Word-Oriented Approximate String Matching Using Occurrence Heuristic Tables: A Heuristic for Searching Arabic Text

Suleiman H. Mustafa
Department of Computer Information Systems, Yarmouk University, Irbid, Jordan. E-mail: smustafa@yu.edu.jo

In this article, a word-oriented approximate string matching approach for searching Arabic text is presented. The distance between a pair of words is determined on the basis of aligning the two words by using occurrence heuristic tables. Two words are considered related if they have the same morphological or lexical basis. The heuristic reports an approximate match if common letters agree in order and noncommon letters represent valid affixes. The heuristic was tested by using four different alignment strategies: forward, backward, combined forward–backward, and combined backward–forward. Using the error rate and missing rate as performance indicators, the approach was successful in providing more than 80% correct matches. Within the conditions of the experiments performed, the results indicated that the combined forward–backward strategy seemed to exhibit the best performance. Most of the errors were caused by multiple-letter occurrences and by the presence of weak letters in cases in which the shared core consisted of one or two letters.

Introduction

Approximate string matching is a fundamental method of text searching. Assessing similarity between different but related words (such as inflections of the same word root) can be important in various areas of text processing, especially in the area of free-text information retrieval. Approximate string matching techniques are capable of finding word variants (Pirkola, Keskustalo, Leppanen, Kansala, & Jarvelin, 2002). But, as the term *approximate* indicates, because the links established between terms or concepts are not based on *exact matching*, as defined by pattern matching algorithms, retrieval is inevitably probabilistic. However, the naive approach of recording a match between two words only if they are fully identical often produces a baseline performance level from which significant improvement can be sought (Gu & Berleant, 2000).

Received March 10, 2004; revised October 8, 2004; accepted October 8, 2004

 Published online 9 September 2005 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/asi.20244

Traditionally, approximate string matching has been carried by out using lexically based conflation techniques, by which words are stemmed to join different word variants that might be considered semantically related under certain conceptual assumptions. A stemming algorithm is a computational procedure that seeks to reduce all words with the same stem to a common form, usually by stripping each word variant of its derivational and inflectional suffixes (Ekmekcioglu, Lynch, Robertson, Sembok, & Willett, 1996). The results largely depend on the technique being used and the inherent lexical structure of the language under consideration. Two words might have the same lexical base but not necessarily have similar semantic contents.

The term *approximate string matching* has frequently been used in the literature to refer to a class of pattern matching techniques, by which K errors are allowed between a pattern and a text substring. It consists of finding all substrings of the text that have at most K errors with the pattern as determined by an *edit distance*. The *edit distance* between two strings a and b is the minimal number of *edit operations* needed to transform a into b. The edit operations allowed are deleting, inserting and replacing a character (Baeza-Yates & Navarro, 1996).

Another class of approximate matching techniques has used N-gram measures to determine the similarity between a pair of strings. In its simplest form, the similarity between a pair of words is a function of the number of N-character substrings that they have in common. Dice's similarity coefficient is usually used to calculate the similarity value (Kosinov, 2001). N-gram techniques group words that contain identical character substrings of length N on the basis of ranking and using a similarity threshold of a given value.

The three classes of techniques differ in their underlying theoretical assumptions and application orientations. Whereas stemming algorithms are word-oriented and language dependent, the other two classes can work with both words and sequences of substrings and apply language-independent statistical measures and algorithms. In natural language processing and information retrieval, word-based analysis is more natural and appropriate. These techniques have been used for these applications, and their performance has been assessed and compared in a number of ways (Zobel & Dart, 1995; Baeza-Yates & Navarro, 1997).

A large body of research has analysed these three classes of approximate matching techniques. A review of the literature of this research is beyond the scope of this article, but it is important to note that their applications have found their way to the literature of Arabic string matching and text searching. A study by Mustafa (2003), has reported that morphology-driven string matching, in which the word-based matching process is supposed to be driven by an automatically computed root, offered more than 90% valid string matching results. Several other studies have addressed word-based string matching using N-gram techniques (De Roeck & Al-Fares, 2000; Mustafa & Al-Radaideh, 2004). The empirical results indicate that these techniques appear to be highly efficient under certain conditions and for certain applications.

