An improved root extraction technique for Arabic words

May Y. Al-Nashashibi, D. Neagu
Department of Computing, University of Bradford,
Bradford, UK,
e-mail: (M.Y.A.Al-nashashibi,D.Neagu)@bradford.ac.uk

Ali A. Yaghi
Department of Computer Science, Petra University,
Amman, Jordan,
e-mail: Ali.Yaghi@uop.edu.jo

***Abstract*— Arabic text interpretation depends among other things on a pre-processing stage in extracting effectively a correct stem or root. We address in this work a linguistic approach for root extraction as a pre-processing step for Arabic text mining. The linguistic approach is composed of a rule-based light stemmer and a pattern-based infix remover. We propose an algorithm to handle weak, eliminated-long-vowel, hamzated, and geminated words since the linguistic approach does not handle such cases and a reasonably large portion of Arabic words in texts are irregular. The accuracy of the extracted roots is determined by comparing them with a predefined list of 5,405 triliteral and quadriliteral roots. The linguistic approach performance (with and without the proposed correction algorithm) was tested on an in-house text collection of eight categories. The proposed correction algorithm improved the accuracy of the linguistic one by about 14%.**

Keywords: Arabic Root Extraction; Natural Language Processing; Text Mining; Rule-Based Stemming.

I. INTRODUCTION

Many Text Mining (TM) and Natural Language Processing (NLP) techniques and algorithms have been explored or are strictly related to English language but few have been proposed for Arabic text automatic interpretation such as [8]. However, applying TM techniques require first a preprocessing stage that would remove punctuation marks, function words and return the remaining words to their stems (i.e. removing prefixes or suffixes if available) or to their roots (i.e. removing prefixes and/or suffixes and/or infixes if present according to specific patterns). For English, researchers differ on the effectiveness of using stems in the representation of documents on improving TM but agree that the stemming step is done in order to reduce the high dimensionality of the document(s) [15]. Our aim is two fold, first to use/improve Arabic stemming/root extraction technique and use the resulting stems and/or roots in the next step. Our second aim is to use the resulting stems/roots instead of their respective Arabic words when applying TM methods and study whether substituting words by either stems or roots improves the performance of such methods and if so whether such improvements are significant. However, in this work, we will concentrate on investigating/improving a rule-based light stemmer/root extractor technique based on the work of Al-Ameed [1] on Arabic. Al-Ameed method [1] was chosen here since it reported an accuracy of root extraction of more than 90% when tested on many derivations of many roots. We propose to improve Al-Ameed's method by handling irregular words in Arabic such as weak, two-letter geminated, hamzated, and eliminated-long-vowel[1] words. Such cases are available in about 13%, 7%, 11% and 2% (12% of weak words) of texts respectively[2], so presenting about 30% of words in Arabic texts. Such percentages are high and thus the importance of our proposed correction algorithm.

The remainder of the paper is organized as follows: Section II includes a brief review of related work and the Arabic language morphology. Section III discusses the text collection used by the root extraction approach and describes briefly the construction of function words list. Section IV discusses the linguistic approach along with our proposed method for improving it. Section V presents the evaluation criteria and experimental results. Finally, conclusions and discussions of future work are found in Section VI.

II. RELATED WORK AND BACKGROUND

A. Related Work

Many research works [2], [14] have been proposed and implemented using different techniques for stemming/morphological analysis for Arabic especially for Information Retrieval (IR). After extensive [14] investigation it was concluded that for IR using light stemming provided the highest performance of IR followed by using root extraction. Both improved IR performance but light stemming outperformed root extraction (for detailed review of this topic, the reader is kindly referred to the works in [2] and [14]). For TM on Arabic, few works, as far as we know, have been conducted to investigate the effect of using stems or roots instead of words on Text Classification (TC) performance such as [8], [12], and [16]. There is a discrepancy among their results. On the one hand some of above works reported that light stemming degraded TC performance. On the other hand, others reported that using either stems or roots improved TC performance. It must be noted here that in all mentioned works above, no significance tests were reported.

