Building a Shallow Arabic Morphological Analyzer in One Day

Kareem Darwish
Electrical and Computer Engineering Dept.,
University of Maryland, College Park
College Park, MD 20742
kareem@glue.umd.edu

Abstract:

The paper presents a rapid method of developing a shallow Arabic morphological analyzer. The analyzer will only be concerned with generating the possible roots of any given Arabic word. The analyzer is based on automatically derived rules and statistics. For evaluation, the analyzer is compared to a commercially available Arabic Morphological Analyzer.

1 Introduction

Due to the morphological complexity of the Arabic language, Arabic morphology has become an integral part of many Arabic Information Retrieval (IR) systems. Some of the Arabic IR systems that use morphology include Swift [1] and Sakhr's electronic publishing software [2]. Some Arabic IR studies have shown that the use of Arabic roots as indexing terms substantially improves the retrieval effectiveness over the use of stems, which in turn improve retrieval effectiveness over words [3] [4] [5].
Arabic words are divided into three types: noun, verb, and particle [6]. Nouns and verbs are derived from a closed set of around 10,000 roots [7]. The roots are commonly three or four letters and are rarely five letters. Arabic nouns and verbs are derived from roots by applying templates to the roots to generate stems and then introducing prefixes and suffixes. Figure 1 shows some templates for 3 letter roots. Figure 2 shows some of the possible prefixes and suffixes and their perspective meaning. The number of unique Arabic words (or surface forms) is estimated to be 6×10^{10} words [8]. Figure 3 shows some of the words that maybe generated from the root كتب "ktb". (Refer to figure I in the Appendix for the mapping between the Arabic letters and their Latin representations).

Further, a word may be derived from several different roots. For example the word إيمان "AymAn" can be derived from five different roots. Figure 4 shows possible roots for the word إيمان "AymAn" and the meaning of the word based on each. For the purposes of this paper, a word is any Arabic surface form, a stem is a word without any prefixes or suffixes, and a root is a linguistic unit of meaning, which has no prefix, suffix, or infix. However, often irregular roots, which contain double or weak letters, lead to stems and words that have letters from the root that are deleted or replaced.

فعل CCC	كتب "ktb" (wrote)	فعال CCAC	كتاب "ktAb" (book)	فاعل CACC	كاتب "kAtb" (writer)
مفعول mCCwC	مكتوب "mktwb" (something written)	فعاعيل CCA$\underline{\text{C}}$yC (middle letter doubled)	كتاتيب "ktAtyb" (Qur'an schools)	فعول CCwC	كتوب "ktwb" (skilled writer)

Figure 1: Some templates to generate stems from roots with an examples from the root (كتب "ktb")

Examples of prefixes					
و "w"	and	فـ "f"	Then	الـ "Al"	the
كـ "k"	like	لـ "l"	To	والـ "wAl"	and the
Examples of suffixes					
ـه "h"	his	ـهم "hm"	Their	ـها "hA"	her
ك "k"	your (singular)	كم "km"	your (plural)	ـي "y"	my

Figure 2: some examples of prefixes and suffixes and their meanings

Building a large-scale morphological analyzers is typically a laborious and time-consuming task. For example, MORPHO3 was developed by RDI in 3 man/years [8]. However, in this paper, we will present a quick method for performing shallow morphological analysis for use in information retrieval, which entails finding the roots of words, in one day. The method is based on collecting statistics from word-root pairs:

1. to build morphological rules for deriving roots from words,
2. to construct a list of prefixes and suffixes, and
3. to estimate the probability that a rule will be used or a prefix or suffix will be seen.

This analyzer, called Sebawai, is possibly the first cross-platform freely-distributable analyzer for Arabic.

Section 2 will provide background on some of the published research in Arabic Morphology. Section 3 will provide a description of the shallow morphological analyzer. Section 4 evaluates the analyzer and will address some of the shortcomings of the system.

