instead of the Kano dialect that dominated all previous lexicographical researches. It is important to recall that Paul [29] is the author of the most comprehensive work on modern Hausa grammar. The majority of well-resourced languages have well-formed corpuses. This is not the case for African languages. The current researches on these languages choose oral and written corpuses as a transitional alternative. Another alternative for African languages is to build a corpus from the Web [30]. For example, a search for four days on the Web ended up with a Hausa language corpus containing 858,734 words in total, including 30,996 different words [20].

Text entry is another difficulty to overcome in the computerization of Hausa and other African languages. Indeed, computer keyboards are not compatible with these languages. Thus entering some Hausa characters on a keyboard now requires an acrobatic work. The solution for this problem is the use of keyboard layouts to write African languages specific characters. An evaluation of such keyboards for 5 Nigerien languages (Fulfulde, Hausa, Kanuri, Songhai - Zarma, Tamasheq) recommended the LLACAN keyboard [31]. In fact, LLACAN covers all the symbols of the alphabets of those languages, produces valid Unicode code and requires less buttons to press.

Word processing softwares such as MS Word and OpenOffice.org Writer can be used for the correction of written Hausa text through the establishment of a user dictionary. However, all existing methods are limited and inadequate in the case of African languages, hence the need to develop spell correctors adapted to these languages [32]. Despite the scarcity of linguistic resources, it is possible to develop such spell correctors and improve them over the time. With the possibility to create extensions for some popular softwares (MS Word, OpenOffice.org Writer, Firefox, etc.), it would be advantageous to develop spell correctors that can be easily integrated to them.

5. DESIGN OF A SPELL CORRECTOR FOR HAUSA LANGUAGE

After synthesizing spelling correction techniques and presenting the Hausa language, we can now design a spell corrector for that language. We expose the approaches and techniques chosen and the implementation details of the proposed solution.

5.1. Chosen Techniques

In this section, we present the data structures used for the design of the corrector and the procedures necessary for the detection and correction of errors in Hausa. We decided to approach the subject with an algorithmic design inspired from Java [33]. We will not dwell on the theory of the underlying concepts such as class, object, method, attribute, instance, etc. [33] and [34] are good references on this subject. Taking into account the linguistic resources available to us, a technique based on a dictionary seems to be more suitable for the design of the corrector. Regarding the dictionary, we opted for that of Mijinguini [28] for its assets and accessibility to us. The dictionary contains all the words (including inflections and derivations). It is stored in secondary memory as a text file using UTF-8 character encoding. Error detection is independent of the context. An erroneous word is identified by a simple dictionary lookup. To represent the dictionary in primary memory, we use either a hash table or a trie. The implementation must allow at least the following operations:

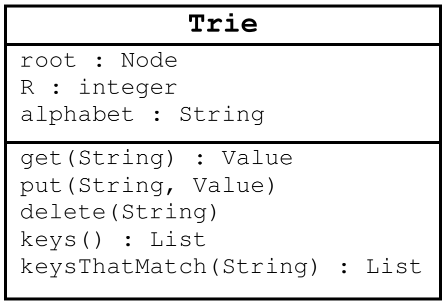
• Add a word to the dictionary (add method)

• Check if a word is in the dictionary (contains method)

• Delete a word from the dictionary (remove method)

Each node in the trie has as many links as there are characters in the alphabet and the latter ones are stored implicitly in the data structure. Each valid character string is assigned a value. This may be of any type. It can be used here to store information on every word in the dictionary (definition, grammatical class, translation into another language, etc.).

In the object notation, a trie is as shown in Figure 2. Each node of the trie is represented by the Node data structure of Figure 3.



```
Trie
root : Node
R : integer
alphabet : String
get(String) : Value
put(String, Value)
delete(String)
keys() : List
keysThatMatch(String) : List
```

FIGURE 2: Representation of the Trie Data Structure.

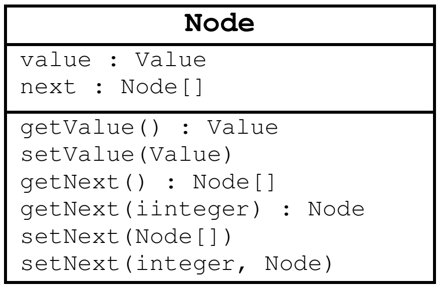


FIGURE 3: Representation of The Node Data Structure.

The R attribute of the class Trie is the number of symbols or letters of the alphabet. Since the digraphs of the Niger Hausa alphabet are not coded as single characters, we shall consider only the monographs which make a total of 28 letters. To these letters, we add the dash ('-') (Unicode code \ u002D) in order to store compound words. If the language were supported by ASCII code, it would not be necessary to have an alphabet attribute for Trie class, as the characters are represented by consecutive numbers from 0 to 127 therefore by indices of *next* array (Node [ ]) . This is not the case for the Hausa language where letters have code points in the following ranges:

• Uppercase letters: 39, 65-80, 82-85, 87-90, 385, 394, 408, 435.

• Lowercase letters: 97-112, 114-117, 119-122, 595, 599, 409, 436.

Representing characters by indices of *next* array will set the value of R to 599 instead of 56 (28x2). This will inevitably lead to a waste of memory space and additional checks to prevent foreign words from being added to the trie. To avoid this problem, a trick [35] is to find a mapping function between indices of the next array and letters of the alphabet. That is why the *alphabet* attribute is present in the class *Trie*. It is here of type String but it may also be an array of characters. Two additional methods, toChar and toIndex, assure the conversion from indices to characters and vice versa. The charAt and the indexOf methods of the String class can be effectively used. And to make the trick more flexible, we can totally delegate this task to an interface *Alphabet* that defines *toChar* and *toIndex*. The *KeysThatMatch* is another interesting method. Indeed, it allows to search the trie for words that match a given pattern. The patterns used here are those with a wildcard, for example a dot ('.'). For instance, given the pattern '.ada', this method will return the dictionary words which consist of a letter (any) followed by the suffix 'ada' : dada, fada, kada, lada, tada, wada. It is this possibility that we use to implement the reverse editing distance. The *KeysThatMatch* method uses a data structure *List* (a linked list of Strings) to keep the search results. The *List* class has methods to add an item, to verify the existence of an item and to delete an item.

To abstract the implantation of the real dictionary, add flexibility, simplify maintenance and facilitate scalability of the spell corrector, an abstract dictionary is represented by a class (*TrieBasedDico* or *HashBasedDico*) that implements *Dico* interface (or abstract class). It defines the methods (add, remove, contains) needed to operate on a dictionary. *TrieBasedDico* and *HashBasedDico* classes are designed by composition from Trie and HashSet (hash) classes respectively as shown in Figure 4.