In all methods used for Arabic morphological analysis, one must address the issues of under-stemming and over-stemming when developing the stemming or morphological analysis method [2]. Other issues that require handling for

1 We use here Haywood and Nahmad 1998 terminology for describing Arabic irregular forms.

2 Percentage values presented here are gathered by 1st author from 40 texts chosen arbitrarily in the collection.

stemmers are compound words, proper nouns, foreign Arabized words (i.e. transliterations), irregular words (i.e. weak, eliminated-long-vowel, hamzated and geminated words). Many of the proposed stemmers, available in literature, do not handle some special cases in Arabic such as weak, eliminated-long-vowel, hamzated or geminated words properly. If they do, they handle part of these cases but not all and none handle eliminated-long-vowel words, except for the works of Beesley [5] and El-Sadany and Hashish [10]. Thus, they perform poorly. Although few of the available stemmers such as the Khoja stemmer[3] do handle weak and geminated words yet to our knowledge none so far handle all. It is note worthy that in El-Sadany and Hashish [10] work, no results were provided of the system implemented. Also, in Beesley's work [5], the Xerox demo [4] is available, and although efficient, it requires usually a relatively long time to provide the required roots (from about 1 hour to 7 day according to number and type of words given). Thus, there is a need to build an algorithm that provides the correct root for such irregular cases. Here we propose addressing some of the weaknesses presented above (i.e. handling weak, geminated, hamzated and eliminated-long-vowel words) by using and improving a linguistic approach derived from the work of Al-Ameed [1] as will be explained thoroughly in Section IV.

B. Background

Arabic language is one of the Semitic languages that have [11] 28 letters, all consonants: three of these letters are also used as long vowels '***A***', '***w***', and '***y***'[5]. Arabic language has also short vowels that are not letters (also named diacritics). The presence of such vowels in a word reduces the ambiguity of the meaning of that word and simplifies checking its grammar and extracting the proper root. Arabic has also other markers such as nunation [11], assimilation, and kasheeda (used to elongate the appearance of a letter in words). These different aspects are highly important in spoken Arabic and in natural Arabic language understanding. However, Modern Standard Arabic (MSA) usually does not include short vowels, nunations or assimilation markers in printed, electronic texts, and the understanding thus correct pronunciation of the word is left for the native Arabic reader to correctly decide within its context. Verbs are categorized in Arabic [11] as sound and unsound verbs. The sound verbs are verbs with no long vowels or hamza. The unsound verbs are further categorized into weak and compromising verbs. Weak verbs have a long vowel as one or more of its original letters, while comprising verbs are either hamzated (i.e. have the hamza in it) or geminated (i.e. have a letter that is doubled). In specific cases of weak verbs, these verbs have roots that are different from them by replacing their long vowel with another according to specific rules in Arabic. Also, in other specific cases long vowels in weak words are deleted from such words according to specific rules. Such words are called eliminated-long-vowel ones. In MSA usually markers indicating germination are omitted. If the above cases are not handled, the results of stemming will be incorrect. As mentioned in Section I, Arabic stemmers lack the capacity of handling properly this category of irregular verbs and in this paper we propose a method for handling it. Also, words, whether verbs or nouns, are represented in Arabic by their patterns. A pattern is a form that Arabic linguists usually use to present words that might contain affixes in specific positions in a word. For any triliteral-root word, its basic three letters are presented by the letters *f, E, l* respectively. An example of word presentation using a pattern is the word *IstxdAmhm* that possibly means "their use" and has the root *xdm*. Thus, this root has the pattern *fEl*. So, the word *IstxdAmhm* would have the presentation ***IstfEAlhm*** where here ***Ist***, ***hm*** are the prefix and suffix respectively and *A* inside *fEAl* is the infix. Thus, the pattern presenting this word is here *fEAl*. Next, the construction of both text collection and function words list are presented.

III. TEXT COLLECTION AND FUNCTION WORDS LIST

In order to support and test our work we collected 380 Arabic texts (about 193,500 words) arbitrarily chosen from various online Arabic newspapers, magazines, academic and other sources published online in the period 23/7/2008 - 1/2/2009, according to eight subject categories: politics, economics, social, sports, music, education and health, arts, culture and literature, and religious texts. In each category about 50 texts were chosen randomly. These categories were chosen in such a way so that they would contain articles, short stories etc.