كتب “ktb”	He wrote	يكتب “yktb”	He is writin g	أكتب “Aktb”	I write
كاتب “kAtb”	Writer	كتاب “ktAb”	Book	كتابه “ktAbh ”	His book
وكتابه “wktAb h”	And his book	كتابهم “ktAbhm”	Their book	كتب “ktb”	books

Figure 3: some of the words that can derived from the root form كتب “ktb”

Root	Meaning	pronunciation
أمن “Amn”	peace	إِيمَان——Eman
أيم “Aym”	two poor people	أَيِّمَان——Ayyiman
مأن “mAn”	will he give support	أَيَمْأَنُ——Ayama’nu
يمن “ymn”	covenant	أَيْمَان——Ayman
يمأ “ymA”	will they (feminine) point to	أَيَمْأَنَّ——Ayama’na

Figure 4: the possible roots for the word ايمان “AymAn” along with the meaning and pronunciation of the word based on the different roots.

2 Background

Significant work has been done in the area of the Arabic morphological analysis. The three main approaches to the problem are:

1. The Symbolic Approach: In this approach, morphotactic (rules governing the combination of morphemes, which are meaning bearing units in the language) and orthographic (spelling rules) rules are programmed into a finite state transducer (FST). Koskenniemi proposed a two-level system for language morphology, which led to Antworth’s two-level morphology system PC-KIMMO [9] [19]. Later, Beesley and Buckwalter developed an Arabic morphology system, ALPNET, that uses a slightly enhanced implementation of PC-KIMMO [10]. Currently, ALPNET is owned by Xerox and uses Xerox Finite-State Morphology tools [11]. However, this approach was criticized by Ahmed [8] for requiring excessive manual processing to state rules in an FST and for the ability only to analyze words that appear in Arabic dictionaries. Kiraz summarized many variations of the FST approach [12]. Much information on two-level morphology and PC-Kimmo is available in the PC-KIMMO user’s guide [20].
2. The Statistical Approach: Goldsmith proposed an unsupervised learning automatic morphology tool called AutoMorphology [14]. This system is advantageous because it learns prefixes, suffixes, and patterns from a corpus or word-list in the target language without any need for human intervention. However, such a system would not be effective in Arabic morphology, because it does not address the issues of infixation, and would not detect uncommon prefixes and suffixes.
3. The Hybrid Approach: This approach uses rules in conjunction with statistics. This approach employs a list of prefixes, a list of suffixes, and templates to transform from a stem to a root. Possible prefix-suffix-template combinations are constructed for a word to derive the possible roots. RDI’s system called MORPHO3 utilizes such this model [8]. Although such systems achieve broader morphological coverage of the Arabic language, manual derivation of rules is laborious, time-consuming and requires a good knowledge of Arabic orthographic and morphotactic rules. In fact, MORPHO3 was built in 3 man/years [8]. Large-scale morphological analyzers provide more information than just the root of a word. They may provide information such as the meaning of prefixes and suffixes and may perform root disambiguation [8] [10] [11]. However, this paper is concerned with

morphological analysis for the purpose of IR. Arabic IR is enhanced when the roots are used in indexing and searching [3] [4] [5].

3 System Description

Sebawai, the system discussed here, is similar to the hybrid approach used by RDI's MORPHO3 [8]. However, this system does not require manually constructed lists of rules and affixes. Instead, the system replaces the manual processing with automatic processing.

The system has two main modules. The first utilizes a list of Arabic word-root pairs (1) to derive a list of prefixes and suffixes, (2) to construct stem templates, and (3) to compute the likelihood that a prefix, a suffix, or a template would appear. The second accepts Arabic words as input, attempts to construct possible prefix-suffix-temple combinations, and outputs a ranked list of possible roots.

3.1 Getting a list of Word-Root Pairs

The list of word-root pairs may be constructed either manually, using a dictionary, or by using a pre-existing morphological analyzer such as ALPNET or MORPHO3 [8] [10].

1. Manual construction of word-root pair list: Building the list of several thousand pairs manually is time consuming, but feasible. Assuming that a person who knows Arabic can generate a root for a word every 5 seconds, the manual process would require about 14 hours of work to produce 10,000 word-root pairs.
2. Automatic construction of a list using dictionary parsing: Extracting word-root pairs from an electronic dictionary is a feasible process. Since Arabic words are looked up in a dictionary using their root form, an electronic dictionary such as Lisan Al-Arab may be parsed to generate the desired list. However, some care should be given to throw away dictionary examples and words unrelated to the root.
3. Automatic construction using a pre-existing morphological analyzer: This process is simple, but requires the availability of an analyzer.

