A Morphologically Sensitive Clustering Algorithm for Identifying Arabic Roots

Anne N. DE ROECK
Department of Computer Science
University of Essex
Colchester, CO4 3SQ, U.K.
deroe@essex.ac.uk

Waleed AL-FARES
Computer Science Department
College of Business Studies,
Hawaly, Kuwait
al-fareswaleed@usa.net

Abstract

We present a clustering algorithm for Arabic words sharing the same root. Root based clusters can substitute dictionaries in indexing for IR. Modifying Adamson and Boreham (1974), our Two-stage algorithm applies light stemming before calculating word pair similarity coefficients using techniques sensitive to Arabic morphology. Tests show a successful treatment of infixes and accurate clustering to up to 94.06% for unedited Arabic text samples, without the use of dictionaries.

Introduction

Canonisation of words for indexing is an important and difficult problem for Arabic IR. Arabic is a highly inflectional language with 85% of words derived from tri-lateral roots (Al-Fedaghi and Al-Anzi 1989). Stems are derived from roots through the application of a set of fixed patterns. Addition of affixes to stems yields words. Words sharing a root are semantically related and root indexing is reported to outperform stem and word indexing on both recall and precision (Hmeidi et al 1997).

However, Arabic morphology is excruciatingly complex (the Appendix attempts a brief introduction), and root identification on a scale useful for IR remains problematic.

Research on Arabic IR tends to treat automatic indexing and stemming separately. Al-Shalabi and Evans (1998) and El-Sadany and Hashish (1989) developed stemming algorithms. Hmeidi et al (1997) developed an information retrieval system with an index, but does not explain the underlying stemming algorithm. In Al-Kharashi and Evans (1994), stemming is done manually and the IR index is built by manual insertion of roots, stems and words.

Typically, Arabic stemming algorithms operate by “trial and error”. Affixes are stripped away, and stems “undone”, according to patterns and rules, and with reference to dictionaries. Root candidates are checked against a root lexicon. If no match is found, affixes and patterns are re-adjusted and the new candidate is checked. The process is repeated until a root is found.

Morpho-syntactic parsers offer a possible alternative to stemming algorithms. Al-Shalabi and Evans (1994), and Ubu-Salem et al (1999) develop independent analysers. Some work builds on established formalisms such a DATR (Al-Najem 1998), or KIMMO. This latter strand produced extensive deep analyses. Kiraz (1994) extended the architecture with multi-level tape, to deal with the typical interruption of root letter sequences caused by broken plural and weak root letter change. Beesley (1996) describes the re-implementation of earlier work as a single finite state transducer between surface and lexical (root and tag) strings. This was refined (Beesley 1998) to the current on-line system capable of analysing over 70 million words.

So far, these approaches have limited scope for deployment in IR. Even if substantial, their morpho-syntactic coverage remains limited and processing efficiency implications are often unclear. In addition, modern written Arabic presents a unique range of orthographic problems. Short vowels are not normally written (but may be). Different regional spelling conventions may appear together in a single text and show interference with spelling errors. These systems, however, assume text to be in perfect (some even vowelised) form, forcing the need for editing prior to processing. Finally, the success of these algorithms depends critically on root, stem, pattern or affix dictionary quality, and no sizeable and reliable electronic dictionaries exist. Beesley (1998) is the exception with a reported 4930 roots encoded with associated patterns, and an additional affix and non-root stem lexicon[1]. Absence of large and reliable electronic lexical resources means dictionaries would have to be updated as new words appear in the text, creating a maintenance overhead. Overall, it remains uncertain whether these approaches can be deployed and scaled up cost-effectively to provide the coverage required for full scale IR on unsanitised text.

Our objective is to circumvent morpho-syntactic analysis of Arabic words, by using clustering as a technique for grouping words sharing a root. In practise, since Arabic words derived from the same root are semantically related, root based clusters can substitute root dictionaries for indexing in IR and furnish alternative search terms. Clustering works without dictionaries, and the approach removes dictionary overheads completely. Clusters can be implemented as a dimension of the index, growing dynamically with text, and without specific maintenance. They will accommodate effortlessly a mixture of regional spelling conventions and even some spelling errors.