We have constructed the Arabic function words list from 2,549 words [11]. Examples of function words are the separate prepositions, personal pronouns, demonstrative pronouns, interrogative pronouns, relative pronouns, conjunctions, and interjections. Also, '*kAn w OkhwAthA*' or similar verbs such as '*OSbH*' or '*mA zAl*' or '*OmsY*' are considered function words. Dual and plural forms of function words are added to its list. Both, constructed text collection and function word list are used in Section IV for experiments.

IV. ROOT EXTRACTOR CONSTRUCTION

In this work, we are interested in using and improving a rule-based root extraction technique to extract the stems/roots of words in texts as a preprocessing step for TM. The rule-based approach does not handle vowelized words, names of places, countries, cities, months, transliterations, weak, eliminated-long-vowel, hamzated or some cases of geminated words. However, our proposed correction algorithm will handle weak, eliminated-long-vowel, hamzated or geminated words so that the rule-based approach's performance is improved. We will use the rule-

3 Shereen Khoja. Stemming Arabic Text. 1999. Available online at URL: http://zeus.cs.pacificu.edu/shereen/research.thm/. [Accessed 29/11/2009].

4 Xerox demo can be found at http://www.xrce.xerox.com/Research-Development/Historical-projects/Linguistic-Demos/Arabic-Morphological-Analysis-and-Generation

5Transliterations used here are that of Buckwalter' found at: http://www.qamus.org/transliteration.htm based approach on the in-house text collection and test its performance with and without our proposed algorithm. Both approaches are part of a preprocessing step before TM. Their performance will be presented in Section V. The first process before root extraction is to remove from texts English letters, punctuation marks, nunations, assimilation marks, short vowels, kasheeda, function words or numerals. Next, the rule-based approach is described, and then we present the proposed method for correcting irregular words.

A. The Linguistic Approach

The linguistic root extractor is implemented starting from the work of Al-Ameed [1] and is composed of two parts. The first part is a rule-based light stemmer where prefixes and suffixes are removed from the word according to specific rules. The second part is a pattern-based infix remover where infixes are removed from the word according to also specific patterns. These two parts are in the whole algorithm (named here the ***Rule-Based*** algorithm). As cited in Al-Ameed's work, this algorithm was tested against the work of [6] only due to the fact that the later work gave better results than both Darwish's and Larkey's works [7], [13]. The performance of the two algorithms in [1] showed that Al-Ameed's algorithm gave much better performance results. In Section V, the results of this rule-based root extraction approach will be compared with that of the improved one. As can be seen from the algorithm below, it outputs two files. The first contains the stems and the second contains the roots. It should be mentioned here that during the construction of this method, we removed the function words before the algorithm is used and at the middle of it after stemming is performed. However, from preliminary experiments, we found that this removes words that are not function words. Thus, we decided on removing function words only once before implementing this algorithm. The results of implementing this method along with the proposed algorithm are presented in Section V.

Rule-Based Algorithm

```
Inputs: Set of preprocessed documents D = {d1,
d2, …..,dn}, Predefined root lists
Outputs: triliteral and quadriliteral roots for
each new document new_di_2 in output set DD2, stems
for each new document new_di_1 in output set DD1
START
For each document di do {
   LastWord = Count_No_Words(di)
     For j = 1 to LastWord in di do {
       LastLetter = Count_No_Letters(wj,c), m = 0
       If (LastLetter <= 3) {Final_Wordj = wj, m = 1,
go to *}
       New_Wordj = Light_Stemmer_Algorithm(wj)
       LastLetter = Count_No_Letters(New_Wordj,c)
       Write New_Wordj to output document new_di_1
If (LastLetter <= 3) {Final_Wordj = New_Wordj, go
to *}
       New_Wordj = Inf_Remover_Algorithm(New_Wordj)
       LastLetter = Count_No_Letters(New_Wordj,c)
       Write Final_Wordj to output document new_di_2
*      if (m == 1) {Write Final_Wordj to output
document new_di_1}
count = Count_Correct_Roots(Final_Wordj, count)}
Accuracy_of_document_ new_di_2 = (count/LastWord)
100%}
END
```

Next, the light stemmer and the infix remover are explained.