For the purposes of this paper, the third method was used to construct the list. Two lists of Arabic words were fed to ALPNET (which was the only Arabic morphological analyzer available to the author) and then the output was parsed to generate the word-root pairs. One list was extracted from a corpus of traditional Arabic text, called Zad, owned by Al-Areeb Electronic Publishers [15]. The list contains 9,606 words that ALPNET was able to analyze successfully. The original list was larger, but the words that ALPNET was unable to analyze were excluded. The other list was extracted from the LDC Arabic collection (LDC2001T55) containing AFP news-wire stories [16]. This list contains 560,000 words. Of the 560,000 words, ALPNET was able to analyze 270,000 words successfully. The rest of the words (about 290,000) were used for evaluating Sebawai.

3.2 Training

As stated above, this module takes a word-root pair as input. By comparing the word to the root, the system determines the prefix, suffix, and stem template. For example, given the pair (وكتابهم "wktAbhm", كتب "ktb"), the system generates و "w" as the prefix, هم "hm" as the suffix, and فعال "CCAC" as the stem template (C's represent the letters in the root). The system increases the number of occurrences of the prefix و "w", the suffix هم "hm", and the template "CCAC" by one. The system takes into account the cases where there are no prefixes or suffixes and denotes either of them with the symbol "#".

Word	w	k	t	A	b	hm
Root		k	t		b	
Parts	Prefix (و)	Stem template (فعال --CCAC)				Suffix (هم)

Figure 5: The decomposition of the word وكتابهم "wktAbhm" with root كتاب "ktAb"

After that, the lists of prefixes, suffixes, and templates are read through to assign probabilities to items on the lists by dividing the occurrence of each item in each list by the total number of words. The probabilities being calculated are given for character strings S1 and S2 and template T as:

P(S1 begins a word, S1 is a prefix)
P(S2 ends a word, S2 is a suffix)
P(T is a template)

Another potential way of calculating the probabilities of prefixes and suffixes is to use the conditional probabilities that the item appears in the word and is actually a prefix or suffix. For example, if و "w" appeared as the first letter in the word 100 times, 70 times of which it was actually a prefix, then the probability would be .70. In other words, the probabilities being calculated are given for character strings S1 and S2 as:

P(S1 is a prefix | S1 begins a word)

P(S2 is a suffix | S2 ends a word)

Notice that Sebawai's stems are slightly different from standard stems. Standard stem templates may have letters added in the middle and in the beginning. For example the template مفعول "mCCwC" has م "m" placed before the root and و "w" placed in the middle. Both م "m" and و "w" are a part of the stem template. However, the training module has no prior knowledge of standard stem templates. Therefore, for the template مفعول "mCCwC", م "m" is simply treated as a part of the prefix list and the extracted template is فعول "CCwC".

3.3 Root Detection

The detect-root module accepts an Arabic word and attempts to generate prefix-suffix-template combinations. The combinations are produced by progressively removing prefixes and suffixes and then trying matching all the produced stems to a template. For example, for the Arabic word ايمان "AymAn" the possible prefixes are "#", ا "A", and اي "Ay", and the possible suffixes are "#", ن "n", and ان "An".

The resulting feasible stems are:

Stem	Prefix	Template	Suffix	Root
ايمان "AymAn"	"#"	فيعال "CyCAC"	"#"	امن "Amn"
يمان "ymAn"	ا "A"	فعال "CCAC"	"#"	يمن "ymn"
مان "mAn"	اي "Ay"	فعل "CCC"	"#"	مان "mAn"
ايم "Aym"	"#"	فعل "CCC"	ان "An"	ايم "Aym"
يما "ymA"	ا "A"	فعل "CCC"	ن "n"	يما "ymA"

The ones that the system deemed as not feasible are ايما "AymA" and يم "ym". Although ايما "AymA" is not feasible, يم "ym" is actually feasible (comes from the root يمم "ymm"), but the system did not know how to deal with it. The paper will address this problem in the next sub-section. The possible roots are ordered according to the product of the probability that a prefix S1 would be observed, the probability that a suffix S2 would be observed, and the probability that a template T would be used.