1 Clustering and Arabic.

To our knowledge, there is no application of automatic root-based clustering to Arabic, using morphological similarity without dictionary. Clustering and stemming algorithms have mainly been developed for Western European languages, and typically rely on simple heuristic rules to strip affixes and conflate strings. For instance, Porter (1980) and Lovins (1968) confine stemming to suffix removal, yet yield acceptable results for English, where roots are relatively inert. Such approaches exploit the morphological frugality of some languages, but do not transfer to heavily inflected languages such as Arabic.

In contrast, Adamson and Boreham (1974) developed a technique to calculate a similarity co-efficient between words as a factor of the number of shared sub-strings. The approach (which we will call Adamson's algorithm for short) is a promising starting point for Arabic clustering because affix removal is not critical to gauging morphological relatedness.

In this paper, we explain the algorithm, apply it to raw modern Arabic text and evaluate the result. We explain our Two-stage algorithm, which extends the technique by (a) light stemming and (b) refinements sensitive to Arabic morphology. We show how the adaptation increased successful clustering of both the original and new evaluation data.

2 Data Description

We focus on IR, so experiments use modern, unedited Arabic text, with unmarked short vowels (Stalls and Knight 1998). In all we constructed five data sets. The first set is controlled, and was designed for testing on a broad spectrum of morphological variation. It contains selected roots with derived words chosen for their problematic structure, featuring infixes, root consonant changes and weak letters. It also includes superficially similar words belonging to different roots, and examples of *hamza* as a root consonant, an affix and a silent sign. Table 1 gives details.

Table 1: Cluster size for 1st data set

root	size	root	size
ktb *wrote*	49	HSL *obtained*	7
qwm *straightened*	38	s'aL *asked*	6
mr *passed*	26	HSd *cultivated*	5
wSL *linked*	11	shm *shared*	4
r'as *headed*	10		

Data sets two to four contain articles extracted from Al-Raya (1997), and the fifth from Al-Watan (2000), both newspapers from Qatar. Following Adamson, function words have been removed. The sets have domain bias with the second (575 words) and the fourth (232 words) drawn randomly from the economics and the third (750 words) from the sports section. The fifth (314 words) is a commentary on political history. Sets one to three were used to varying extents in refining our Two-stage algorithm. Sets four and five were used for evaluation only.

Electronically readable Arabic text has only recently become available on a useful scale, hence our experiments were run on short texts. On the other hand, the coverage of the data sets allows us to verify our experiments on demanding samples, and their size lets us verify correct clustering manually.

[1] Al-Fedaghi and Al-Anzi (1989) estimate there are around 10,000 independent roots.

3. Testing Adamson's Algorithm

3.1 The Algorithm

Adamson and Boreham (1974) developed a technique expressing relatedness of strings as a factor of shared sub-strings. The algorithm drags an *n*-sized window across two strings, with a 1 character overlap, and removes duplicates. The strings' similarity co-efficient (SC) is calculated by Dice's equation: SC (Dice) = 2*(number of shared unique n-grams)/(sum of unique n-grams in each string)

Table 2: Adamson's Algorithm Illustrated

String	**2-grams**	**Unique 2-grams**
phosphorus	*ph ho os sp ph ho or ru us*	*ph ho os sp or ru us (7)*
phosphate	*ph ho os sp ph ha at te*	*ph ho os sp ha at te (7)*
Shared unique 2-grams		*ph ho os sp (4)*
SC (Dice) = 2(4)/(7+7) = 0.57		

After the SC for all word pairs is known, the single link clustering algorithm is applied. A similarity (or dissimilarity) threshold is set. The SC of pairs is collected in a matrix. The threshold is applied to each pair's SC to yield clusters. A cluster absorbs a word as long as its SC to another cluster item exceeds the threshold (van Rijsbergen 1979). Similarity to a single item is sufficient. Cluster size is not pre-set.