1) Light Stemmer

The algorithm (named here ***Light Stemmer*** algorithm) uses four predefined prefix lists and two predefined suffix lists as shown in Table I. In this approach the concentration of affix removal is on the augmented letters in '*sOltmwnyhA*'. It first replaces specific letters (if appearing in word) by a specific one (i.e. a normalization step). Secondly, it deletes the definite article in Arabic '*Al*' from a word if the word starts with it. Thirdly, it applies the rules using the predefined prefix lists and the predefined suffix lists. It starts with removing single largest available suffix term, then removes the largest single prefix term and at the end it removes a single largest remaining suffix term (if any).

TABLE I. AL-AMEED'S PREFIX AND SUFFIX LISTS

Proposed Prefix lists	***Prefix List1***	{ *A, t, n, w, y , m , k , b , f , l* }
	Prefix List2	{ *Al, ll, sy, sA, st, sn, kA, fA, bA, ly, lt, ln, ft, fy, fn, wt, wy, wm, wA, wb, bm, km, mt, fl, An, mn, lA, wl, wn, wk, fb, fm, lm, yt, yn, tt, tn, tm, nt, nn, At* }
	Prefix List3	{ *wAl, bAl, fAl, kAl, wll, wsy, wst, wsn, wsA, wlA, wly, wlt, wln, fsy, mst, Ast, Alm, AAl, sAl, fAn, AA, yst, nst, tst* }
	Prefix List4	{ *wbAl* }
Proposed Suffix lists	***Suffix List1***	{ *A, t, n, w, y , h , k , Y, p* }
	Suffix List2	{ *An, yn, wn, At, hm, hn, hA, km, kn, nA, wA, tm, ny, tn, th, yh, A`, yA, tA, tk, yp, np, nh ty, Aw, nk, hmA, kmA, thm, tkm, wt* }

Light Stemmer Algorithm

```
Inputs: Set of preprocessed documents D = {d1,
d2, …..,dn}, PrefixList1, PrefixList2, PrefixList3,
PrefixList4, SuffixList1, SuffixList2, 3 Predefined
Replace lists
Outputs: List of triliteral and quadriliteral stems
for each new document new_di_1 in set DD1
START
For each document di do {
  LastWord = Count_No_Words(di)
    For j = 1 to LastWord in di do {
     LastLetter = Count_No_Letters(wj,c)
     If (LastLetter <= 3) then {New_Wordj = wj, go
to *}
     For k = 1 to LastLetter in wj do {
       Perform Normalization for specific letters
       Delete_AL from beginning of word
       New_Wordj = wj
       If ($l_{LastLetter}$ ∈ { A, t, n, w, y, h, k, `, m,
Y }) {{ New_Wordj = RemoveSuffixes1(New_Wordj),
LastLetter = Count_No_Letters(New_Wordj,c)}
       If (LastLetter <= 3) {{New_Wordj = New_Wordj,
go to *}}}
       New_Wordj = RemovePrefixes(New_Wordj)
       LastLetter = Count_No_Letters(New_Wordj,c)
       If (LastLetter <= 3) {New_Wordj = New_Wordj,
go to *}
       If (lLastLetter ∈ { A, t, n, y, h, r , k, `,
m, Y }) {New_Wordj = RemoveSuffixes2(New_Wordj)
     * Put New_wordj in new_di_1}}}
END
```

2) Infix Remover

The infix remover uses the stemmed words resulted from the light stemmer implementation and two predefined pattern lists. The first list is for triliteral words and the second list is for quadriliteral words as shown in Table II. Also, using the predefined pattern lists, the infix remover applies the rules of the first predefined pattern list for triliteral words, then it applies the rules of the second predefined pattern list for quadriliteral words such that at the end it provides the root for words. From the patterns shown, this approach handles some frequent cases of broken plurals. Nevertheless, it does not handle weak, hamzated or eliminated-long-vowel words but handles partially geminated words.