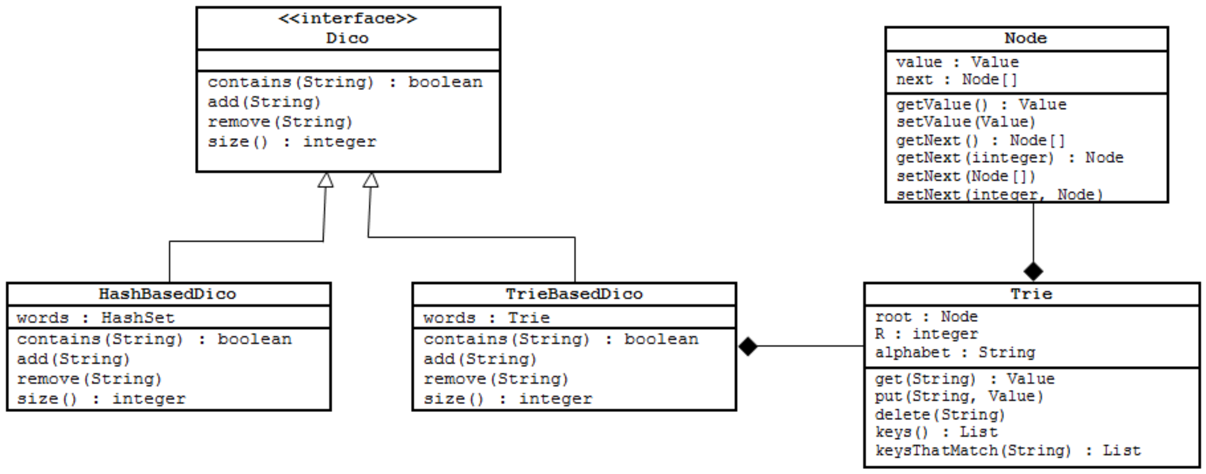


FIGURE 4: Class Diagram For The Implantation of The Dictionary.

The list of candidate words for the correction of an erroneous word is determined in several steps that we describe here. Once a word is identified as being erroneous, the procedure for determining the type of the error follows. We defined three types of errors (inspired by our research on OpenOffice.org):

- IS_NEGATIVE_WORD: Error caused by the presence of a number or a character not belonging to the alphabet (eg x , v, q, etc.) in the word. The word is called negative.
- CAPTION_ERROR: Case Error. This is when a word that should be written with the first letter capitalized is written entirely in lowercase.
- SPELLING_ERROR: represents all other types of spelling errors.

The types of errors are short integers encapsulated as static fields in the *LySpellFailure* class. The corrector has two methods for the determination of errors. First, the *getSpellFailure* method which analyzes a given word and returns -1 if the word is correct or one of the three types of errors mentioned above otherwise. Then *isValid* method that checks whether a given word is valid according to the result returned by *getSpellFailure* and spellchecking settings. If *getSpellFailure* returns a value:

- equal to -1, the word is valid and isValid returns true
- other than -1, the correction parameters are taken into account to determine the validity of the word. For example when you choose not to correct words with numbers and the erroneous word contains digits, *isValid* returns true. This method can be exploited to correct spelling as you type.

currentLanguage represents the language being supported by the spell corrector. It is an instance of *Language* class. Searching suggestions is performed by *propose* which is an instance of a class that implements the interface *Proposer*.

The method *getProposals* provides correction suggestions for an invalidated word by *isValid* depending on the type of error detected by *getSpellFailure*.

a) Use of the characteristics of the alphabet

The processed language is represented by the Language class given in figure 5:

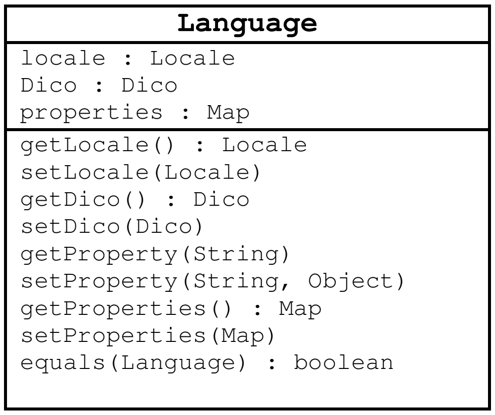


FIGURE 5: Language Class Diagram.

After several attempts, we decided that the dictionary is an attribute of the language and not the reverse. The Local attribute of the Language class stores information about the processed language. It is of type Locale (representation of a language in Java) and provides among others : a 2-letter ISO 639-1 code of the language, a 2-letter ISO 3166 code of the country as well as the complete names of the language and the country. This corresponds to ha, NE, Hausa (Niger) respectively for Hausa of Niger. We use this data for naming resources and for user display. The *properties* attribute is of type Map (mapping key / value) and stores other properties of the language that we use to design the spell corrector and which are not provided by *Locale*. They are currently the alphabet of the language (value of the key "alphabet"), the special characters in the alphabet (value of the key "specialChars"), the characters that look like special characters (value of the key "specialCharsLike") and the punctuation symbols that we divided into two parts: word separators (value of the key "punctuation") and end of sentence signs (value of key "endOfSentence"). All the characters of the alphabet are coded in Unicode. The class in charge of finding suggestions implements *Proposer* interface which defines two methods: *isNegativeWord* and *propose* as shown in the class diagram of Figure 6. The *TrieBasedDicoProposer* and *HashBasedDicoProposer* classes use some features of the alphabet to find candidate words.

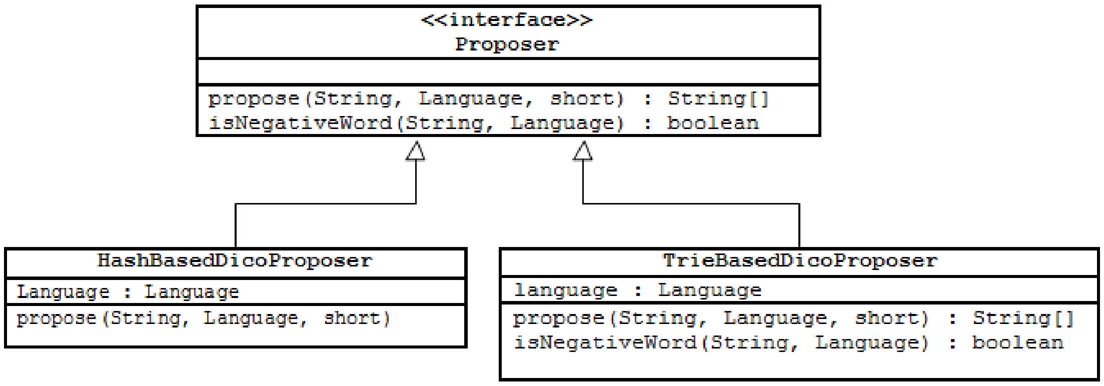

<<interface>>
Proposer
propose(String, Language, short) : String[]
isNegativeWord(String, Language) : boolean
HashBasedDicoProposer
Language : Language
propose(String, Language, short)
TrieBasedDicoProposer
language : Language
propose(String, Language, short) : String[]
isNegativeWord(String, Language) : boolean

FIGURE 6: Class Diagram For Correction Suggestions.

b) Use of the reverse edit distance to find candidate words

Candidate words are found using the reverse edit distance as follows:

- All words having an edit distance equal to 1 with the wrong word are generated by applying edit operations such as insertion, deletion, substitution and transposition. A total of 60n+28 words are generated for a wrong word of length n.
- Each previously generated word is searched in the trie or the hash table (which represents the dictionary). If it is there, then it is retained as a possible correction of the erroneous word.

The research is conducted by a private method called *proposeByReverseEditDistance*. This method is actually based on *keysThatMatch*. It takes an argument of type *TrieBasedDico* and a word or a pattern and returns the result as an array of Strings. A similar method is designed in the case of hash table. Methods that perform editing operations on a given word are provided by the *StringTools* class which consists of tools shared by different classes.

c) Use of edit distance to rank candidate words

The minimum edit distance is used to rank the suggested words. Those who are closest to the wrong word are placed at the top of the list. To implement that, a comparator was designed.

