In this article we propose a distance-based classifier for categorizing Arabic text. Each category is represented as a vector of words in an $m$-dimensional space, and documents are classified on the basis of their closeness to feature vectors of categories. The classifier, in its learning phase, scans the set of training documents to extract features of categories that capture inherent category-specific properties; in its testing phase the classifier uses previously determined category-specific features to categorize unclassified documents. Stemming was used to reduce the dimensionality of feature vectors of documents. The accuracy of the classifier was tested by carrying out several categorization tasks on an in-house collected Arabic corpus. The results show that the proposed classifier is very accurate and robust.

Introduction

Text categorization is the process of assigning a category label from a set of predefined category labels to an unlabeled document (Al-Mubaid, 2003; Meretakis, Fragoudis, Lu, & Likothanassis, 2000; Sebastiani, 2002). Text categorization is a vital research area nowadays as there are huge volumes of data available online, and it has many applications such as classification of news stories, e-mail messages, and Web pages (Al-Mubaid, 2003). A wide range of text categorization algorithms have been developed such as the naive Bayes classifier (Eyheramendy, Lewis, & Madiagn, 2003), the chain augmented naive Bayes classifier (Peng, Schuurmans, & Wang, 2004), support vector machines (He, Tan, & Tan, 2003), generalized discriminant analysis (Li, Zhur, & Ogiwara, 2003), the $k$-nearest neighbors algorithm (He et al., 2003), the optimized $k$-nearest neighbors using P trees (Rahal & Perrizo, 2004), neural networks (Ruiu & Srinivasan, 1999), and generalized instance sets (Lam, 2003). Most of these algorithms were tested against English text; however, some were applied to Chinese text as well.

Very few text categorization techniques were applied to Arabic text. Sawaf, Zaplo, and Ney (2001) used maximum entropy technique for Arabic document clustering. Document clustering, in their approach, started by randomly assigning documents to clusters. In subsequent iterations, documents were shifted from a cluster to another only if an improvement was gained. The algorithm terminated when no further improvement could be achieved. Therefore, they used unsupervised learning for Arabic text classification, whereas the work reported here uses supervised learning. A direct comparison of our results and theirs is not feasible because different techniques were employed and different corpora were used. The work of El-Kourdi, Bensaid, and Rachidi (2004), by comparison, used a naive Bayes classifier to classify in-house collected Arabic documents. Again direct comparison of their work and ours is not feasible because of the differences in the corpora and categorization algorithms employed.

This article proposes a distance-based classification technique in which categories are represented as feature vectors in an $m$-dimensional category space. Unclassified documents are classified on the basis of their closeness to category vectors. During training phase, the classifier scans the training documents once to extract category-specific features. These features are represented by using the bag-of-words model (Dumais, Plat, Heckerman, & Sahami, 1998); i.e., the model does not take word order into consideration. At the end of training, each category will be described as a feature vector that contains words that appeared in documents that are known to belong to the category under consideration. The accuracy of the proposed classifier was tested against an in-house collected Arabic text corpus that was gathered from online newspapers and magazines. The collected corpus contains 1,000 documents that vary in length and writing styles and fall into 10 categories (100 documents per category). One-half of these documents were used for training and the other half were used for testing. The documents were preprocessed by removing punctuation marks and stopwords. Word stemming was used as a filtering mechanism to reduce the number of features extracted from documents and therefore to reduce the dimensionality of the feature vectors. The classifier performed very well, as shown in the experimentation section of this article.
The rest of the article is organized as follows: The Arabic Language provides a gentle introduction to the Arabic language. Document Preprocessing, on the other hand, highlights the major functions that are usually performed in preprocessing documents to prepare them for text categorization with emphasis on the Arabic language. The Proposed Classifier section describes the proposed classifier in detail. Experimentation and Result Analysis follows. The final section discusses conclusions and future work.

The Arabic Language

Arabic is used by more than 250 million Arabs. It is also understood by more than a billion Muslims worldwide as the Koran (the Muslims’ holy book) is written in Arabic.

Arabic belongs to the Semitic group of languages. The Arabic alphabet consists of 28 characters:

In addition, the Arabic hamza (ا) is sometimes considered a letter in Arabic linguistics. The letters (أوغ) are vowels; the others are consonants. Arabic is written from right to left. Arabic letters have different styles when appearing in a word depending where the letter appears (beginning, middle, or end of a word) and on whether the letter can be connected to its neighbor letters or not. For example, the letter (ت) has the following styles: (ت) if it appears at the beginning of a word (such as the word تُريِ، which means body); (ت) if the letter appears in the middle of a word (such as the word دُعِيِ، which means honorable); (ت) if the letter appears at the end of a word (such as the word دُعِيِ، which means heart). Finally, the letter (ب) can appear as (ب) if it appears at the end of a word but disconnected from the letter located to its right (such as the word دُلَّلِ، which means fabulous).