The string matching technique being presented in this article is word oriented. It is based on single textual words rather than on random sequences of substrings. It takes words of a text as independent units for matching and assumes an underlying lexical structure that has to be considered in matching a query word with the text. Formally stated, given a text T of n words and a query word Q of length m, both being sequences of letters from a natural language alphabet Σ, find the set of morphological and lexical variants of Q (denoted F_Q) in T along with the multiple occurrences of F_Q.

Word-to-word parallelism based on occurrence heuristic tables as used in this research draws upon ideas from exact pattern matching algorithms and approximate string matching techniques. The occurrence heuristic table is simply an array of the same size as the alphabet for storing a given textual word. It is used for establishing the required parallelism between a source word (presumably from a text) and a target word. The idea of using this type of string matching structure was introduced by Boyer and Moore (1977) in their well-known algorithm for pattern matching (which became known as the *Boyer–Moore algorithm*). Later the idea was used by Baeza-Yates and Perleberg (1996) for approximate pattern matching. In both algorithms, the pattern used was a random sequence of characters.

A Matching Heuristic

A General Model

Word-oriented string matching is based on a general model (Figure 1), in which a given word Q is looked up in a given text T by matching Q with every word W in T. In approximate matching, a match is reported if Q and W are similar at a certain level of similarity. The intended final target is a set of all similar words in T. The measure of similarity can be determined by means of a language-independent statistical procedure or by means of morphology-driven alignment. The heuristic presented here adopts the latter approach.

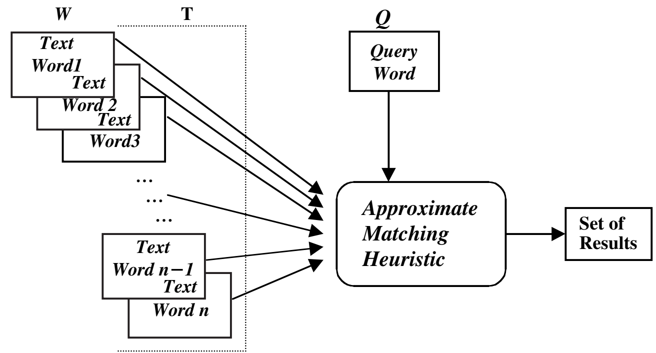

FIG. 1. A general model of word-oriented string matching.

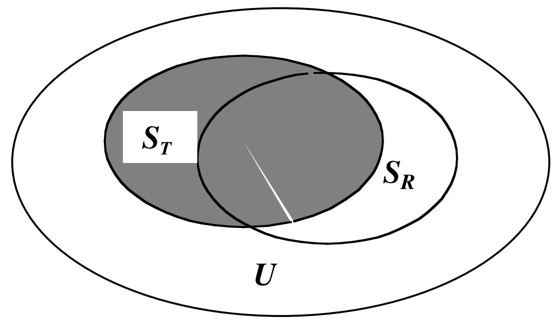

FIG. 2. The actual subset of words similar to the query word (S_T) and the recognized subset as determined by a heuristic.

The kind of heuristic used determines both the accuracy and the completeness levels of the set of results obtained. Let

- U denote the set of all words in T
- S_T denote the subset of all words that are similar to Q in T ($S_T \leq U$)
- S_R denote the subset of words judged by the heuristic to be similar to Q ($S_R \leq U$).

The best we can aim for is $S_R = S_T$, which is rarely the case in approximate matching. The normal case is similar to the situation shown in Figure 2. Some of the elements of S_T might be skipped but others from outside S_T might be erroneously inserted into S_R.

Occurrence Heuristic Tables

The matching process relies on two major data representations: The first is a hashed parallel-array structure for representing the letters of a pair of words, Q and W (Figure 3); the other is a parallel-array structure for distinguishing shared from nonshared letters between Q and W (Figure 4).