TABLE II. AL-AMEED INFIX PATTERN LISTS

Triliteral Infix Patterns	*{fEl, fAEl, fAEwl, fEA'l, fEAl, fEwl, fEyl, fwEl, fwAEl, fwAEyl, fyEl, fyAEl, fEyEyl, fEyEAl, fwyEl, ftEl, ftEAl, fAEyl, fEwEl}*
Quadriliteral Infix Patterns	*{fEll, fEAlyl, fEAll, fElAl, fElwl, fElyl, fEyll}*

Inf_Remover Algorithm

```
Inputs: Set of stemmed documents DD1 = {new_d1_1,
new_d2_1,...,new_dn_1}, Infix Pattern List1, Infix
Pattern List2.
Outputs: List of triliteral and quadriliteral roots
for each new document new_di_2 in output set DD2
START
For each document new_di_1 do {
  LastWord = Count_No_Words(new_di_1)
  For j = 1 to LastWord in new_di_1 do {
   LastLetter = Count_No_Letters(New_Wordj,c)
   If (LastLetter <= 3) then {Final_Wordj =
New_Wordj, go to *} % pattern is fEl
   If (LastLetter == 6) {{if patterns fwAEyl or
fEyEyl/fEyEAl or fEAlyl are satisfied
respectively{ delete w and/or A and/or y according
to pattern, Final_wordj = New_Wordj , go to *}}}
   If (LastLetter == 5) {{if any of patterns
fyAEl/fwAEl or fAEwl or fAEyl or fwyEl or ftEAl or
fEA'l or fEAll or fEyll or fEwEl or
fEl|l/fElwl/fElyl are satisfied
respectively{ delete extra vowels, Final_Wordj =
New_Wordj , go to *}}}
   If (LastLetter == 4) {{if any of patterns
fEAl/fEwl/fEyl or fAEl/fwEl/fuel/ftEl or fEll are
satisfied respectively{ Delete extra vowels,
Final_Wordj = New_Wordj, go to *}}}
  * Put Final_Wordj in new_di_2}}
END
```

B. Root Correction for Specific Words

We propose hereby a simple algorithm in order to correct irregular words. This proposed algorithm handles weak words by replacing the long vowel in it by another long vowel according to specific rules in Arabic. Also, this algorithm handles eliminated-long-vowel words where, for some of these triliteral words, when for example their tense is changed from past to present tense the vowel is cancelled and an extra letter is added to that word (whether at beginning or end). A third issue that this algorithm handles is two-letter geminated roots as '*rd*' when it starts/ends with any of '*y*, *t*, *n*, or *A*'/'*t*, *p*, or *h*' respectively then the extra letter must be deleted and the letter '*d*' is doubled. Also, this algorithm corrects specific cases of Hamza (if present) in a root. The algorithm proposed includes 5,737 possible corrections of words in 71 predefined lists (collected from [3] and [4]) according to specific rules for only triliteral roots. Since this algorithm corrects some special triliteral words then it is named here ***Correction*** algorithm.

The accuracy of the rule-based algorithm is calculated by first comparing each extracted root with a predefined list of triliteral and quadriliteral list of 5,405 roots (4,655 triliteral roots and 750 quadriliteral roots) gathered from ([3], [4], and [9]), then counting the roots that match the ones in the predefined list, and finally calculating the percentage of correct roots. The same applies when the correction algorithm is added at the end of the rule-based approach. In other words when our algorithm is added at the end of the rule-based approach, first the extracted root is checked whether in the predefined root list. If not, the extracted root is checked if triliteral and if any of the rules in the correction algorithm apply for it. Then, if it belongs to the predefined list for that rule and if so, the root is thus changed to the proper one according to specific rules.