3.2 Background Assumptions

This experiment tests Adamson's algorithm on Arabic data to assess its ability to cluster words sharing a root. Each of the data sets was clustered manually to provide an ideal benchmark. This task was executed by a native Arabic speaker with reference to dictionaries.

Since we are working with very small texts, we sought to remove the effects of sampling in the tests. To assess Adamson's algorithm's potential for clustering Arabic words, we preferred to compare instances of optimal performance. We varied the SC to yield, for each data set, the highest number of correct multi-word clusters.

Note that the higher the SC cut-off, the less likely that words will cluster together, and the more single word clusters will appear. This has the effect of growing the number of correct clusters because the proportion of correct single word clusters will increase. As a consequence, for our purposes, the number of correct multi-word clusters (and not just correct clusters) are an important indicator of success.

A correct multi-word cluster covers at least two words and is found in the manual benchmark. It contains all and only those words in the data set which share a root. Comparison with a manual benchmark inevitably introduces a subjective element. Also, our evaluation measure is the percentage of correct benchmark clusters retrieved. This is a "recall" type indicator. Together with the strict definition of correct cluster, it cannot measure cluster quality. Finer grained evaluation of cluster quality would be needed in an IR context.

However, our main concern is comparing algorithms. The current metrics aim for a conservative gauge of how Adamson's algorithm can yield more exact clusters from a full range of problematic data.

Table 3: Adamson's Algorithm Test Results

Data set	**Set 1**	**Set 2**	**Set 3**	**Set 4**	**Set 5**
Benchmark: **Total Manual Clusters(A)**	9	267	337	151	190
Multi-word (B)	9	130	164	50	63
Single word (C)	0	137	173	101	127
SC cut-off[2]	0.50	0.54	0.75	0.58-0.60	0.61-0.66
Test:(% of Benchmark) **Correct Clusters (% of A)**	11.11%	56.55%	60.83%	70.86%	74.21%
Multi-word (% of B)	11.11%	38.46%	21.95%	40%	34.92%
Single word (% of C)	0.0%	73.72%	97.69%	86.14%	93.70%

[2] Ranges rather than specific values are given where cut-offs between the lower and higher value do not alter cluster distribution.

Our interpretation of correct clustering is stringent and therefore conservative, adding to the significance of our results. Cluster quality will be reviewed informally.

3.3 Adamson's Arabic Test Results

Table 3 shows results for Adamson's algorithm. The figures for the first data set have to be suitably interpreted. The set deliberately did not include single word clusters.

The results suggest that the algorithm is very successful at identifying single word clusters but performs poorly on multi-word clusters. The high success rate for single word clusters is partly due to the high SC cut-off, set to yield as many correct multi-word clusters as possible.

In terms of quality, however, only a small proportion of multi-word clusters were found to contain infix derivations (11.11%, 4.76%, 0.0% 4.35% and 9.09% for each data set respectively), as opposed to other variations. In other words, strings sharing character sequences in middle position cluster together more successfully. Infix recognition is a weak point in this approach.

Whereas the algorithm is successful for English, it is no surprise that it should not perform equally well on Arabic. Arabic words tend to be short and the chance of words derived from different roots sharing a significant proportion of characters is high (eg K^{h}br (*news*) vs K^{h}bz (*bread*)). Dice's equation assumes the ability to identify an uninterrupted sequence of root consonants. The heavy use of infixes runs against this. Similarly, affixes cause interference (see 4.1.1).

4 The Two-Stage Algorithm.

The challenge of root based clustering for Arabic lies in designing an algorithm which will give relevance to root consonants only. Using Adamson's algorithm as a starting point, we devised a solution by introducing and testing a number of successive refinements based on the morphological knowledge and the first three data sets. The rationale motivating these refinements is given below.

4.1 Refinements

4.1.1 Affixes and light stemming:

The high incidence of affixes keeps accurate cluster formation low, because it increases the SC among words derived from different roots, and lowers the SC between derivations of the same root using different affixes, as illustrated in tables 4 and 5. Following Popovic and Willet (1992), we introduced stemming to minimise the effect of affixes. We found empirically that light stemming, removing a small number of obvious affixes, gave better results than heavy stemming aimed at full affix stripping. Heavy stemming brought the risk of root consonant loss (eg t'amyn (*insurance*) from root amn (*sheltered*): heavy stemming: t'am, light stemming: t'amn). Light stemming, on the other hand, does little more than reducing word size to 3 or 4 characters.