The first representation is intended to draw a correspondence between a query word Q and a word W from the text T. It is a parallel-array structure of the same size as the Arabic alphabet. As shown in Figure 3, the two components of the structure (α_Q and α_W) are indexed by the Unicode collating sequence of the Arabic alphabet (which has its basis in ASMO-49). Given, for instance, the two Arabic words,

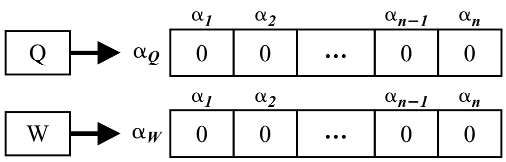

FIG. 3. α_Q and α_W form a parallel-array structure for mapping the correspondence between the letters of a pair of words, Q and W. The subscript α_i represents the Unicode collating sequence of the Arabic alphabet.

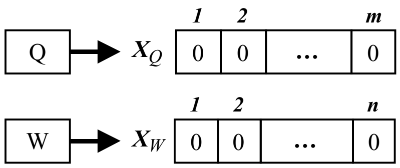

FIG. 4. X_Q is an array showing the letters of Q that are not found in W, and X_W is an array showing the letters of W that are not found in Q. The subscript refers to the proper order of letters in Q or W.

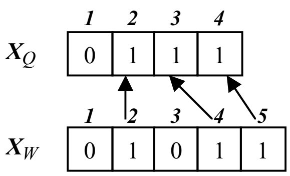

FIG. 5. X_Q and X_W for representing the similarity between *YLBS* and *MLABS*.

YLBS (wearing) and *MLABS* (clothes),[1] when the letters of each word are hashed to their proper locations, we can tell that the two words have three letters in common, *L*, *B*, and *S*, since these three letters hash to the same locations.

The second data representation, on the other hand, is intended to distinguish shared from nonshared letters between a query word Q and a word W from the text T. A zero value in some position P in X_Q indicates that the letter $Q[P]$ is not available in W. By the same token, a zero value in some position P in X_W indicates that the letter $W[P]$ is not available in Q. In either case, a nonzero value indicates otherwise. Given that the shared letters of Q and W match in order, X_W and X_Q are taken as a basis for making the necessary checking on word affixes in W and Q. In the same example, X_Q and X_W take the values indicated in Figure 5.

To determine that the two words are related, we must determine that the letters indicated by zero values are legal affixes.

The Word Matching Process

In order to determine that W is probably related to Q, there are three conditions that must be satisfied: The two words must have a number of letters in common, the common letters must agree in order, and the nonshared letters must form valid affixes in the language under consideration. Figure 6 gives the basic framework of the word matching heuristic.

1. Get a query word
   Initialize α_Q to zeros, and X_Q to zeros
   Input a query word Q
   For each letter in Q increment its position in α_Q
2. Repeat
   2.1 Get the next textual word W
      Initialize α_Q and X_W to zero, and the temporary strings C_{QW} and C_{WQ} to null
      Extract the next word W from the text T
   2.2 Find the letters shared between Q and W
      For each letter in W (except *alif*), if $\alpha_Q[W_i] > 0$ then
         Append W_i to C_{WQ} (i.e., maintain the order of these letters in W) and Inc $X_W[I]$
      For each letter in Q (except the long-vowel letter *alif*), if $\alpha_W[Q_i] > 0$ then
         Append Q_i to C_{QW} (i.e., maintain the order of these letters in Q) and Inc $X_Q[I]$
   2.3 Match C_{QW} with C_{WQ}. If *MATCH* (C_{QW} , C_{WQ}) then
      Check query affixes based on the zero values in X_Q
         If the letters corresponding to zero values (if any) in X_Q form valid prefixes, suffixes, and infixes then *QAFX* is true
      Check word affixes based on the zero values in X_W
         If the letters corresponding to zero values (if any) in X_W form valid prefixes, suffixes, and infixes then *WAFX* is true
   2.4 If MATCH and *QAFX* and *WAFX* then
      add W to the list of match results
   Until no more words in T

FIG. 6. A matching heuristic based on within-word parallelism.

To satisfy the first condition, the process of matching Q with W starts by mapping the letters of Q to their corresponding positions in α_Q. Similarly, as the text T is being tokenized, the letters of W must be mapped to the proper positions in α_W. In doing so, we can make the necessary mapping between Q and W, to determine which letters are common to both.