P(root) =

P(S1 begins a word, S1 is a prefix)

* P(S2 ends a word, S2 is a suffix)

* P(T is a template)

The probabilities of stems, suffixes, and templates are assumed to be independent. The independence assumption is made to simplify the ranking, but is not necessarily a correct assumption because certain prefix-suffix combinations are not allowed. Using the system requires some smoothing which will be discussed in the next subsection. The generated roots are compared to a list of 10,000 roots extracted automatically from an electronic copy of Lisan al-Arab to verify their existence in the language [7].

3.4 Missed or Erroneous Roots

As seen above, the system deemed the stem يم "ym" not feasible, while in actuality the stem maps to the root يمم "ymm". Other cases where the system failed were when the root had weak letters. Weak letters are ا "A", ي "y", and و "w". The weak letters are frequently substituted for each other in stem form or dropped all together. For example, the word قال "qAl" has the root قول "qwl" or قيل "qyl" which would make the word mean 'he said' or 'he napped' respectively. Also, the word ف "f" has the root وفي "wfy" where the letters و "w" and ي "y" are missing. To compensate for these problems, two letter stems were corrected by introducing new stems that are generated by doubling the last letter (to produce يمم "ymm" from يم "ym") and by adding weak letters before or after the stem. As for stems with a weak middle letter, new stems are introduced by substituting the middle letter with the other weak letters. For example, for قال "qAl", the system would introduce the stems قول "qwl" and قيل "qyl". This process over-generates potential roots. For example, from the three potential roots قال "qAl", قول "qwl", and قيل "qyl", قال "qAl" is not a valid root and is thus removed (by comparing to the list of valid roots). To account for the changes, the following probabilities were calculated: (a) the probability that a weak letter *w*1 would be transformed to another weak letter *w*2, (b) the probability that a two letter word would have a root with the second letter doubled (such as يمم "ymm"), and (c) the probability that a two letter word was derived from a root by dropping an initial or trailing weak letter. The new probability of the root becomes:

P(root) =
P(S1 begins a word, S1 is a prefix)
* P(S2 ends a word, S2 is a suffix)
* P(T is a template)
* P(letter substitution or letter addition)

As for smoothing the prefix and suffix probabilities, Witten-Bell discounting was used [17]. The smoothing is necessary because many prefixes and suffixes were erroneously produced. This is a result of word-root pair errors. Using this smoothing strategy, if a prefix or a suffix is observed only once, then it is removed from the respective list. As for the list of templates, it was reviewed by an Arabic speaker (the author of the paper) to insure the correctness of the templates. The Arabic examiner was aided by example words the system provided for each template. If a template was deemed not correct, it was removed from the list.

3.5 Particles

To account for particles, a list of Arabic particles was constructed with aid of *An-Nahw Ash-Shamil* (an Arabic grammar book) [6]. If the system matched a potential stem to one of the words on the particle list, the system would indicate that the word is a particle. Note that particles are allowed to have suffixes and prefixes. A complete list of the particles used by Sebawai is available upon request.

3.6 Letter Normalizations

The system employs a letter normalization strategy in order to account for spelling variations and to ease in the deduction of roots from words. The first normalization deals with the letters ي "y" and ى "Y" (alef maqsoura). Both are normalized to ي "y". The reason behind this normalization is that there is no one convention for spelling ي "y" or ى "Y" when either appears at the end of a word (Note that ى "Y" only appears at the end of a word). In the Othmani script of the Holy Qur'an for example, any ي "y" is written as ى "Y" when it appears at the end of a word [18]. The second normalization is that of "ء" (hamza), "آ" (alef maad), "أ" (alef with hamza on top), "ؤ" (hamza on w), "إ" (alef with hamza on the bottom), and "ئ" (hamza on ya). The reason for this normalization is that all forms of hamza are represented in dictionaries as one in root form namely "ء" or "أ", depending on the dictionary, and people often misspell different forms of alef. All are normalized to the symbol ا "A".

4 Evaluation and Discussion

To evaluate Sebawai, it was compared to ALPNET. A random set of a 100 word-root pairs produced by ALPNET was manually examined to verify their correctness and consequently verify the correctness of ALPNET. ALPNET produces some possible roots for each given word in unranked order, but all pairs were correct.

Three experiments were preformed. In the first and second experiments, Sebawai is trained on a large list and a small list of word-root pairs respectively. After the training, a list of words is fed into Sebawai and ALPNET for analysis. The correctness of analysis and coverage of both systems are compared. In the third experiment, a document collection is indexed using roots produced by both systems. Retrieval effectiveness of indexing using roots produced from each system is examined.