4.1.2 Weak letters, infixes and "cross":

Weak letters (*alif*, *waw*, *ya*) occur freely as root consonants as well as affixes. Under derivation, their form and location may change, or they may disappear. As infixes, they interfere with SC, causing failure to cluster (table 6). Their effects were reduced by a method we refer to as "cross". It adds a bi-gram combining the letters occurring before and after the weak letter.

Table 4: Inflected words from different roots: ?Lm (*learned*) and arb (*arabised*)

String	Unique 2-grams with affixes	Unique 2-grams without affixes
aL?aLmyh (the universal)	aL L? ?a Lm my yh (6)	?a Lm (2)
aL?rbyh (the Arabic)	aL L? ?r rb by yh (6)	?r rb (2)
SC (Dice)	2(3)/(6+6) = 0.50	2(0)/(2+2) = 0

Table 5: Inflected words from the same root: mrr (*passed*)

String	Unique 2-grams with affixes	Unique 2-grams without affixes
mstmr (continuous)	ms st tm mr (4)	mr (1)
mr (passed)	mr (1)	mr (1)
SC (Dice)	2(1)/(4+1) = 0.40	2(1)/(1+1) = 1.0

Table 6: Infix derivation from root wqf (stopped) - post light stemming

String	Unique 2-grams without cross	Unique di-grams with cross
qaf	qa af (2)	qa af qf (3)
wqf	wq qf (2)	wq qf (2)
SC (Dice)	2(0)/(2+2) = 0	2(1)/(2+3) = 0.4

4.1.3 Suspected affixes and differential weighting:

Our objective is to define an algorithm which gives suitable precedence to root consonants. Light stemming, however does not remove all affixes. Whereas fool proof affix detection is problematic due to the overlap between affix and root consonants, affixes belong to a closed class and it is possible to identify "suspect" letters which might be part of an affix.

Following Harman (1991) we explored the idea of assigning differential weights to sub-strings. Giving equal weight of 1 to all substrings equates the evidence contributed by all letters, whether they are root consonants or not. Suspected affixes, however, should not be allowed to affect the SC between words on a par with characters contributing stronger evidence. We conducted a series of experiments with differential weightings, and determined empirically that 0.25 weight for strings containing weak letters, and 0.50 for strings containing suspected non-weak letter affixes gave the best SC for the first three data sets.

4.1.4 Substring boundaries:

N-gram size can curtail the significance of word boundary letters (Robertson and Willet 1992). To give them opportunity to contribute fully to the SC, we introduced word boundary blanks (Harman 1991).

Also, the larger the n-gram, the greater its capacity to mask the shorter substring which can contain important evidence of similarity between word pairs (Adamson and Boreham 1974). Of equal importance is the size of the sliding overlap between successive n-grams (Adams 1991).

Table 7: Blank insertion with "cross"

String	Unique 2-grams (no)
qaf	*q qa af qf f* (5)
wqf	*w wq *q qf f* (5)
SC (Dice)	2(3)/(5+5) = 0.60

The problem is to find the best setting for n-gram and overlap size to suit the language. We sought to determine settings experimentally. Bi-grams with single character overlap and blank insertion (* in the examples) at word boundaries raised the SC for words sharing a root in our three data sets, and lowered the SC for words belonging to different roots.

4.1.5 SC formula:

Dice's equation boosts the importance of unique shared substrings between word pairs, by doubling their evidence. As we argued earlier, since Arabic words tend to be short, the relative impact of shared substrings will already be dramatic. We replaced the Dice metric with the Jaccard formula below to reduce this effect (see van Rijsbergen 1979). SC (Jac) = shared unique n-grams/(sum of unique n-grams in each string - shared unique n-grams)

4.2 The Two-stage Algorithm

The Two-stage algorithm is fully implemented. Words are first submitted to light stemming to remove obvious affixes. The second stage is based on Adamson's algorithm, modified as described above. From the original, we retained bi-grams with a one character overlap, but inserted word boundary blanks. Unique bi-grams are isolated and cross is implemented. Each bi-gram is assigned a weight (0.25 for bi-grams containing weak letters; 0.5 for bi-grams containing potential non-weak letter affixes; 1 for all other bi-grams). Jaccard's equation computes a SC for each pair of words. We retained the single-link clustering algorithm to ensure comparability.