To maintain the proper order of the common letters in both Q and W, two temporary strings are used: C_{QW} (i.e., the common letters of Q in W) and C_{WQ} (i.e., the common letters of W in Q). Using the example given in the previous section, these two strings are as follows:

$$(C_{QW} = LBS \text{ and } C_{WQ} = LBS)^2$$

Two words may have two or more letters in common but not necessarily in the same order. If, for instance, W refers to the textual word *MSLUB*[3] (stolen), Q and W have three common letters that are not in the same order, as follows:

$$(C_{QW} = LBS \text{ and } C_{WQ} = SLB)^4$$

[1] "يلبس", which is composed of four letters and pronounced *YALBAS,* with *A* a short vowel, and "ملابس", which is composed of five letters and pronounced *MALABIS,* with first *A* and *I* short vowels, and the second *A* a long vowel.

[2] That is, "لبس", the common letters "ل", "ب", and "س" for both cases.

[3] That is, "مسلوب", which is composed of five letters and pronounced *MASLOUB*.

[4] That is, "لبس" for the first case and "سلب" for the second.

A matching common core in a pair of words (e.g., $C_{QW} = C_{WQ}$) does not necessarily indicate that the two words are morphologically related. Therefore, the next step is to determine that the difference (if any) between Q and W is caused by the presence of affixes. We do so by referring to X_Q and X_W. Using the example given in Figure 5, we can see that

- Q has a one-letter prefix that does not exist in W (i.e., the letter *ya*, Y), and
- W has a one-letter prefix (i.e., the letter *meem*, M) and a one-letter infix (i.e., the long-vowel letter *alif*, A) that are not found in Q.

These noncommon letters are checked against the appropriate set of Arabic affixes.[5] If all of them are valid affixes, we can conclude that W is morphologically related to Q and hence we have an approximate word-based match.

It is important to note that step 2.2 in the heuristic can be carried out in two different ways: forward parallelism and backward parallelism. In the first method, X_Q and X_W are constructed by following the natural order of letters in a given word (i.e., starting from the first letter and moving forward until the last letter). In the other method, the parallelism between Q and W is derived by tracing them in reverse order (i.e., starting from the last letter and moving in reversed order to the first letter). Consider again the example in Figure 5. In forward parallelism, the construction of X_Q, for instance, follows this order: $X_Q[1], X_Q[2], \ldots, X_Q[4]$. In backward parallelism, the construction of X_Q, follows reversed order: $X_Q[4], X_Q[3], \ldots, X_Q[1]$. The advantage of using either way or combining the two will be shown in the next section.

Another note that has to be made here concerning step 2.2 is the exclusion of the letter *alif* from the common core of Q and W as represented by C_{QW} and C_{WQ}. The decision to remove it was made because the lexical structure of the majority of Arabic textual words involves one or more occurrences of this letter. In some cases, it represents a long vowel; in others it performs a suffixing or prefixing function. After some experimentation, it was found that ignoring the *alif* would have a significant impact on the results produced by the heuristic.

Experimental Testing

The Data Set

The work presented in this article is based on a corpus of Arabic textual data that represents different subject areas and on a set of textual query words that have been selected randomly from the corpus. The corpus is taken from the author's own experimental data sets, which have been used in previous studies. Each query word has been carefully checked within the corpus, and its morphological and lexical variants and their multiple occurrences have been determined manually. The list of query words, which is too long to be presented here, comprised 1,500 (of 6,490 in the corpus) distinct variants and 7,255 repeated occurrences, with an overall average of about 5 occurrences per word. It was important to make sure that the list represented all types of verbs and nouns and several affixation patterns. Table 1 shows a sample of the list of query words used in the experiments as explained in the following sections.

TABLE 1. A sample of the query words used in the study along with their variants and occurrences in the textual corpus.

Word		Relevant variants	Repeated occurrence
أموال	*amwal*	15	53
الإسكان	*aliskan*	9	32
تطوير	*tadhweer*	24	99
ضمانة	*damanah*	9	21
للمجتمع	*lilmojtama'*	52	369
والفنون	*walfonoun*	10	22
وصلاحياتها	*wasalahyatoha*	18	37

Experiments

The word matching heuristic outlined previously has been exposed to four experiments, each of which represents a different strategy: two major strategies (which are referred to as the *forward strategy* and the *backward strategy*) and two variations on them (which are referred to as the *combined for–back strategy* and the *combined back–for strategy*).