Diacritics are signals placed below or above letters to double the letter in pronunciation or to act as a short vowel. The following diacritics are used in Arabic:

(ợ) Arabic shada, which is used to double the letter on which it appears, for example, the word مَحَمَّد (محمّد) would appear as مَحَمَّد (محمّد); the letter (ا) is written twice without the shada, which is an extremely strange style for Arabic readers.

(ọ) Arabic dama, which may appear above a letter and act as a short waw (א).

(ọ) Arabic fathah, which may appear above a letter and act as a short alif (א).

(ọ) Arabic kasra, which may appear below a last letter of a word and mean that the last letter of the word is pronounced as if it has kasra followed by the sound of letter (א).

A letter with a diacritic is represented, in the computer, as two characters: one for the letter and one for the diacritic. Diacritics are often omitted when writing Arabic texts. Native Arabic readers can pronounce the words correctly even when diacritics are missing. Usually the position of a word in a sentence determines its diacritics. Different letter styles and diacritics make parsing Arabic text a nontrivial task.

Document Preprocessing

Preprocessing is the first step in dealing with text documents to present them in a format or intermediate form suitable for the task at hand such as document clustering, categorization, and summarization. Text documents consist of strings of characters, digits, and special symbols. In document preprocessing, words that best describe a document are extracted and others are ignored. Therefore, punctuation marks, formatting tags, prepositions, pronouns, conjunctions, and auxiliary verbs (referred to as stopwords) are often removed. The remaining words are retained and referred to as keywords. At the end of document preprocessing, the document is mapped from sequences of strings to a list of keywords usually called a feature vector.

Arabic is a very rich language; often a verb in its root pattern is augmented with prefixes, infixes, and suffixes to refer to the time during which the event occurred and whether the verb is plural or singular, as well as the gender of the participants in the verb. For example, the word (ذَهَبُ) , which corresponds to the English verb go, can have several patterns, for instance, if the prefix (ص) is added to the verb, it becomes (صذَهَبُ) and the time of the verb is present, done by one male. But if, on the other hand, the suffix (ت) is added to the verb, then it becomes (ذَهَا) and means that the time of the event is past, number of participants is two, and they are males. Table 1 shows a few examples of the format in which the verb go (ذَهَبُ) may appear.

The richness of the Arabic language creates a dimensionality problem for the feature vectors: a feature vector of a document can grow very large because of the different forms of the verb. For instance, the verb (ذَهَبُ) has the following different forms:

<table>
<thead>
<tr>
<th>Verb</th>
<th>Time</th>
<th>Number of participants</th>
<th>Gender of subjects</th>
</tr>
</thead>
<tbody>
<tr>
<td>ذَهَبُ</td>
<td>Past</td>
<td>One</td>
<td>Male</td>
</tr>
<tr>
<td>ذَهَبُ</td>
<td>Past</td>
<td>One</td>
<td>Female</td>
</tr>
<tr>
<td>ذَهَبُ</td>
<td>Past</td>
<td>Two</td>
<td>Male</td>
</tr>
<tr>
<td>ذَهَبًا</td>
<td>Past</td>
<td>Two</td>
<td>Female</td>
</tr>
<tr>
<td>ذَهَا</td>
<td>Past</td>
<td>Three or more</td>
<td>Male</td>
</tr>
<tr>
<td>ذَهَنَ</td>
<td>Past</td>
<td>Three or more</td>
<td>Female</td>
</tr>
<tr>
<td>ذَهَنُ</td>
<td>Present</td>
<td>One</td>
<td>Male</td>
</tr>
<tr>
<td>ذَهَنَ</td>
<td>Present</td>
<td>One</td>
<td>Female</td>
</tr>
<tr>
<td>ذَهِنٌ</td>
<td>Future</td>
<td>One</td>
<td>Male</td>
</tr>
<tr>
<td>ذَهِنٌ</td>
<td>Future</td>
<td>Three or more</td>
<td>Female</td>
</tr>
</tbody>
</table>

TABLE 1. Different forms of the verb go (ذَهَبُ) in Arabic.
a word can take. Fortunately, Arabic has a built-in filtering mechanism; words can be mapped into their root patterns by using stemming. Root patterns in Arabic are three-, four-, five-, or six-letter patterns. More than 80% of Arabic words can be mapped into three-letter root patterns.