The entire class diagram of the designed spellchecker is given in Figure 7 below:

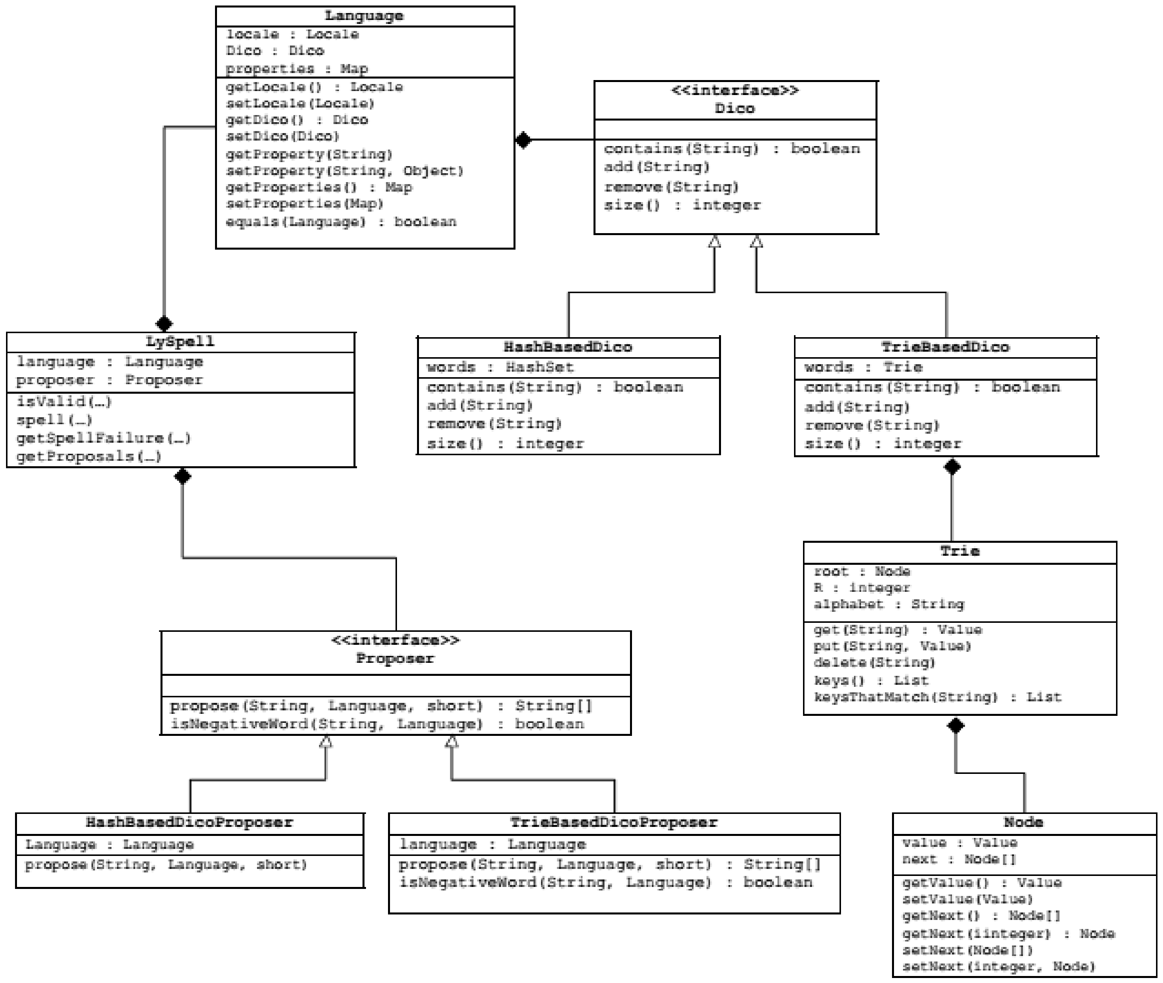


FIGURE 7: Global Class Diagram.

5.2. Implementation of The Spell Corrector

To implement the solution, we opted for Java and NetBeans IDE. Two versions have been developed.

a) Standalone LyTexEditor and LySpell

This solution includes a text editor "LyTextEditor" that integrates a spell corrector "LySpell". LyTextEditor gives the following possibilities:

• type a text;
• open an existing text file;
• correct a text with LySpell;
• save a text.
Spell checking and correction is accessed via the Tools menu or by pressing F7.

b) Add-on for OpenOffice.org

With the help of the OpenOffice.org Developer's Guide [36], we were able to develop the add-on.

With the portability of Java, LyTextEditor and LySpell can normally be used on all platforms. LySpell also offers the opportunity to correct spelling errors for other languages without any need to modify the code. One can do so by simply providing a dictionary file and the alphabet of the intended language.

6. RESULTS

In this work, we developed a spell corrector for Hausa language. The final product was tested as a standalone program through a text editor designed for this purpose and as an extension for the OpenOffice.org office suite. These results show that it is possible, from proven techniques and language resources, to develop automatic processing tools for African languages in general and for Hausa in particular. They also confirm that the data structures trie and hash table offer better performance for storing a dictionary and compare very well with [9] and [13]. A trie is actually a DAWG (Directed Acyclic Word Graph), a way to represent an acyclic deterministic finite automaton. However, the possibilities and the results provided by the data structure trie are significantly better than those of the hash table. Note that when the number of wildcards is greater than 1, only the trie gives easily a satisfactory result. For example, for the incorrect word "zurmakakke", the correct word "zurmaƙaƙƙe" is suggested when the dictionary is implanted using a trie while no suggestion is obtained in the case of the hash table.

The corrector LySpell resulting from this study uses only a dictionary as language resource and the alphabet of Hausa language. However, it was designed and implemented so that it can also be used for other languages. The specificities of the Hausa and other African languages are efficiently handled.

Although we were not able to perform all necessary tests on the performance of LySpell, we believe that the results we have obtained will add value to the computerization of the Hausa language and contribute to its effective use in institutions of education and on media.

To improve the performance of the designed corrector, it may be considered in future works the possibility to:

- use the morphologic rules of the Hausa language. This will have a triple advantage. First the size of the dictionary in memory will be significantly reduced. Then, the suggestions for correction could be more precise. Finally, it is possible to create a Hunspell oriented spell corrector that can be integrated easily and appropriately to a wide range of programs.

- Strengthen the spell checking and correction by adding grammar checking.

7. REFERENCES

[1] K. Kukich. "Techniques for automatically correcting words in text", ACM Computing Surveys, vol. 24, No. 4, 1992.

[2] M.N. Pierre. "An introduction to language processing with Perl and Prolog", Springer-Verlag Berlin Heidelberg, p.2-3, 2006.

[3] J.L. Peterson . "Computer Programs for Detecting and Correcting Spelling Errors", Comm. ACM, vol. 23, No. 12, December 1980.

[4] V. Suzan. "Context-sensitive spell checking based on word trigram probabilities", Master thesis, February - August 2002.

[5] N.A. Cyril. "String similarity and misspellings", Communications of the A.C.M., vol. 10, no. 5, pp. 302-313, May 1967.

[6] F.J. Damerau. "A technique for computer detection and correction of spelling errors", Comm. ACM 7, vol. 3 , pp. 171-176, March 1964.