The forward strategy represents the normal approach of building a matching parallelism between a pair of words. In the first experiment, X_Q and X_W were constructed by following the natural order of letters in a given word (i.e., starting from the first letter and moving forward to the last letter). Figure 7 shows how this strategy was implemented in Pascal code. When the procedure is called for a given word W, the list of parameters takes the form (W, C_{WQ}, α_Q, X_W). When, on the other hand, the procedure is called for a given query word Q, the list of parameters takes the form (Q, C_{QW}, α_W, X_Q).

Consider, for instance, a query word such as *ASALIBHA*[6] and the textual word *ASLUB*[7] (with the first *A*, in both words, not a long vowel). Applying the forward approach gives us the representations shown in Figure 8, in which C_{QW} and C_{WQ} have four common letters (i.e., *A, S, L, B*).

The backward strategy, on the other hand, scans a given word in reversed order. Hence, X_Q and X_W were constructed

[5] For efficiency purposes, binary search is used for accessing the list of prefixes and suffixes.

[6] That is, "أساليبها", which is composed of eight letters and pronounced *ASALIBOHA*.

[7] That is, "أسلوب", which is composed of five letters and pronounced *OSLOUB*.

```
Procedure forward (token: toktype; var C:
  toktype; var α: α type; var X: xtype);
var i: integer;
begin
   for i := 1 to LEN(token) do
     if α[token[i]] > 0 then
         begin
         if(token[i] <> '|') then
             begin
             inc(X[i]);
             C := C + token[i]; {concatenate}
             end;
         dec(α[token[i]])
         end;
end;
```

FIG. 7. Using the forward strategy to implement Step 2.2 in the heuristic.

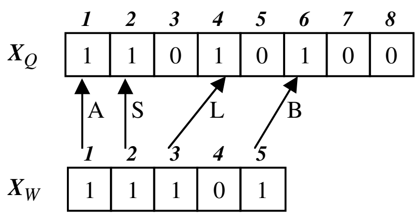

FIG. 8. Representing the two words *ASALIBHA* (X_Q) and *ASLUB* (X_W) using the forward strategy.

```
Procedure backward (token: toktype; var C:
  toktype; var α: α type; var X: xtype);
var i: integer;
begin
   for i := LEN(token) down to 1 do {*}
     if α[token[i]] > 0 then
         begin
         if(token[i] <> '|') then
             begin
             inc(X[i]);
             C := token[i] + C; {*}
             end;
         dec(α[token[i]])
         end;
end;
```

FIG. 9. Using the backward strategy to implement Step 2.2 in the heuristic.

in the second experiment starting from the last letter. Figure 9 shows how this strategy was implemented in Pascal code. The difference between the code in this figure and the code in Figure 7 appears in only two places as marked by the comment symbol {*}. The form of procedure call also remains the same as indicated previously.

For the majority of words, the results of applying both strategies are the same, as is the case with the example shown in Figure 8. But consider the textual word *ALASLUB*[8]

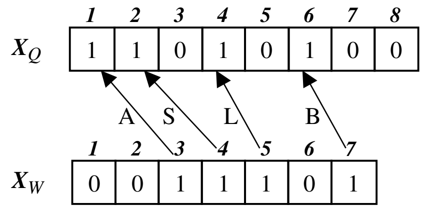

FIG. 10. Representing the two words *ASALIBHA* (X_Q) and *ALASLUB* (X_W) using the backward strategy.

which is related to the same query word in Figure 7. If we apply the forward strategy, the matching heuristic fails to recognize this word *W* as one of the relevant variants of *Q* in the text.

The reason is that this textual word has two occurrences of the letter *lam*, *L*, whereas we have one occurrence of this letter in the query word. Constructing X_Q and X_W in forward direction leads to mapping the first occurrence of *lam* in *W*, thus leaving the second occurrence to take a zero value. Because the letter *lam* does not occur as an infix, the forward matching strategy fails to relate *W* to *Q*.

But this is not the case with backward matching. Figure 10 shows how *Q* and *W* are represented in X_Q and X_W using the backward strategy. Because the two sets of common letters match in order, this strategy succeeds in recognizing the word *ALASLUB* as a relevant variant of the query word *ASALIBHA*. The reader should be reminded that according to the matching heuristic, the long-vowel letter *alif* is ignored; hence, a zero value is given for each occurrence of this letter in both representations (X_Q and X_W).