Reducing a word to its root pattern reduces the number of words from hundreds of thousands to 4,749 (Eldos, 2003). Roots are semantically weak in the sense that several words can be mapped into the same root, thus losing the sense (past, present, or future), the participants in the verb, and so on. For example, the several forms of the verb go ( \( \text{ذَهَبَ} \) ), which appear in Table 1, are reduced to the three-letter root pattern ( \( \text{ذَهَبُ} \) ). Despite this, mapping a word to its root pattern reduces the dimensionality of document feature vectors.

Root extracting or stemming algorithms for Arabic text fall into two groups. First are algorithms that remove prefixes, infixes, and suffixes from words and then map them into a set of predefined root patterns. In this style, a word may need to be scanned several times before it can be mapped into its root pattern. The work reported by Al-Shalabi and Evans (1998), El-Sadany and Hashish (1989), Gheith and El-Sadany (1987), and Hilal (1990) is in this category. Second are algorithms that employ a letter weight and order scheme in which letters in a word are given weights and are assigned ranks or orders; then the root is extracted by processing these assigned weights and ranks. The work reported by Al-Shalabi, Kanaan, and Al-Serhan (2003) is an example of an algorithm in the second category.

This article uses the work reported by Al-Shalabi and colleagues (2003) for root extraction. Al-Shalabi, Kanaan, and Al-Serhan (2003) extract word roots by assigning weights and ranks to the letters that constitute a word. Weights are real numbers in the range 0 to 5. The mapping of weights to letters was determined by extensive experimentation with Arabic text. In the study by Al-Shalabi and associates (2003), the Arabic alphabet was divided into six groups; the first group consisted of the letters ( \( \text{غ} \) ) and its members were assigned a weight of 5. The second group consisted of the letters ( \( \text{ه} \) ) and its members were assigned a weight of 3.5. The third group consisted of the letters ( \( \text{م} \) ) and its members were assigned a weight of 3. Fourth group consisted of the letters ( \( \text{ن} \) ) and its members were given a weight of 2. The fifth group consisted of the letters ( \( \text{س} \) ) and its members were assigned a weight of 1. The sixth group consisted of the rest of the Arabic alphabet and was assigned a weight of 0.

The rank or order of letters in a word depends on the length of that word and on whether the word contains odd or even number of letters. Table 2, adapted from Al-Shalabi and coworkers (2003), shows the assignment of ranks to letters. \( N \) is the number of letters in a word.

After determination of the weight and rank of every letter in a word, letter weights are multiplied by the letter rank. The three letters with the smallest product value constitute the root (read from right to left). Table 3 shows an example of using Al-Shalabi, Kanaan, and Al-Serhan’s (2003) root extractor.

### Table 2. Rank or order of letters in a word (adapted from Al-Shalabi, Kanaan, and Al-Serhan, 2003).

<table>
<thead>
<tr>
<th>Letter position from right</th>
<th>Rank (if word length is even)</th>
<th>Rank (if word length is odd)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( N )</td>
<td>( N )</td>
</tr>
<tr>
<td>2</td>
<td>( N - 1 )</td>
<td>( N - 1 )</td>
</tr>
<tr>
<td>3</td>
<td>( N - 2 )</td>
<td>( N - 2 )</td>
</tr>
<tr>
<td>[ \frac{N}{2} ]</td>
<td>( \frac{N}{2} + 1 )</td>
<td>( \frac{N}{2} )</td>
</tr>
<tr>
<td>[ \frac{N}{2} ]</td>
<td>( \frac{N}{2} + 1 - 0.5 )</td>
<td>( \frac{N}{2} + 1 - 1.5 )</td>
</tr>
<tr>
<td>[ \frac{N}{2} ]</td>
<td>( \frac{N}{2} + 2 - 0.5 )</td>
<td>( \frac{N}{2} + 2 - 1.5 )</td>
</tr>
<tr>
<td>[ \frac{N}{2} ]</td>
<td>( \frac{N}{2} + 3 - 0.5 )</td>
<td>( \frac{N}{2} + 3 - 1.5 )</td>
</tr>
</tbody>
</table>

### Table 3. An example of using the Al-Shalabi, Kanaan, and Al-Serhan (2003) root extractor.

<table>
<thead>
<tr>
<th>Word</th>
<th>( \text{ة} )</th>
<th>( \text{د} )</th>
<th>( \text{ر} )</th>
<th>( \text{م} )</th>
<th>( \text{س} )</th>
<th>( \text{ن} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Letters</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Weight</td>
<td>6.5</td>
<td>5.5</td>
<td>4.5</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Rank</td>
<td>32.5</td>
<td>5.5</td>
<td>22.5</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>Product</td>
<td>32.5</td>
<td>5.5</td>
<td>22.5</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
</tbody>
</table>

Al-Shalabi and colleagues’ (2003) work handles only three-letter roots. Fortunately, more than 80% of Arabic words have three-letter roots and the words that are commonly used in Arabic writings have three-letter roots. The accuracy of their root extractor was reported to be above 90%.