[7] L.L. Hsuan. "Spell Checkers and Correctors: a unified treatment", Master dissertation, November 2008

[8] B. Laurent. "Production de logiciels et d'utilitaires pour le traitement informatique de langues africaines dans un contexte de NTIC multilingues", 2nd World Congress of Community Networks, Buenos Aires, Argentine, du 5 au 7 décembre 2001.

[9] P.N. Mark. "A Comparison of Dictionary Implementations", April 10, 2009.

[10] E.M. Zamora. Pollock J. J. and Antonio Z., "The use of trigram analysis for spelling error detection", Information Processing & Management, vol. 17. No. 6, pp. 305-316, 1981.

[11] K. Donald. "The Art of Computer Programming", Addison-Wesley Publishing Co., Philippines, vol. 3, 1973.

[12] A.V. Aho and M.J. Corasick. "Efficient String Matching: An Aid to Bibliographic Search", Communications of the ACM, vol. 18, No. 6, pp. 333-340, June 1975.

[13] G.S. Lehal and K. Singh. "A Comparative Study of Data Structures for Punjabi Dictionary", 5th International Conference on Cognitive Systems, reviews & previews, ICCS’99, pp. 489-497, 2000.

[14] J. Daniel and H.M. James, "Speech and Language Processing", Prentice Hall, Englewood Cliffs, Inc., 2000.

[15] B. Horst. "A Fast Algorithm for Finding the Nearest Neighbor of a Word in a Dictionary", IAM-93-025, November 1993.

[16] The Online Encyclopedia of Writing Systems & Language, 16/12/2013 09:31, http://www.omniglot.com/writing/definition.htm.

[17] http://www.humnet.ucla.edu/humnet/aflang/Hausa/hausa.html.

[18] A. Mijiguin and H. Naroua. "Règles de formation des noms en haoussa", Actes de la conférence conjointe JEP-TALN-RECITAL 2012, Atelier TALAf 2012: Traitement Automatique des Langues Africaines, pp. 63-74, 2012.

[19] M.G. Maman and H.H. Seydou. "Les Langues de scolarisation dans l'enseignement fondamental en Afrique subsaharienne francophone : cas du Niger", Rapport d'étude pays, 2010.

[20] A.V. Van Der and D.S. Gilles-Maurice. " The African Languages on the Internet: Case Studies for Hausa, Somali, Lingala and isiXhosa", Cahiers Du Rifal, vol. 23, pp. 33–45, 2003.

[21] C. Bernard. "Les langues au Nigeria", Notre Librairie, Revue des littératures du Sud, Littératures du Nigéria et du Ghana, vol. 2, no. 141, pp. 8-15, 2000

[22] Arrêté N°0212 MEN/SP-CNRE du 19 oct. 1999 modifiant et complétant l'arrêté n°01/MEN/SCNRE/MJSC/MESR/M.INF/MDR/MI du 15 mars 1981 relatif à l'orthographe de la langue hausa.

[23] N. Ahmed. "Adaptation des écritures et de la lecture des langues étrangères au pays Haoussa de l'Afrique de l'Ouest", Synergies Algérie n°6 – 2009, pp. 61-69, 2009.

[24] C. Chanard and A. Popescu-Belis. "Encodage informatique multilingue : application au contexte du Niger", Les Cahiers du Rifal, No. 22, pp. 33-45, 2001.[33]Christophe D., "Apprendre à programmer, algorithmes et conception objet", 2e ed., Eyrolles, 2008.

[25] O. Don. "Les langues africaines a l'ère du numerique, défis et opportunités de l'informatisation des langues autochtones", Les Presses de l'Université Laval, CRDI 2011.

[26] G.P. Bargery. "A Hausa-English Dictionary and English-Hausa Vocabulary", Oxford University Press, London, 1934.

[27] M.N. Roxana and N. Paul. "The Hausa Lexicographic Tradition", Lexikos11, AFRILEX-reeks, series 11, pp. 263-286, 2001

[28] A. Minjinguini. "Dictionnaire élémentaire hausa-français", les éditions GG, 2003.

[29] N. Paul. "The Hausa Language An Encyclopedic Reference Grammar", Yale University Press, New Haven, 2000.

[30] D.S. Gilles-Maurice. "Web for/as Corpus: A Perspective for the African Languages", Nordic Journal of African Studies, vol. 11, No. 2, pp. 266-282, 2002.

[31] C. Enguehard and H. Naroua. "Evaluation of Virtual Keyboards for West-African Languages", Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08), Marrakech, Morocco, 28-30 May 2008 .

[32] C. Enguehard and C. Mbodj. "Des correcteurs orthographiques pour les langues africaines", Bulletin de Linguistique Appliquée et Générale, 2004.

[33] D. Christophe. "Apprendre à programmer, algorithmes et conception objet", 2e ed., Eyrolles, 2008.

[34] M. Brett, P. Gary and W. David. "Head First Object-Oriented Analysis and Design", O'Reilly, Nov. 2006.

[35] S. Robert and W. Kevin. "Algorithms", 4e ed., Addison Wisley, 2011

[36] OpenOffice.org 3.1 Developer's Guide, https://wiki.openoffice.org.

KSUCCA: A Key To Exploring Arabic Historical Linguistics

Maha Alrabia *msrabiah@gmail.com*
Department of Computer Science
King Saud University
Riyadh, Saudi Arabia

AbdulMalik Al-Salman *salman@ksu.edu.sa*
Department of Computer Science
King Saud University
Riyadh, Saudi Arabia

Eric Atwell *e.s.atwell@leeds.ac.uk*
Faculty of Engineering
Leeds University
Leeds, United Kingdom

Nawal Alhelewh *drnawalh@gmail.com*
Department of Arabic
Princess Nora bint Abdul Rahman University
Riyadh, Saudi Arabia

Abstract

Classical Arabic forms the basis of Arabic linguistic theory and it is well understood by the educated Arabic reader. It is different in many ways from Modern Standard Arabic which is more simplified in its lexical, syntactic, morphological, phraseological and semantic structure. King Saud University Corpus of Classical Arabic is a pioneering corpus of around 50 million words of Classical Arabic. It is initially constructed for the purpose of studying distributional lexical semantics of the Quran and Classical Arabic, however, it is designed in a general way making it also appropriate for other researches in Linguistics and Computational Linguistics. In this paper, we will briefly describe the structure of our corpus, and then we will demonstrate how it can be used to depict some aspect of Arabic language change between the classical and the modern periods.

Keywords: Historical Linguistics, Corpus Linguistics, Classical Arabic, Modern Standard Arabic, Lexical change, Syntactic Change, Morphological Change, Phraseological Change, Semantic Change.

1. INTRODUCTION

Historical linguistics, also called diachronic linguistics, is the study of why and how languages change or maintain their structure during time [1]. Nowadays, much research in historical linguistics is based on corpora containing texts from earlier periods of the target language allowing linguists to conduct more systematic studies on the evolution of languages and how various linguistic aspects are effected during the course of time [2]. This type of corpora that contain texts from past periods are usually referred to as historical corpora; there exist many historical corpora for English and many other languages. For example, the Helsinki Corpus of English Texts: Diachronic and Dialectal is a well-known historical corpus of English, which consists of two parts; a diachronic part containing 1.5 million words texts from the period between 750AD and 1700AD, and a dialect part consisting of transcripts of interviews with speakers of British rural dialects from the 1970's [3]. Another example is the Corpus of Historical American

English (COHA), which contains 400 million words of American English texts covering the period from 1810 until 2009 [4].