In the third and fourth experiments, a combination of the forward and backward strategies was used. In the third experiment, the forward strategy was supplemented by the backward strategy. In the fourth experiment, the backward strategy was supplemented by the forward strategy. If a common core exists between *Q* and *W* and the basic strategy fails to recognize a relationship between the two words, the other strategy is called in for verification. If both strategies fail, *W* is reported as being outside the set of suggested relevant variants or multiple occurrences of *Q* in *T*.

Experimental Results

The results of conducting the four experiments discussed were analyzed in terms of two major performance parameters: the error rate and the missing rate. Each refers to a different type of incorrect judgment made by the heuristic in view of the valid relevant word variants in the experimental data set. The first type is given as a ratio of the number of erroneous hits to the total number of words suggested by the heuristic as relevant variants. The second type is a ratio of the number of actual relevant word variants that were not recognized by the heuristic to the total number of actual variants in the data set.

Table 2 presents the results of this error analysis in terms of distinct word variants along with their multiple

[8] That is, "الأسلوب", which has seven letters and is pronounced *ALOSLOUB*.

TABLE 2. Mean average performance of the heuristic using four strategies, in terms of distinct textual words and their multiple occurrences in the text *T*.

	Forward		Backward		Combined for–back		Combined back–for	
Performance	Distinct	Multiple	Distinct	Multiple	Distinct	Multiple	Distinct	Multiple
Error rate	0.17	0.27	0.14	0.22	0.15	0.20	0.19	0.25
Missing rate	0.35	0.44	0.16	0.24	0.13	0.09	0.12	0.22

TABLE 3. Distribution of items (as distinct textual words) judged by the heuristic as being relevant on the basis of the core of letters common to *Q* and *W*.

		With common core = 1 or 2		With common core (3 to $N-1$)		Exact match common core = N	
Strategy	Grand total	Total	Percentage	Number	Percentage	Number	Percentage
Forward	1,206	219	18.2	887	73.5	100	8.3
Backward	1,518	262	17.3	1,156	76.2	100	6.6
For–bak	1,595	273	17.1	1,222	76.6	100	6.3
Bak–for	1,688	322	19.1	1,266	75.0	100	5.9

TABLE 4. Distribution of valid relevant items (as distinct textual words) based on the common core for *Q* and *W*.

		With common core = 1 or 2		With common core = (3 to $N-1$)		Exact match common core = N	
Strategy	Relevant total	Relevant	Percentage	Relevant	Percentage	Relevant	Percentage
Forward	1,004	78	7.8	827	82.4	100	10.0
Backward	1,300	102	7.8	1,098	84.5	100	7.7
For–bak	1,352	105	7.8	1,147	84.8	100	7.4
Bak–for	1,360	125	9.2	1,134	83.4	100	7.4

occurrences in *T*. It shows the performance of the four strategies performed in respect to the two measures: error rate and missing rate. Given the forward matching strategy, for instance, 0.17 of the distinct items suggested by the strategy were not in the valid set of word variants. This value is represented by 0.27 multiple occurrences in the data set. Taking a balanced combination of both measures, the combined for–back strategy (i.e., forward supplemented by backward) seems to exhibit the best performance in comparison to the other strategies.

A judgment made by the matching heuristic to consider *W* morphologically related to *Q* does not necessarily hold true for all cases. A pair of words might be related merely by coincidence. Consider, for example, the case of W = *ALBSTAN*[9] (farm). This word has three letters in common with *Q* (i.e., $C_{WQ} = LBS$) and by coincidence, the remaining four letters can be treated by the matching heuristic as valid affixes. According to the heuristic, this word is judged as morphologically related to *YLBS*, whereas it actually is not.

Table 3 presents the results judged as being relevant approximate matches by the heuristic distributed over three categories of common core size: common = 1 to 2 letters, common = 3 to $n - 1$ and common = n (i.e., exact match). Using the same distribution, Table 4 gives for each strategy the number of valid cases of the cases in Table 3 and their distribution, as a percentage, over the three common core categories.