4.1 Using a Large Training Set

A list of 270K words was used for training the system and a list of 9,606 Arabic words was used for evaluation. Of the small test set, ALPNET analyzed all the words, while Sebawai analyzed 9,497 and failed on 112. For the generated roots, three different automatic evaluations were done:

<u>First (Auto-Eval-1)</u>: The top generated root is compared to the roots generated by ALPNET. If the root is on the list, it is considered correct. Using this method, 8,206 roots were considered correct.

<u>Second (Auto-Eval-2)</u>: The top two generated roots from Sebawai were compared to the list of roots that were generated by ALPNET. If either root appeared in the list then the morphological analysis was considered correct. Using this evaluation method, 8,861 roots were considered correct.

<u>Third (Auto-Eval-*n*)</u>: All the generated roots are compared to the ones generated by ALPNET. If any match is found, the analysis is considered correct. Using this method, 9,136 roots were considered correct.

However, this automatic evaluation has two flaws:

1. The number of Arabic roots in ALPNET's inventory are only 4,600 roots while the number

of roots used by Sebawai are more than 10,000. This could result in a correct roots being missed by ALPNET.
2. ALPNET often under-analyzes. For example the word في "fy" could be the particle في "fy" or could be a stem with the root فيي "fyy". ALPNET only generates the particle في "fy", but not the other root فيي "fyy". This could lead to false negatives.
Therefore manual examination of reject roots was necessary. However, due to the large number of rejected roots, 100 rejected roots from the evaluation Auto-Eval-1 and Auto-Eval-2 were selected at random for examination to estimate the shortfall of the automatic evaluation. Of the 100 rejected roots:

Evaluation Method	Correct	Incorrect
Auto-Eval-1	46	54
Auto-Eval-2	38	62

Results summary:

Eval. method	No. of words	Failures	No. or roots deemed (correct /incorrect)	No. of roots estimated to be correct (manual evaluation)
Auto-Eval-1	9,606	112 (1.17%)	8,206 / 1,291 (86.4 / 13.6%)	8,800 (92.7%)
Auto-Eval-2	9,606	112 (1.17%)	8,861 / 636 (93.3 / 6.7%)	9,136 (96.1%)
Auto-Eval-*n*	9,606	112 (1.17%)	9,136 / 360 (96.2 / 3.8%)	-

Another list of 292,216 words that ALPNET was unable to recognize were fed to Sebawai. Sebawai analyzed 128,169 words (43.9%), and failed otherwise. To verify the correctness of the system, 100 words were taken at random from the list for manual examination. Of the 100, 47 were actually analyzed correctly. Many of the failures were named-Entities. Extrapolating from the results of the manual examination, Sebawai would successfully recognize an estimated 60,000 words (20% of the original list).

Results summary:

Number of words	Number of roots detected	An estimate of the correctly detected roots
292,216	128,169 (43.9%)	60,000 (20%)

The failure of ALPNET and the low accuracy of Sebawai warrant further investigation. A quick review of the list shows a high frequency of named entities, misspelled words, and obscure words.

4.2 Using a Small Training Set

The 9,606 words list was used for training and the 270K words list was used for evaluation. The same automatic evaluation method mentioned above was used. Of the 270,468 words, the system was unable to analyze 84,421, and analyzed 186,047. Similar to the experiment with the large training set, three automatic evaluations were used: Auto-Eval-1, Auto-Eval-2, and Auto-Eval-*n*. For Auto-Eval-1 and Auto-Eval-2, 100 of the rejected roots were manually examined to verify correctness. Of the 100 roots examined:

Eval. Method	Correct	Incorrect
Auto-Eval-1	30	70
Auto-Eval-2	45	55

Results summary:

Eval. Method	No. of roots	Words not analyzed	No. or roots deemed (correct /incorrect)	No. of roots estimated to be correct (manual evaluation)
Auto-Eval-1	270,468	84,421 (31.21%)	130,990 / 55,057 (70.4 / 29.6%)	147,507 (79.3%)
Auto-Eval-2	270,468	84,421 (31.21%)	149,906 / 36,141 (80.6 / 19.4%)	166,169 (89.3%)
Auto-Eval-*n*	270,468	84,421 (31.21%)	159, 02 / 27,020 (85.5 / 14.5%)	-