On the other hand, most of research in Arabic historical linguistics are not corpus based [5], which is a consequence of the lack of available appropriate corpora that can be used in such studies. Therefore, in order to study changes in Arabic language, very large corpora of Classical Arabic, which forms the basis of Arabic linguistic theory, should be made available to linguists so that they can compare them to existing Modern Standard Arabic (MSA) corpora in order to observe how and why did Arabic language change.

There exist a decent amount of MSA corpora with different types. For example, the Tim Buckwalter corpus for Modern Standard Arabic is the first corpus developed for Arabic. It was constructed in 1986 from an Arabic newspaper. The corpus was initially around 40000 words, and then it was expanded to more that 2.5 million words when the electronic Arabic content was available in the web. This corpus was designed for lexicographical purposes, and is not freely available for the public[1] [6]. On the other hand, Al-Sulaiti and Atwell [7] constructed the Corpus of Contemporary Arabic (*CCA*) of around one million words for the purpose of teaching Arabic as Foreign Language (*TAFL*). The corpus contains Arabic text from various categories and it is freely available online[2].

Moreover, Alansary et al., [8] compiled the International Corpus of Arabic (*ICA*) of about 100 million words of written MSA collected from a wide range of Arabic regions to insure diversity of writing styles, which makes it a good candidate for linguistic researchers who are interested in studying the influence of nationality on the speakers of MSA. They relied on machine readable sources to compile the corpus; containing newspaper articles, magazines, novels, books, web articles and other academic articles. ICA includes the following main genres: strategic sciences, social sciences, sports, religions, literature, humanities, natural sciences, applied sciences, arts and biographies. In addition, Sawalha and Atwell [9] constructed a large broad-coverage lexical resource for Arabic, which is a corpus of 23 machine-readable lexicons organized into roots, words formed from these roots and the meanings of those words. The authors evaluated the coverage of their corpus using three available Arabic corpora [9]; it scored 65-68% when using exact word matches and 82-85% when a lemmatizer was used to remove clitics. Moreover, Alfaifi and Atwell [10] developed the Arabic Learner Corpus (ALC), which is a 31272 words corpus consisting of texts written by learners of Arabic in Saudi Arabia. The corpus covers both native Arabic students who are learning to improve their Arabic language abilities and foreign students who are learning Arabic as a second language. For other examples of MSA corpora, I refer the reader to [6].

On the other hand, and to the best of the authors knowledge, there exist only two corpora of Classical Arabic; one is part of the King Abdulaziz City for Science and Technology Arabic Corpus (KACST Arabic Corpus)[3] and the other is the Classical Arabic Corpus (CAC) [11]. However, neither of the two corpora is adequate for research in distributional semantics; the former has a limited number of genres and it only contains 17+ million words, which is not very sufficient. While the latter is even smaller with only 5 million words. Therefore, it was essential to design and compose a new corpus of Classical Arabic bearing in mind that it should be large enough, balanced, and representative so that any result obtained from it can be generalized for Classical Arabic. In this paper, we will give a brief description of the design and construction of King Saud University Corpus of Classical Arabic (KSUCCA), which is a very large corpus of Classical Arabic that can be used in various corpus linguistic studies. In addition, we will demonstrate how KSUCCA corpus can be used in historical linguistics.

1 http://www.qamus.org
2 http://www.comp.leeds.ac.uk/eric/latifa/research.htm
3 http://www.kacstac.org.sa/Pages/Default.aspx

The paper is structured as follows. Section 2 provides a brief description of the corpus. Section 3 demonstrates and discusses some aspects of change in Arabic language using KSUCCA. Finally, Section 4 discusses the conclusions of the work presented.

2. King Saud University Corpus of Classical Arabic (KSUCCA)

Texts included in KSUCCA are Arabic texts dating back to the period of the pre-Islamic era until the end of the fourth Hijri[4] century (equivalent to the period from the seventh until early eleventh century CE) [12]. The corpus is classified into 6 broad genres (Religion, Linguistics, Literature, Science, Sociology and Biography) covering most of the topics that were popular in that period of time, which is a strong indication of the corpus representativeness. These genres are further classified into 27 subgenres as shown in Table 1. It can be noticed that the number of texts and tokens are not evenly distributed between genres. However, this is consistent with the knowledge of the overall writing trends at that period of Arab history, and it is an indication of the balance of the corpus [13].

Genre	Subgenre	No. of documents	No. of tokens	Percentage
Religion	The Holy Quran	1	78245	0.15 %
	Hadith	44	5784326	11.43 %
	Exegesis of The Quran	13	7061862	13.96 %
	Quranic Studies	29	3665288	7.24 %
	Hadith Studies	10	643144	1.27 %
	Belief	23	486801	0.96 %
	Jurisprudence	26	5567407	11.00 %
	Principles of Jurisprudence	4	358014	0.71 %
Literature	Poetry	42	1265696	2.50 %
	Novels	2	172695	0.34 %
	Literature and Eloquence	60	5786113	11.43 %
Linguistics	Grammar and Morphology	16	1400951	2.77 %
	Language	6	401308	0.79 %
	Lexicons	27	4855732	9.60 %
	Proverbs	7	435975	0.86 %
Science	History	19	3750498	7.41 %
	Geography and Travel	14	609979	1.21 %
	Medicine	3	1837452	3.63 %
	Physics	1	61347	0.12 %
	Astronomy	2	112695	0.22 %
	Philosophy	1	24760	0.05 %
	Politics	1	4674	0.01 %
	Miscellaneous	1	27728	0.05 %
Biography	Prophet Muhammad Peace be	8	1163795	2.30 %
	Other biographies	18	2336153	4.62 %
Sociology	Ethics and Morals	23	1081566	2.14 %
	Genealogy	9	1628208	3.22 %
Total		410	50602412	100 %

TABLE 1: Classification of KSUCCA Texts.

KSUCCA is designed as a general corpus analogous to the Brown [14], LOB [15], BNC [16], Corpus of Contemporary Arabic (CCA) [6] and other general corpora that can be used for a variety of Linguistics and Computational Linguistics research. In the next section, we will demonstrate how KSUCCA can be used to detect various aspects of language change between Classical Arabic and MSA.

4 The *Hijri* calendar is the official calendar for Muslims. Its first year was the year when the *Hijra*, migration, of Prophet Muhammad from *Makkah* to *Madinah* occurred, which is equivalent to 622 CE.

3. KSUCCA AND HISTORICAL LINGUISTICS

A key factor in understanding how language change is to look at the change in frequencies of the linguistic phenomenon under study. In fact, the change of frequency of a given word in time varying corpora can be an indication of historical, cultural or social changes [4]. Many Arabic words that were popular in the Classical Arabic period are used rarely in MSA. On the other hand, many new words have evolved in MSA Arabic due to cultural and social changes. In addition, noticeable drifts in the meanings of some words between the classical and the modern periods of Arabic have occurred. In this section, we will demonstrate how KSUCCA can be used to depict some lexical, Phraseological, vocabulary and semantic changes between Classical Arabic and MSA.

3.1 Lexical Change

One example of lexical change is the usage of the word (الغيث) (*Alghaith*), which is one of the synonyms of the word *rain*; this word witnessed a major drop in frequency in MSA. Table 2 and Figure 1 show the frequency rate (3/100,000) of this word in classical literature, taken from KSUCCA, compared to its frequency rate (0.58/100,000) in modern literature[5]. This decrease in frequency for the word *Alghaith* in modern Arabic literature was accompanied by an increase in the frequency rate (11.5/100,000) of another synonym of the word rain (المطر) (*Almatar*), as in Figure 1.