Correction Algorithm

```
Inputs: Arabic triliteral word, 71 predefined lists
Output: corrected triliteral Arabic word
START
Let ch1 <- first character of Word; ch2 <- second
character of Word; ch3 <- third character of Word
If ch1 is either y, t, &, A, n, or } {% (hamzated,
eliminated-long-vowel cases, or both)
     if word is in specific lists { change ch1
according to specific cases, go to *.}}
If ch3 is either y, Y, &, }, w, or A {% (weak,
eliminated-long-vowel, hamzated cases)
     if word is in specific lists { change ch3
according to specific cases, go to *.}}
If ch2 is either }, w, O or A {% (weak, eliminated-
long-vowel, hamzated cases)
     if word is in specific lists { change ch2
according to specific cases, go to *. }}% handling
geminated words (2 different rules)
If ch1 is either " t, y, n, or A {
    if word is in specific lists { delete ch1 and
double ch3 according to specific cases, go to *}}
If ch3 is either h, or p {
     if word is in specific lists { delete ch3 and
double ch2 according to specific cases, go to *}}%
handling one geminated, 6 hamzated (some are
composite with eliminated-long-vowel cases) or
eliminated-long-vowel cases (18 cases for pattern
fEt, 4 for flt) (23 different rules)
If ch3 is t {if word is in specific lists {
  either Delete ch3 and double ch2 according to
specific cases %(geminated cases)
  OR replace ch3 by only one of letters A, y, w, or
Y according to specific rules % (eliminated-long-
vowel & hamzated cases)}}
* Return corrected word
END
```

As can be seen from the algorithm above that not only rules were used to specify each case but also following the rule the word was compared with a predefined root list of words that do indeed follow that rule in Arabic. This additional step was performed in order to minimize the effect of wrongly extracted roots that apply to the rule but are not the correct ones.

V. Experimental Results

Here experimental results are presented of the accuracy for the linguistic approach with/without our proposed correction algorithm. Results in Table III show that adding our proposed correction algorithm to linguistic approach increased the latter's accuracy by about 14%. Also, an in-coder[6] analyzed the algorithms performance and found that the correction algorithm relatively improved the rule-based one by about 10%. In Table III and Fig. 1 the following stand for: **RB-A**: ***Rule-Based*** algorithm, **RB-A-corr**: ***Rule-Based and Correction*** algorithm. Al-Ameed's algorithm was tested in [1] using a specially customized test set which was composed of 199,584 distinct words derived from 24 distinct triliteral roots and 119,700 words derived from 25 distinct quadriliteral roots. Since this test set is not available to us, we used our text collection to test both Al-Ameed's algorithm along with our proposed correction one.

TABLE III. Performance of Algorithms in All Categories

	RB-A(%)	RB-A-corr (%)
Politics	58.89	73.3
Economics	58.16	71.39
Religious Texts	62.99	75.01
Social	60.56	74.79
Music	58.7	73.78
Educational ...	60.67	74.81
Sports	56.91	70.37
Arts ..	61.41	74.27
Average	***59.79***	***73.47***

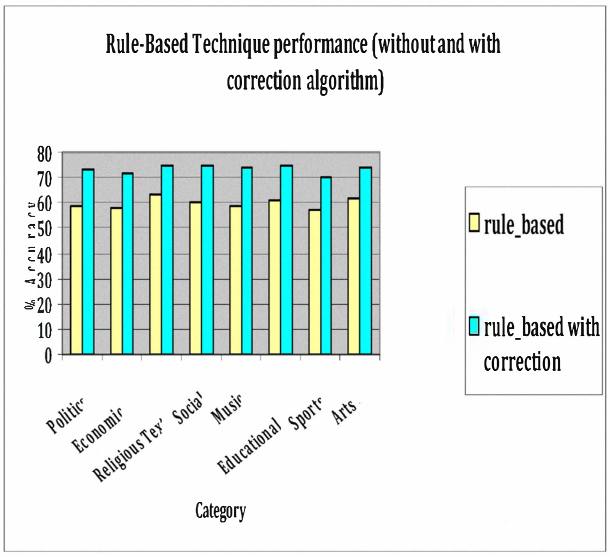

Figure 1. Comparison of algorithms performance.

6 The 1st author of this paper.. This was done as a preliminary step using 40 texts (5 from each class chosen arbitrarily).

VI. Conclusions and Future Work

In this paper we present a new algorithm that is used to: 1- replace long vowels appearing in words that require to be changed in order to have the correct root for a word according to specific rules; 2- delete an extra letter (at the beginning or end) of two-lettered geminated roots and doubling the second letter; 3- handle specific cases of eliminated-long-vowel words; 4- handle specific cases of hamzated words. This correction algorithm was included in the rule-based approach for root extraction proposed in [1]. Also, this algorithm improved the performance of the rule-based one by about 14% (with a relative improvement of about 23%). The experiments showed our correction algorithm is promising and worth to be implemented for other stemmers. A next step would be to compare this root extractor with other available stemmers. As for the rule-based approach, the main limitations were in the rather limited number of patterns used, so it can be improved by including more patterns in the infix remover, and extracting two-letter geminated roots (as a first step). In general, the performance of the linguistic approach or our proposed correction algorithm can be increased by adding further rules and restrictions.