4.3 Testing the Two-stage Algorithm

Table 8 shows the results of the Two-stage algorithm for our data sets. The maximally effective cut of point for all sets lies closer. Figures for the first set have to be treated with caution. The perfect clustering is explained by the text's perfect spelling and by the sample containing exactly those problematic phenomena on which we wanted to concentrate.

Table 8: Two-stage Algorithm Test Results

Data set	**Set 1**	**Set 2**	**Set 3**	**Set 4**	**Set 5**
Benchmark: **Total Manual Clusters (A)**	9	267	337	151	190
Multi-word (B)	9	130	164	50	63
Single word (C)	0	137	173	101	127
SC cut-off	0.42-0.66	0.54	0.54	0.53-0.54	0.62-0.66
Test: (% of Benchmark) **Correct Clusters (% of A)**	100%	88.05%	86.94%	94.04%	86.84%
Multi-word (% of B)	100%	85.39%	82.93%	94%	74.60%
Single word (% of C)	-	90.51%	90.75%	94.06%	92.91%

The algorithm deals with weak letter mutation, and infix appearance and disappearance in words sharing a root (eg the root qwm and its derived words, especially the role of *Hamza* as an infix in one of its variations). Even though the second and third data sets informed the modifications to a limited extent, their results show that the improvements stood up to free text. For the second data set, the Two-stage algorithm showed 31.5% improvement over Adamson's algorithm. Importantly, it discovered 84.13% of the multi-word clusters containing words with infixes, an improvement of 79.37%. The values for single word clustering are close and the modifications preserved the strength of Adamson's algorithm in keeping single word clusters from mixing, because we were able to maintain a high SC threshold.

On the third data set, the Two-stage algorithm showed an 26.11% overall improvement, with 84% successful multi-word clustering of words with infixes (compare 0% for Adamson). The largest cluster contained 14 words. 10 clusters counted as unsuccessful because they contained one superficially similar variation belonging to a different root (eg TwL (*lengthened*) and bTL (*to be abolished*)). If we allow this error margin, the success rate of multi-word clustering rises to 90%. Since our SC cut-off was significantly lower than in Adamson's base line experiment, we obtained weaker results for single word clustering.

The fourth and fifth data sets played no role in the development of our algorithm and were used for evaluation purposes only. The Two-stage algorithm showed an 23.18% overall improvement in set four. It successfully built all clusters containing words with infixes (100% - compare with 4.35% for Adamson's algorithm), an improvement of 95.65%. The two-stage algorithm again preserved the strength of Adamson at distinguishing single word clusters, in spite of a lower SC cut-off.

The results for the fifth data set are particularly important because the text was drawn from a different source and domain. Again, significant improvements in multi and single word clustering are visible, with a slightly higher SC cut-off. The algorithm performed markedly better at identifying multi-word clusters with infixes (72.72% - compare with 9.09% for Adamson).

The results suggest that the Two-stage algorithm preserves the strengths of Adamson and Boreham (1994), whilst adding a marked advantage in recognising infixes. The outcome of the evaluation on fourth and fifth data sets are very encouraging and though the samples are small, they give a strong indication that this kind of approach may transfer well to text from different domains on a larger scale.

5 Two-stage Algorithm Limitations

Weak letters can be root consonants, but our differential weighting technique prevents them from contributing strong evidence, whereas non-weak letters featuring in affixes, are allowed to contribute full weight. Modifying this arrangement would interfere with successful clustering (eg after light stemming: *t* is a root consonant in ntj (*produced*) and an infix in Ltqy (from root Lqy - *encountered*). These limitations are a result of light stemming.