	Alghaith	***Almatar***
Classical Literature	3	5.62
Modern Literature	0.58	11.5

TABLE 2: Frequency rates of the words *Alghaith* and *Almatar* in classical and modern literature

These figures are of strong indication of a cultural and linguistic crisis of Arabic. This is due to the fact that the two synynoms *Alghaith* and *Almatar* are not absolute synonyms. In fact, it is belived by many ancient and contemporary Arabic linguists that there are no absolute synonyms in Arabic, and that there exists, definitely, a differnce in meaning between every pair of synonyms. This thoery applies to the two synonyms *Alghaith* and *Almatar*; the word *Alghaith* refers to the rain that falls when people and crops are of great need and thurst, and also to the rain that does not cause any damage to people, cattle, crops, property, etc. On the other hand, the word Almatar can be used to describe the rain that causes damages or the rain that does not [17]. A look at the figures in Table 2 shows a drastic decrease in the usage of the word *Alghaith* in modern literature. The use of that word decreases even further, as expected, in common daily language use as in newspapers articles where it reaches a frequency rate of (0.27/100,000)[6]; it is barely used.

This may indicate that Arabic speakers no longer taste language and their linguistic background does not allow them to use proper synonyms in their proper contexts, which results in a linguistic phenomenon known as semantic generalization. Semantic generalization is a strong sign of language decay, which has a shrinking effect on contemporary lexicons reducing the amount of their vocabulary tremendously.

5 http://arabicorpus.byu.edu.

6 The All Newspapers corpus is a 135,360,804 word sub corpus of arbiCorpus (http://arabicorpus.byu.edu), which contains newspapers from the period between 1996 and 2010.

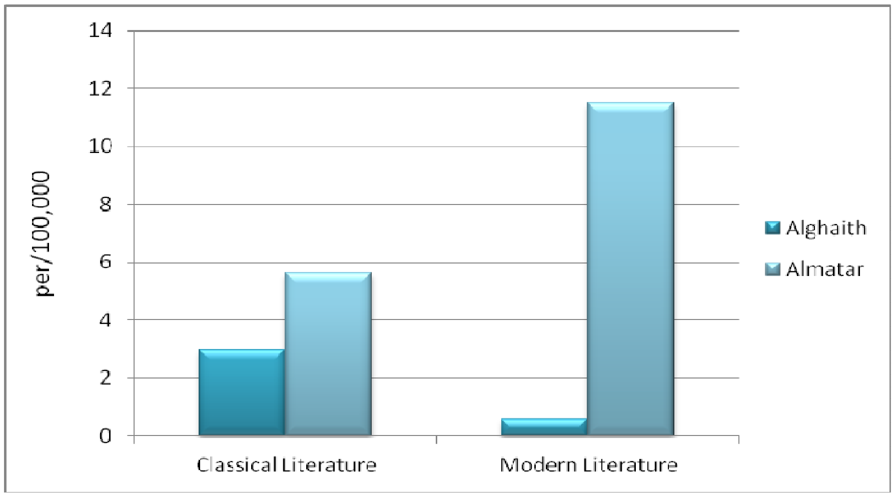


FIGURE 1: Frequency rates of the words *Alghaith* and *Almatar* in classical and modern literature.

Another example of lexical change is the emergence of the use of the word *Azya'a*, which means *fashion* in English. The frequency of this word in KSUCCA is only 1, and it was not used to mean fashion, as we know it today, it is merely the sum of the word *Zay*, which means *clothes* or *costume*. On the other hand, the word *Azya'a* is used very frequently (0.9/100,000) in contemporary newspapers with the meaning of fashion. Figure 2 shows the frequency rate of the word *Azya'a* in KSUCCA compared to its frequency rate in the All Newspapers corpus[7]. This indicates severe cultural and social changes caused by the influence of the western culture on Arabic societies, and the way that Arabic language tries to adapt and cope with these changes.

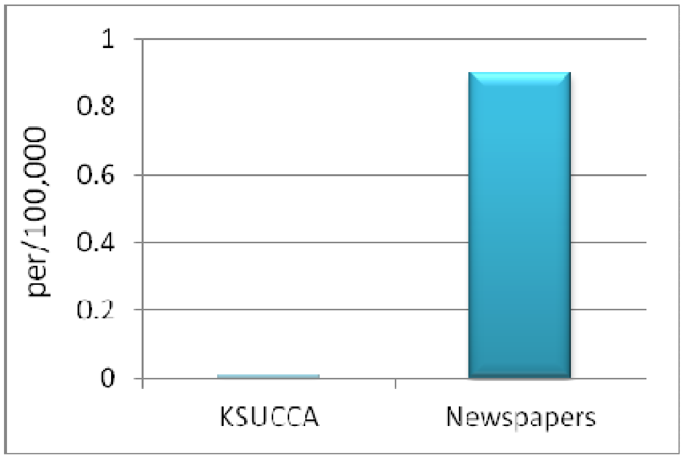


FIGURE 2: Frequency rates of the word *Azya'a* in KSUCCA and All Newspapers corpus.

3.2 Phraseological Change

Phraseology is the study of multi-word units in language, which have a range of subtypes [18]. In this section, we will discuss how KSUCCA can be used to detect phraseological changes between Classical Arabic and MSA. We will focus on collocations, which are considered one of the subtypes of the multi-word units included in phraseology. A collocation is the tendency of two or more words to appear together conveying a meaning by their association [19]. The Arabic word *Raghaba aan* (رغب عن) is an example of a collocation; it means *abstain from* in English. This

7 http://arabicorpus.byu.edu.

collocation was common in Classical Arabic; Table 3 shows the frequency rate of its usage in KSUCCA (0.334/100,000). However, it is no longer used very frequently in MSA; it only appeared 24 times in the whole 135,360,804 words All Newspapers corpus, with a frequency rate of (0.018/100,000)[8].

	KSUCCA	**Newspapers**
Ragheba aan	0.334	0.018

TABLE 3: Frequency rates of the collocation *Raghiba aan.*

Figure 3 visualizes the severe difference in frequency rates of the collocation *Raghiba aan* in both corpora.

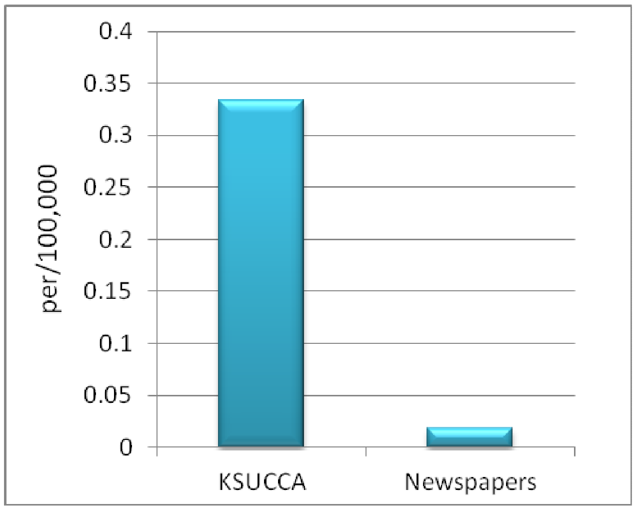


FIGURE 3: A comparison of the frequency rates of *Raghiba aan.*

There are many other collocates that were common in the classical period of Arabic and are no longer used much; they are being replaced by new ones. One of them is the collocation referring to the name of the holy mosque in *Almadinah Almonaurah* city; this mosque is commonly referred to, nowadays, as *Almasjid Alnabawy*, which means *The Prophetic Mosque*. However, a simple search for this collocation in KSUCCA would reveal nothing, because that mosque was only referred to at that period of time as *The Mosque of Allah's Apostle* or, in Arabic, as *Masjid Rasool Allah*. Table 4 shows the frequency rates of the two collocates in both KSUCCA and the All Newspapers corpus.