Further analysis of the results indicates that the majority of the erroneous cases that were reported by the heuristic as relevant variants belong to a category of words that have a common core of one or two letters (i.e., smaller than the size of a trilateral Arabic root). As Table 5 indicates, about 60% or more of the invalid variants (depending on the strategy used) fall into this category. Note that when the common core is equal to *N* (i.e., exact match), the value of nonrelevant becomes zero for all strategies. As we examine the cases in which we have the majority of errors, we find that most of these cases represent words that have *weak letters*: letters that are known to be subject to different types of transformation.

[9]That is, "البستان", which has seven letters and is pronounced *ALBOSTAN*.

TABLE 5. Distribution of nonrelevant items (as distinct textual words) according to the core of letters common to Q and W.

Strategy	Total nonrelevant	With common core = 1 or 2		With common core = (3 to $N-1$)	
		Nonrelevant	Percentage	Nonrelevant	Percentage
Forward	201	141	70.1	60	29.9
Backward	218	160	73.4	58	26.6
For–bak	243	168	69.1	75	30.9
Bak–for	329	197	59.9	132	40.1

Discussion

As we examine the technique adopted in this research and the empirical results presented so far we can make a number of observations: The first is that the use of occurrence tables provides an efficient way of finding the common core between a pair of words. However, computing the distance between a pair of words is sensitive to the lexical structure of words. A large number of errors of judgment made by the heuristic were caused by the presence of multiple-letter occurrences that are introduced by prefixes or suffixes.

Words that have different letters are expected to exhibit a different level of performance, but this condition is not normal. A relevant example in this context is that Arabic dictionaries are organized on the basis of roots, an organization that raises the following question: Does it make a difference if we start by removing candidate suffixes and prefixes from the pair of words being matched (i.e., Q and W) and then apply the matching heuristic presented in this study? This question is worth considering in further research.

Another observation that has to be made relates to an implied assumption about affixes: that affixes are equally likely to occur in their proper lexical positions in any word. Although this assumption might be practically true for the majority of words that have a matching common basis, in some cases this assumption led to the erroneous results, especially when the common core was one or two letters. A recent unpublished investigation by the author indicated that more than 60% of Arabic textual words involve valid prefixes or suffixes. We should add to that the fact that prefixes and suffixes can be combined to compose longer affixes. Given that point, considering nonmatched letters as valid affixes might degrade the matching process.

A third observation that should be made relates to the type of words that are considered relevant variants of a given query word. The concept of relevance, as applied in this study, is root based. A textual word is considered a relevant variant of a given query word if the two words are morphologically related. Although the searching technique presented here is textual anyway and, as in all text searches, has no semantic component, the morphological relevance issue might raise the following question at the application level in information retrieval: Do all words that are morphologically related necessarily have similar semantic content?

This question involves a philosophical and debatable issue because it has no definite answer. In some cases, the semantic link between different forms of the same root is quite strong. In some others, the link cannot be clearly identified. For information retrieval applications, the answer depends on the levels of recall and precision. The closer we get to the root in building hyperlinks, the higher is the expected level of recall and the lower is the precision. As far as the heuristic approach of this study is concerned, the matching ideas presented here can still be used for stem-based matching as well. In this case, a modification of the part that deals with affixes in the heuristic is required.

Finally, as we consider the complexity of a word-based approximate matching method, as presented in this article, we should note that its efficiency is influenced by a number of factors. The first is the number of words in the text T, which should be extracted and then matched with the query word Q, and the word length, which is on average about five Arabic letters. The second factor is the type of data structures used in the matching heuristic and the way they are accessed. The heuristic relies on three basic structures:

1. α, an array structure indexed by the alphabet as determined by the collating sequence in the standard character set
2. X, an array structure indexed by the positions of letters in the pair of words being matched
3. AFX, a sorted list of prefixes and suffixes accessed by using binary search

The number of direct accesses to α and X is determined by the number of letters in a given word; access to AFX is determined by the presence or absence of a prefix or a suffix in the word being processed.

The third factor is the mode by which letters that are common to the pair of words (W and Q being matched) are compared. The heuristic matches W and Q on the basis of shared letters as represented by the two temporary strings: C_{QW} (i.e., the common letters of Q in W) and C_{WQ} (i.e., the common letters of W in Q). The comparison of C_{QW} and C_{WQ} is carried out by using the string operation provided by the given programming language.