Although the current results are promising, evaluation was hampered by the lack of a sizeable data set to verify whether our solution would scale up.

Conclusion

We have developed, successfully, an automatic classification algorithm for Arabic words which share the same root, based only on their morphological similarities. Our approach works on unsanitised text. Our experiments show that algorithms designed for relatively uninflected languages can be adapted for highly inflected languages, by using morphological knowledge.

We found that the Two-stage algorithm gave a significant improvement over Adamson’s algorithm for our data sets. It dealt successfully with infixes in multi-word clustering, an area where Adamson’s algorithm failed. It matched the strength of Adamson in identifying single word clusters, and sometimes did better. Weak letters and the overlap between root and affix consonants continue to cause interference. Nonetheless, the results are promising and suggest that the approach may scale up

Future work will concentrate on two issues. The light stemming algorithm and the differential weighting may be modified to improve the identification of affixes. The extent to which the algorithm can be scaled up must be tested on a large corpus.

Acknowledgements

Our thanks go to the Kuwait State's Public Authority for Applied Education and Training, for the supporting research studentship, and to two anonymous referees for detailed, interesting and constructive comments.

Appendix - Arabic in a Nutshell

The vast majority of Arabic words are derived from 3 (and a few 4) letter roots via a complex morphology. Roots give rise to stems by the application of a set of fixed patterns. Addition of affixes to stems yields words.

Table 9: Stem Patterns

Root		Pattern	Stem	
ktb	*wrote*	fa?L	katb	*writer*
		mf?wL	mktwb	*document*
qtL	*killed*	fa?L	qatL	*killer*
		mf?wL	mqtwL	*corpse*

Table 9 shows examples of stem derivation from 3-letter roots. Stem patterns are formulated as variations on the characters f?L (pronounced as *f'l* - ? is the symbol for *ayn*, a strong glottal stop), where each of the successive consonants matches a character in the bare root (for ktb, k matches f, t matches ? and b matches L). Stems follow the pattern as directed. As the examples show, each pattern has a specific effect on meaning. Several hundred patterns exist, but on average only about 18 are applicable to each root (Beesley 1998).

The language distinguishes between long and short vowels. Short vowels affect meaning, but are not normally written. However, patterns may involve short vowels, and the effects of some patterns are indistinguishable in written text. Readers must infer the intended meaning.

Affixes may be added to the word, either under derivation, or to mark grammatical function. For instance, walktab breaks down as w (*and*) + al (*the*) + ktab (*writers*, or *book*, depending on the voweling). Other affixes function as person, number, gender and tense markers, subject and direct object pronouns, articles, conjunctions and prepositions, though some of these may also occur as separate words (eg wal (*and the*)).

Arabic morphology presents some tricky NLP problems. Stem patterns “interdigitate” with root consonants, which is difficult to parse. Also, the long vowels a (*alif*), w (*waw*) and y (*ya*) can occur as root consonants, in which case they are considered to be weak letters, and the root a weak root. Under certain circumstances, weak letters may change shape (eg *waw* into *ya*) or disappear during derivation. Long vowels also occur as affixes, so identifying them as affix or root consonant is often problematic.

The language makes heavy use of infixes as well as prefixes and suffixes, all of which may be consonants or long vowels. Apart from breaking up root letter sequences (which tend to be short), infixes are easily confused with root consonants, whether weak or not. The problem for affix detection can be stated as follows: weak root consonants are easily confused with long vowel affixes; consonant affixes are easily confused with non-weak letter root consonants. Erroneus stripping of affixes will yield the wrong root.

Arabic plurals are difficult. The dual and some plurals are formed by suffixes, in which case they are called external plurals. The broken, or internal plural, however, changes the internal structure of the word according to a set of patterns. To illustrate the complexity, masculine plurals take a -wn or -yn suffix, as in mhnds (*engineer*), mhndswn. Female plurals add the -at suffix, or change word final -h to -at, as in mdrsh (*teacher*), mdrsat. Broken plurals affect root characters, as in mal (*fund* from root mwl), amwal, or wSL (*link* from root wSL), ‘aySaL. The examples are rife with long vowels (weak letters?). They illustrate the degree of interference between broken plural patterns and other ways of segmenting words.