	Newspapers	**KSUCCA**
The mosque of Allah's apostle	0.01	0.47
The prophetic mosque	0.115	0

TABLE 4: The Frequency Rates of The Two Collocates.

It is clear from Figure 4 that the new collocation *Almasjid Alnabawy* is the common and most used collocation, nowadays, to refer to the holy mosque, and that the original name, *Masjid*

8 http://arabicorpus.byu.edu.

Rasool Allah, is barely used. These two examples can be seen as a consequence of the principle of least effort, also known as Zipf's law [19], where people try to exchange long terms by shorter ones, and which is considered to be responsible for many linguistics changes [20].

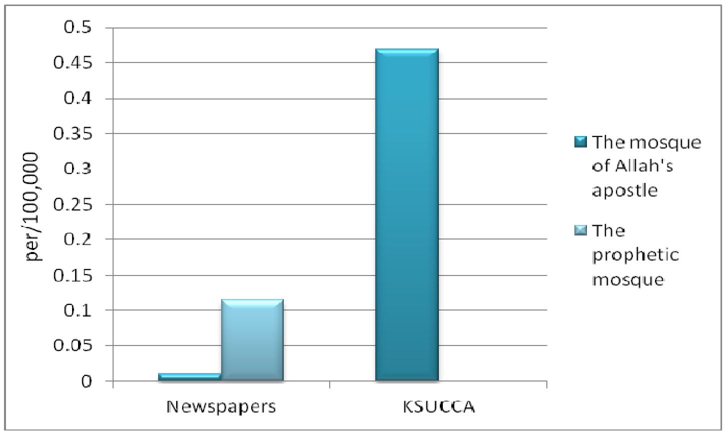


FIGURE 4: A Comparison Between The Frequency Rates of The Two Collocates.

3.3 Vocabulary Change

The Hadith, classical poems and other classical writtings are very rich in vocabulary, which is an indication of the very solid linguistic base that pople had at that era. Unfortunately, this is not the case with MSA; any regular Arabic reader can sense the decline in vocabulary wealth in MSA writings compared to Classical Arabic. One way to prove this assumption, is to study the number of roots used in samples of Classical Arabic writings from different genres and compare them to other samples from MSA with the same genres; where a root is the three letters word that form the basic source of all the forms of a given word.

We have chosen three samples, 100 word each, from KSUCCA covering the following genres: Quranic studies, philosophy and ethics and morals. To represent the MSA, we have also chosen three samples, 100 word each, falling under the same genres from the the Comprehensive library "*Almaktabah Alshamilah*" site[9] . Then we used Alkhalil morphological analyzer [21] to extract the roots of the words in each sample. Table 5 shows the numbers of unique roots extracted from each sample.

Genre	No. of roots in Classical Arabic samples	No. of roots in MSA samples
Quranic studies	58	51
Philosophy	53	49
Ethics and morals	47	31

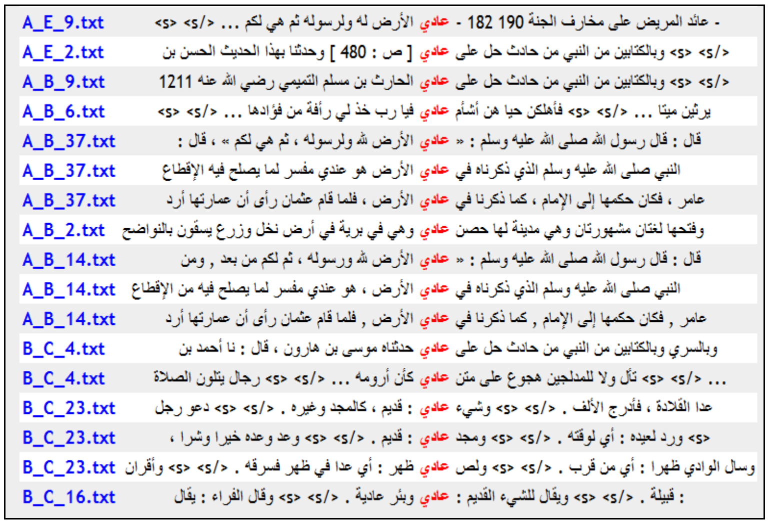
TABLE 5: Number of Unique Roots In Each Sample.

It is obvious that the number of roots in the Classical Arbic samples are larger than their equivalent samples from MSA. This can be considered as an evidence of the decline of the average vocabulary in MSA writings, which is another sign of language decay.

9 http://shamela.ws

3.4 Semantic Change

Many Arabic words went through a series of semantic changes from the classical period until now, and sometimes their meanings were completely altered. One of these words is the word *Aady* (عادي), which was originally used to mean *old* or *an aggressor.* This word was not common in Classical Arabic, which is confirmed by looking at the concordance[10] of the word *Aady* in KSUCCA in Figure 5. The word is used with law frequency rate as in Table 6.



A_E_9.txt	- عائد المريض على مخارف الجنة 190 182 - عادي الأرض له ولرسوله ثم هي لكم ... </s> <s>
A_E_2.txt	</s> <s> وبالكتابين من النبي من حادث حل على عادي [ص : 480] وحدثنا بهذا الحديث الحسن بن
A_B_9.txt	</s> <s> وبالكتابين من النبي من حادث حل على عادي الحارث بن مسلم التميمي رضي الله عنه 1211
A_B_6.txt	يرثين ميتا ... </s> <s> فأهلكن حيا هن أشأم عادي فيا رب خذ لي رأفة من فؤادها ... </s> <s>
A_B_37.txt	قال : قال رسول الله صلى الله عليه وسلم : « عادي الأرض لله ولرسوله ، ثم هي لكم » ، قال :
A_B_37.txt	النبي صلى الله عليه وسلم الذي ذكرناه في عادي الأرض هو عندي مفسر لما يصلح فيه الإقطاع
A_B_37.txt	عامر ، فكان حكمها إلى الإمام ، كما ذكرنا في عادي الأرض ، فلما قام عثمان رأى أن عمارتها أرد
A_B_2.txt	وفتحها لغتان مشهورتان وهي مدينة لها حصن عادي وهي في برية في أرض نخل وزرع يسقون بالنواضح
A_B_14.txt	قال : قال رسول الله صلى الله عليه وسلم : « عادي الأرض لله ورسوله ، ثم لكم من بعد , ومن
A_B_14.txt	النبي صلى الله عليه وسلم الذي ذكرناه في عادي الأرض ، هو عندي مفسر لما يصلح فيه من الإقطاع
A_B_14.txt	عامر , فكان حكمها إلى الإمام , كما ذكرنا في عادي الأرض , فلما قام عثمان رأى أن عمارتها أرد
B_C_4.txt	وبالسري وبالكتابين من النبي من حادث حل على عادي حدثناه موسى بن هارون ، قال : نا أحمد بن
B_C_4.txt	... </s> <s> تأل ولا للمدلجين هجوع على متن عادي كأن أرومه ... </s> <s> رجال يتلون الصلاة
B_C_23.txt	عدا القلادة ، فأدرج الألف . </s> <s> وشيء عادي : قديم ، كالمجد وغيره . </s> <s> دعو رجل
B_C_23.txt	<s> ورد لعيده : أي لوقته . </s> <s> ومجد عادي : قديم . </s> <s> وعد وعده خيرا وشرا ،
B_C_23.txt	وسال الوادي ظهرا : أي من قرب . </s> <s> ولص عادي ظهر : أي عدا في ظهر فسرقه . </s> <s> وأقران
B_C_16.txt	: قبيلة . </s> <s> ويقال للشيء القديم : عادي وبئر عادية . </s> <s> وقال الفراء : يقال

FIGURE 3: A Concordance of The Word Aady From KSUCCA.