Also, the 292,216 words that ALPNET was unable to recognize were fed to Sebawai. Sebawai analyzed 92,929 words (31.8%). To verify the correctness of the system, 100 words were taken at random from the list for manual examination. Of the 100, 55 were actually analyzed correctly. Extrapolating from the results of the manual examination, Sebawai would successfully recognize an estimated 60,000 words (20% of the original list).
Results summary:

Number of words	Number of roots detected	An estimate of the correctly detected roots
292,216	92,929 (31.8%)	51,000 (17%)

4.3 Retrieval Effectiveness

In the third part of the evaluation, the Zad document collection, which contains 4,000 documents, was used for retrieval evaluation. Associated with the collection was a set of 25 queries and their relevance judgments. Sebawai

was trained using the list of 270K words. InQuery was the retrieval engine used.
In the evaluation, 4 different runs were performed. In the first two, the collection was indexed using one root and two roots produced by ALPNET. In the later two, the collection was indexed using the top root and the top two roots generated by Sebawai. Mean average precision was used as the figure of merit in comparing the runs. For statistical significance, a paired two-tailed *t*-test was used. Statistical significance was concluded if the *p*-value of *t*-test was lower than .05.
Results summary:

Index term	Mean Avg. Precision
ALPNET – 1 root	0.34
ALPNET – 2 root	0.36
Sebawai – 1 root	0.45
Sebawai – 1 root	0.29

Using Sebawai's guess of the most likely root resulted in a higher mean average precision than when using one root produced by ALPNET (Note that ALPNET randomly ordered the possible roots). Further, using two roots from ALPNET slightly improved mean average precision, but the improvement was not statistically significant. Using the top two roots from Sebawai significantly harmed retrieval. A likely reason for the fall in mean average precision when the second root was introduced is that the second root amounted to noise.

4.4 Success and Limitations

The evaluation method clearly shows the effectiveness of Sebawai. In fact, Sebawai significantly outperformed ALPNET in retrieval experiments. The analyzer is often able to detect roots that were missed by a commercially available system. Also, due to the fact that rule are derived automatically, Sebawai was developed very rapidly. It was built in less than 12 hours using about 200 lines of Perl code [21]. Further, the analyzer is able to derive the roots of 40,000 words per minute on a Pentium class machine with 256 MB of RAM running Linux. Also, Sebawai is twice as fast as ALPNET on the same machine. Rewriting Sebawai in a compiled language such as C is likely to improve the analysis speed.
Furthermore, the method used to develop this Arabic morphological analyzer can potentially be used to rapidly develop morphological analyzers for other languages. Some languages exhibit morphological properties similar to those of Arabic such as Hebrew [12].
However, the system is restricted in the following aspects:
1. Since it limits the choice of roots to a fixed set, it does not stem words transliterated from other languages such as transliterated named entities. For example, the English word Britain is transliterated as بريطانيا "bryTAnyA". From بريطانيا "bryTAnyA", some the words that maybe generated are: بريطاني "bryTAny" (British), البريطاني "AlbryTAny" (the British), and البريطانيين "AlbryTAnyyn" (Englishmen).
2. Some words in Arabic are 1 letter long, but have 3 letter roots. For example, the word ق "q" means "protect (in the form of command)". Since they are very rare, they may not appear in the training set.
3. Some individual words in Arabic constitute complete sentences. For example, the word أنلزمكموها "AnlzmkmwhA" means "will we forcefully bind you to it?" These also are rare and may not appear in a training set.
4. The analyzer lacks the ability to decipher which prefix-suffix combinations are legal. Although deciphering the legal combinations is feasible using statistics, the process would potentially require a huge number of examples to insure that the system would not disallow legal combinations.

5 Conclusion and Future Work

The paper presented a way to rapidly develop a shallow Arabic morphological analyzer. The analyzer is based on automatically derived rules and statistics. The analyzer is cross-platform and freely-distributable. Although some knowledge of the Arabic language was required to verify the correctness of derived rules for example, the amount of time required to build the rules is reduced to hours rather than days or weeks. Some the possible future work includes:
1. Integrating stemming with the analyzer to handle words the analyzer failed on.
2. Attempting to develop morphological analyzers for other language using the same method describe in the paper.
3. Collecting statistics on legal prefix-suffix combinations to further improve the analyzer.
4. Comparing the retrieval effectiveness when indexing is done using this analyzer compared to

another commercially available analyzer such as ALPNET or MORPHO3.
5. Examining the words for which ALPNET was unable to produce roots. This would give insight into the strength and weakness of ALPNET.