Regional spelling conventions are common: eg. three versions of word initial *alif* occur. The most prominent orthographic problem is the behaviour of *hamza*, (**’**), a sign written over a carrier letter and sounding a lenis glottal stop (not to be confused with *ayn*). *Hamza* is not always pronounced. Like any other consonant, it can take a vowel, long or short. In word initial position it is always carried by *alif*, but may be written above or below, or omitted. Mid-word it is often carried by one of the long vowels, depending on rules whose complexity often gives rise to spelling errors. At the end of words, it may be carried or written independently. *Hamza* is used both as a root consonant and an affix, and is subject to the same problems as non-weak letter consonants, compounded by unpredictable orthography: identical words may have differently positioned *hamzas* and would be considered as different strings.

References

Adams, E. (1991) *A Study of Trigrams and their feasibility as Index Terms in a full text Information Retrieval System*. PhD Thesis, George Washington University, USA.

Adamson, George W. and J. Boreham (1974) *The use of an association measure based on character structure to identify semantically related pairs of words and document titles*. Information Storage and Retrieval,. Vol 10, pp 253-260

Al-Fedaghi Sabah S. and Fawaz Al-Anzi (1989) *A new algorithm to generate Arabic root-pattern forms.* Proceedings of the 11th National Computer Conference, King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia., pp04-07

Al-Kharashi, I. and M. Evens (1994) *Comparing words, stems, and roots as Index terms in an Arabic Information Retrieval system*. Journal of the American Society for Information Science, 45/8, pp. 548-560

Al-Najem, Salah R. (1998). *An Explanation of Computational Arabic Morphology.* DATR Documentation Report, University of Sussex.

Al-Raya (1997) Newspaper. Quatar.

Al-Shalabi, R. and M. Evens (1998) *A Computational Morphology System for Arabic*. Proceedings of COLING-ACL, New Brunswick, NJ.

Al-Watan (2000) Newspaper. Qatar.

Beesley, K.B. (1996) *Arabic Finite-State Morphological Analysis and Generation.* Proceedings of COLING-96, pp 89-94.

Beesley, K.B. (1998) *Arabic Morphological Analysis on the Internet.* Proceedings of the 6th International Conference and Exhibition on Multi-Lingual Computing, Cambridge.

El-Sadany, T. and M. Hashish (1989) *An Arabic morphological system*. IBM System Journal, 28/4

Harman, D. (1991) *How effective is suffixing?* Journal of the American Society for Information Science, 42/1, pp 7-15.

Hmeidi, I., Kanaan, G. and M. Evens (1997) *Design and Implementation of Automatic Indexing for Information Retrieval with Arabic Documents*. Journal of the American Society for Information Science, 48/10, pp. 867-881.

Kiraz, G. (1994) *Multi-tape two-level Morphology: a case study in Semitic non-linear morphology.* Proceedings of COLING-94, pp180-186.

Lovins, J.B. (1968) *Development of a Stemming Algorithm*. Mechanical Translation and Computational Linguistics, 11/1.

Popovic, M. and P. Willet (1992) *The effectiveness of stemming for natural language access to Sloven textual data*. Journal of the American Society for Information Science, 43/5, pp. 384-390.

Porter, M.F. (1980) *An Algorithm for suffix stripping*. Program, 14 /3, pp 130-137

Stalls, B. and Knight, K. (1998) *Translating names and technical terms in Arabic text*. Proceedings of COLING-ACL, New Brunswick, NJ, 1998

van Rijsbergen, C. J. (1979) *Information Retrieval*. Butterworths, London.

Robertson, A. and Willett, P.(1992) *Searching for historical word-forms in a database of 17th-century English text using spelling-correction methods*. 15th Annual International Conference SIGIR.

Ubu-Salem H., Al-Omari M., and M. Evens (1999) *Stemming methodologies over individual query words for an Arabic information retrieval system*. Journal of the American Society for Information Science. 50/6, pp 524-529.