However, the meaning of this word have drift from time to time until it is used now in MSA to mean *normal* or *ordinary*. This drift in its meaning was accompanied, of course, by an increase in its frequency rate nowadays compared to its limited use in Classical Arabic, as can be seen in Table 5.

	KSUCCA	Newspapers
Aady	0.9	6.25

TABLE 6: The Frequency Rates of The Word *Aady.*

4. CONCLUSION

Most of the previous work on Arabic historical linguistics was not corpus-based; one major reason for that is the fact that there are no available corpora of Classical Arabic that are large enough to conduct such studies. In addition to the fact that most traditional Arabic linguists are not familiar with the use of computerized corpora in language researches.

In this paper, we present KSUCCA a pioneering 50 million words corpus of Classical Arabic with various genres and sub genres that can be used in various types of Linguistic and Computational Linguistic research. We also showed how can KSUCCA be used to detect some interesting lexical, Phraseological, vocabulary and semantic changes in Arabic language. We believe that the construction of KSUCCA and the work presented in this paper will encourage Arab linguists to take steps forward in exploring corpus-based historical linguistics and discovering other interesting aspects of language change.

10 Using Sketch Engine (http://www.sketchengine.co.uk).

5. REFERENCES

[1] T. Bynon. Historical Linguistics. Cambridge University Press, 1977.

[2] C. F. Meyer. English corpus linguistics: An introduction. Cambridge University Press, 2002.

[3] M. Kytö. "Manual to the diachronic part of the Helsinki corpus of English texts, 3rd ed." University of Helsinki, 1996.

[4] M. Davies. "Expanding horizons in historical linguistics with the 400-million word Corpus of Historical American English." Corpora, 2012.

[5] M. Mansour. "The absence of Arabic corpus linguistics: a call for creating an Arabic national corpus." International Journal of Humanities and Social Science, vol. 3, no. 12, 2013.

[6] L. Al-Sulaiti and E. Atwell. "The design of a corpus of contemporary Arabic." International Journal of Corpus Linguistics, vol. 11, pp. 135-171, 2006.

[7] L. Al-Sulaiti and E. Atwell. "Extending the Corpus of Contemporary Arabic." In Proceedings of Corpus Linguistics conference, University of Birmingham, UK, 2005.

[8] S. Alansary, N. Magdi and N. Adly. "Building an international corpus of Arabic (ICA): Progress of Compilation Stage." In 7th Int. Conference on Language Engineering, Cairo, Egypt, pp.1-30, 2007.

[9] M. Sawalha and E. Atwell. "Constructing and Using Broad-coverage Lexical Resource for Enhancing Morphological Analysis of Arabic." In proceeding of: Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, Valletta, Malta, 2010.

[10] A. Alfaifi and E. Atwell. "Arabic Learner Corpus v1: A New Resource for Arabic Language Research." In proceedings of the Second Workshop on Arabic Corpus Linguistics (WACL-2), Lancaster University, UK, 2013.

[11] A. Elewa. "Did they translate the Qur'an or its exegesis?." 3rd Languages and Translation Conference and Exhibition on Translation and Arbization in Saudi Arabia, Riyadh, Saudi Arabia, 2009.

[12] M. Eid. Manifestations Emerging on Arabic. A'alam Alkutub, Cairo, pp. 20, 1980.

[13] M. Alrabiah, A. Al-Salman and E. Atwell. “The design and construction of the 50 million words KSUCCA King Saud University Corpus of Classical Arabic." In Second Workshop on Arabic Corpus Linguistics (WACL-2), Lancaster University, UK, Monday 22nd July 2013.

[14] W. N. Francis and H. Kucera. "Brown Corpus Manual: Manual Of Information To Accompany A Standard Corpus of Present-Day Edited American English, for use with Digital Computers." Internet: http://khnt.hit.uib.no/icame/manuals/brown/INDEX.HTM, 1964 [March. 2, 2014].

[15] S. Johansson, E. Atwell, R. Garside and G. Leech. "The Tagged LOB Corpus: Users' manual." ICAME, The Norwegian Computing Centre for the Humanities, Bergen University, Norway, 1986.

[16] L. Burnard. "British National Corpus: User's reference guide for the British National Corpus." Oxford, Oxford University Computing Service, 1995.

[17] A. Alaskari, Linguistic Differences. (in Arabic), Dar Alkutub Alelmiah, 2010.

[18] S. Granger and F. Meunier. Phraseology: An Interdisciplinary Perspective. Amsterdam: John Benjamins, 2008.

[19] C.D. Manning and H. Schuetze. Foundations of Statistical Natural Language Processing, 1st ed., The MIT Press, 1999.

[20] C.M. Millward. A Biography of the English Language. 2nd ed. Harcourt Brace, 1996.

[21] A. Boudlal, A. Lakhouaja, A. Mazroui, A. Meziane, M. Ould Abdallahi Ould Bebah and M. Shoul. "Alkhalil MorphoSys: A Morphosyntactic analysis system for non vocalized Arabic." Seventh International Computing Conference in Arabic (ICCA 2011), Riyadh, 2011.

INSTRUCTIONS TO CONTRIBUTORS

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.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Started with Volume 5, 2014, IJCL aim to appear with more focused issues related to computational linguistics studies. Besides normal publications, IJCL intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

We are open to contributions, proposals for any topic as well as for editors and reviewers. We understand that it is through the effort of volunteers that CSC Journals continues to grow and flourish.

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 Acquisition of Word-order
- Models that Combine Linguistics Parsing
- Models that Employ Statistical/probabilistic Gramm
- Models that Employ Techniques from machine learning
- Natural Language Processing
- Quantitative Linguistics
- Speech Analysis/Synthesis
- Speech Recognition/Understanding
- Spoken Dialog Systems
- Web Information

CALL FOR PAPERS

Volume: 5 - **Issue:** 3

i. Paper Submission: May 31, 2014 **ii. Author Notification:** June 30, 2014

iii. Issue Publication: July 2014

CONTACT INFORMATION

CSC PUBLISHERS © 2014
COMPUTER SCIENCE JOURNALS SDN BHD
B-5-8 PLAZA MONT KIARA
MONT KIARA
50480, KUALA LUMPUR
MALAYSIA

PHONE: 006 03 6204 5627
FAX: 006 03 6204 5628
EMAIL: cscpress@cscjournals.org