Acknowledgment

The first author would like to thank Petra University, Amman, Jordan for partially financing her PhD study.

References

[1] H.K Al-Ameed. A proposed new model using a light stemmer for increasing the success of search in Arabic terms. PhD Thesis, Bradford, UK: University of Bradford. 2006.

[2] I.A. Al-Sughaiyer and I.A Al-Kharashi. "Arabic morphology analysis techniques: A comprehensive survey". J. of the American Society for Information Science & Technology JASIST. Feb. 2004, 55(3), pp. 189 – 213.

[3] Imam Mohammed Ibn Abi Baker Ar-Rhazi. Mukhtar us-Sihah. Beirut: Librairie du Liban Publishers. 1986. (in Arabic).

[4] H. Bayyomee, Kh. Kolfat, and A. Al-Shafe'e. Lexicon for Arabic verbs morphology. Cairo: Dar Ilias Modern Publishing Comp. 1989. (in Arabic).

[5] K.R. Beesley. "Finite-State morphological analysis and generation of Arabic at Xerox Research: status and plans in 2001". In: ARABIC NLP Workshop: Status and Prospects ACL-EACL2001, Toulouse, France 6 July 2001, pp. 1 – 8.

[6] A. Chen and F. Gey. "Building an Arabic stemmer for Information Retrieval". In: NIST Special Publication: The Eleventh Text Retrieval Conf. TREC 2002, edited by E.M. Voorhees and D.K. Harman, Eds., Gaithersburg, Maryland, NIST: USA, 19-22 Nov. 2002.

[7] K. Darwish. "Al-stem: A light Arabic stemmer". As part of Dissertation Work Probabilistic Methods for Searching OCR-Degraded Arabic Text, University of Maryland, College Park, 2002.

[8] R. M. Duwairi, M. N. Al-Refai and N. Khasawneh. "Feature reduction techniques for Arabic TC". Journal of the American Society for Information Science and Technology, 2009, 60(11), pp. 2347 – 2352. ASIS&T.

[9] A. El-Dahdah. A Dictionary of Arabic Grammar in Charts and Tables. Beirut: Librairie du Liban Publishers. 2008. Revised by: Dr. GM Abdul-Massih (in Arabic).

[10] T. A. El-Sadany and M. A. Hashish. "An Arabic morphological system". IBM Systems Journal, 28(4), 1989, pp. 600-612.

[11] J.A. Haywood and H.M. Nahmad. A new Arabic grammar of written language. London: Lund Humphries Publishers. 1998.

[12] G. Kanaan, R. Al-Shalabi, S. Ghwanmeh and H. Al-Ma'adeed. "A comparison of TC techniques applied to Arabic text". Journal of the American Society for Information Science and Technology, 2009, 60(9), pp. 1836 – 1844. ASIS&T.

[13] L.S. Larkey, L. Ballesteros, M.E. Connell. "Improving stemming for Arabic Information Retrieval: Light stemming and co-occurrence analysis". In: SIGIR'02, Aug. 2002, Tampere, Finland, ACM. pp. 275 – 282.

[14] Leah S. Larkey, Lisa Ballesteros, and Margaret E. Connell. "Light stemming for Arabic Information retrieval". In Abdelhadi Soudi, Antal van den Bosch and Gunter Neumann, Eds., Arabic computational morphology knowledge-based and empirical methods, text, speech and language technology series, 2007, Vol. 38, Part IV, pp. 221 – 243, The Netherlands: Springer.

[15] F. Sebastiani, and Nazionale Delle Ricerche Consiglio. "Machine learning in automated Text Categorization". ACM Computing Survey, 34(1), 2002, pp. 1 – 47.

[16] M. M. Syiam, Z. T. Fayed and M. B. Habib. "An intelligent system for Arabic TC". IJICIS, 2006, 6(1), pp. 1 – 19. World Sci. Publ. Co.