Given these three parameters, the performance of the heuristic is almost comparable to that of the classic brute-force solution to the linear searching problem, which is estimated at $O(mn)$, where n refers to the size of T and m refers to the length of the query pattern Q. Considering the fact that an approximate matching approach to word-based string matching has a broader objective than that of exact matching techniques, the lower-level efficiency of inexact matching techniques is justifiable. An important feature of the technique described in this article, in terms of efficiency, is its reliance on heuristic tables that are accessed directly as described earlier. Likewise, the access of affixes is kept at the minimal level of execution cost.

Conclusion

This article reports the results of applying a heuristic approach for approximate word-based matching. The matching heuristic presented here was applied to a set of Arabic data. An underlying rationale of the research was that word-based approximate matching can be performed on the basis of finding the distance between a pair of words on a lexical basis. Given a core of letters that are common to the source word and target word, one of the two words can be transformed into the other if the noncommon letters are determined to be valid affixes.

The results indicate that the method adopted was almost as efficient as other techniques that have been reported in respect to Arabic string searching. Of the four word alignment strategies that were investigated, the combined forward–backward strategy appeared to provide the best performance in terms of the two indicators used: the error rate and missing rate.

Further research should address some of the ideas discussed in the previous section. It is possible to modify the heuristic and apply it to different levels of word stemming. But the results will be highly influenced by both the type and the level of stemming used. One can also suggest that the heuristic presented in this paper be applied to information retrieval applications.

References

Baeza-Yates, R., & Navarro, G. (1996). A fast heuristic for approximate string matching. Proceedings of the Third South American Workshop on String Processing (WSP'96) (pp. 47–63). Retrieved June 2002, from http://ftp.dcc.uchile.cl/pub/users/gnavarro/wsp96.2. ps.gz

Baeza-Yates, R., & Navaro, G. (1997). Fast approximate string matching in a dictionary. Retrieved August 2003, from ftp://ftp.doc.uchile.cl/pub/users/rbaeza/papers/ vocab.ps.gz

Baeza-Yates, R., & Perleberg, C.H. (1996). Fast and practical approximate string matching. Information Processing Letters, 59, 21–27.

Boyer, R., & Moore, S. (1977). A fast string matching algorithm. Communications of the ACM, 20, 762–772.

De Roeck, A.N., & Al-Fares, W. (2000). A morphologically sensitive clustering algorithm for identifying Arabic roots. Proceedings of the 38th Annual Meeting of the ACL, Hong Kong. Retrieved June 2003, from http://citsseer.nj.nec.com /deroeck00morphologically.html

Ekmekcioglu, F., Lynch, M.F., Robertson, A.M., Sembok, A.M., & Willett, P. (1996). Comparison of N-gram matching and stemming for term conflation in English, Malay, and Turkish texts. Text Technology, 6, 1–14.

Gu, Z., & Berleant, D. (2000, October). Hash table sizes for storing N-grams for text processing (Technical Report, No. 10-00a). Ames: Iowa State University. Retrieved June 2003, from http://citenseer.nj.nec.com/347012.html

Kosinov, S. (2001, September). Evaluation of N-grams conflation approach in text-based information retrieval. InfoTech Oulu: International Workshop on Information Retrieval, Oulu, Finland. Retrieved June 2002, from www.syslab.ceu.hu/~serge/ir200/ir2001_kosinov_s. pdf

Mustafa, S.H. (2003). A morphology-driven string matching approach to Arabic text searching. The Journal of Systes and Software, 67, 77–87.

Mustafa, S.H., & Al-Radaideh, Q.A.(2004). Using N-grams for Arabic text searching. Journal of the American Society for Information Science & Technology, 55(11), 1002–1007.

Pirkola, A., Keskustalo, H., Leppanen, E., Kansala, A., & Jarvelin, K. (2002). Targeted s-gram matching: A novel n-gram matching technique for cross- and monolingual word form variants. Information Research, 7(2). Retrieved June 2003, from http://InformationR.net/ir/7-2/paper126.html

Zobel, J. & Dart, P. (1995) Finding approximate matches in large lexicons. Software—Practice and Experience, 25(3), 331–345.