Acknowledgements:

I would like to thank Douglas W. Oard, Nizar Habash, and Amy Weinberg for all their valuable feedback.

References:

1. RDI (Research and Development International), Cairo, Egypt. www.rdi-eg.com
2. Sakhr, Cairo, Egypt. www.sakhr.com
3. Al-Kharashi, Ibrahim and Martha Evens, "*Comparing Words, Stems, and Roots as Index Terms in an Arabic Information Retrieval.*" JASIS. 45 (8): 548-560, 1994.
4. Abu-Salem, Hani, Mahmoud Al-Omari, and Martha Evens, "*Stemming Methodologies Over Individual Query Words for Arabic Information Retrieval.*" JASIS. 50 (6): 524-529, 1999.
5. Hmeidi, Ismail, Ghassan Kanaan, and Martha Evens, "*Design and Implementation of Automatic Indexing for Information Retrieval with Arabic Documents*." JASIS. 48 (10): 867-881, 1997.
6. Abdul-Al-Aal, Abdul-Monem, *An-Nahw Ashamil*. Maktabat Annahda Al-Masriya, Cairo, Egypt, 1987.
7. Ibn Manzour, *Lisan Al-Arab.* www.muhaddith.org.
8. Ahmed, Mohamed Attia, "*A Large-Scale Computational Processor of the Arabic Morphology, and Applications*." A Master's Thesis, Faculty of Engineering, Cairo University, Cairo, Egypt, 2000.
9. Koskenniemi, Kimmo, "*Two Level Morphology: A General Computational Model for Word-form Recognition and Production.*" Publication No. 11, Dept. of General Linguistics, University of Helsinki, Helsinki, 1983.
10. Beesley, Kenneth, Tim Buckwalter, and Stuart Newton, "*Two-Level Finite-State Analysis of Arabic Morphology.*" Proceedings of the Seminar on Bilingual Computing in Arabic and English, Cambridge, England, 1989.
11. Beesley, Kenneth, "*Arabic Finite-State Morphological Analysis and Generation.*" COLING-96, 1996.
12. Kiraz, George, "*Arabic Computation Morphology in the West.*" Proceedings of the 6th International Conference and Exhibition on Multi-lingual Computing, Cambridge, 1998.
13. The Temple Project. http://crl.nmsu.edu/tide/
14. Goldsmith, John, "*Unsupervised Learning of the Morphology of a Natural Language*." http://humanities.uchicago.edu/faculty/goldsmith/, 2000.
15. Al-Areeb Electronic Publishers, LLC. 16013 Malcolm Dr., Laurel, MD 20707, USA
16. Cole, Andy, David Graff, and Kevin Walker, "*Arabic Newswire Part 1 Corpus (1-58563-190-6)*", Linguistic Data Consortium (LDC). www.ldc.upenn.edu
17. Jurafsky, Daniel and James Martin, "*Speech and Language Processing*." Prentice Hall, New Jersey, 2000.
18. The Holy Qur'an.
19. Antworth, Evan L, "*PC-KIMMO: a two-level processor for morphological analysis.*" Occasional Publications in Academic Computing No. 16. Dallas, TX: Summer Institute of Linguistics, 1990.
20. Antworth, Evan L, "*PC-KIMMO User's Guide.*" www.sil.org/pckimmo/v2/doc/guide.html, 1995
21. Perl. www.perl.com

Appendix:

figure 1: Letter mappings

A	ا	A	ء	A	آ	A	ؤ	A	أ
A	إ	A	ئ	b	ب	t	ت	p	ة
v	ث	j	ج	H	ح	x	خ	d	د
O	ذ	r	ر	z	ز	S	س	P	ش
S	ص	D	ض	T	ط	Z	ظ	E	ع
j	غ	f	ف	Q	ق	k	ك	l	ل
M	م	n	ن	h	ه	W	و	y	ي