### The Proposed Classifier

This section explains how the classifier was built and used for categorizing new text documents.

#### Training Phase

Let \( \Delta = \{ TD_1, TD_2, \ldots, TD_n \} \) be document examples that belong to category \( C_i \), where \( n \) is the number of documents in that category. Therefore, these documents were categorized by human categorizer and were found to belong to the category \( C_i \) and thus called positive examples. To build the feature vector of category \( C_i \), every \( TD_j \in \Delta \) is processed in the following manner:

First, punctuation marks and stopwords are removed from the document. Even this standard step in processing documents has its special flavor when dealing with Arabic text. For example, the conjunction letter \( \text{و} \) ( \( \text{و} \) ), which corresponds to and in English, can be confused with words that start with the letter \( \text{ف} \). Consider the word ( \( \text{وَدَفَّت} \) ), which means “arrived” in English; this word starts with the letter \( \text{ف} \), but this letter is an integral part of the word. However, the \( \text{ف} \) in the word ( \( \text{فَاتَ} \) ), which means “gone,” is extra and in fact represents a conjunction letter. One way to eliminate this
confusion is to map a word to its root pattern in which extra letters are removed from words.

Second, roots of the remaining words are extracted to reduce the dimensionality of category-specific feature vectors.

Third, the extracted roots are added to the feature vector of category $C_i$.

The process was repeated for every category. Thus, the feature vector of a category consists of keywords in root format, which are present in the union of the training documents that are known to belong to this category. Therefore, if a word appears one or more times in a document that is known to belong to $C_i$, that word is added to the feature vector of category $C_i$. At the end of the training phase, the $m$-dimensional space, against which unclassified documents are compared, is created. It consists of a feature vector per category; $m$ is the number of categories.

Testing Phase

To categorize a new document, such as $X$, it is preprocessed by removing punctuation marks and stopwords, and then extracting the roots of the remaining keywords. After that, the feature vector of $X$ is compared with the feature vectors of the categories one at a time. The Dice measure (as defined in equation 1; Ganesan, Garcia-Molina, & Widom, 2003) was used to compute the similarity. Once the dice similarity of transitioning from category one to another is calculated, the category with the highest similarity is assigned to the document.

$$\text{sim}_{\text{Dice}}(F_X, F_Y) = \frac{2 \times |F_X \cap F_Y|}{|F_X| + |F_Y|}$$

where $F_X$ and $F_Y$ are feature vectors of document $X$ and category $Y$, respectively, and $||$ means the norm (i.e., the number of symbols in a list), for example, $|F_Y|$ means the number of words that appear in the vector $F_Y$.

Experimentation and Result Analysis

To test the performance of the proposed classifier, the Arabic text corpus was collected from online magazines and newspapers; 1,000 documents that vary in length and writing style were collected. These documents fall into 10 predefined categories. Every category contains 100 documents. The set of predefined categories included sports, economic, Internet, art, animals, technology, religion, politics, medicine, and plants. Two individuals manually categorized the collected documents; and every document was assigned to only one category; whenever a document was found to belong to more than one category, it was assigned to the category with the maximal likelihood according to human categorizer’s judgment. For every category, 50 documents were randomly specified and used for training, and the remaining 50 were used for testing.

The accuracy of the classifier is expressed in terms of recall, precision, fallout, and error rate, as described by Lewis (1995). For the benefit of the reader, the formulae are repeated here.

Consider a binary classification problem (i.e., there are only one category and $n$ documents that need to be classified), so a given document either belongs to this category (i.e., positive example) or does not belong to that category (i.e., negative example). Assume that the classification is carried out by two classifiers: the first is a human and the second is a computer program. Then recall ($Re$), precision ($Pr$), fallout, and error rate are defined as

$$Re = \frac{a}{(a + c)}$$

$$Pr = \frac{a}{(a + b)}$$

$$\text{Fallout} = \frac{b}{(b + d)}$$

$$\text{Error rate} = \frac{(b + c)}{(a + b + c + d)}$$

where $a =$ number of documents that both the human and the computer classify as positive examples, $b =$ number of documents that the human classifies as negative examples but the computer classifies as positive examples, $c =$ number of documents that the human classifies as positive example but the computer classifies as negative examples, $d =$ number of documents that both the human and the computer classify as negative documents, and $a + b + c + d = n$ (total number of test documents).

The experiment that we have performed included 500 test documents to be classified against 10 categories; i.e., the experiment was a 10-way classification problem. Table 4 shows recall, precision, fallout, and error rate for every category. It also shows the microaverage of these values for all categories. As can be seen from Table 4, recall reaches its highest value (0.98) for the economic category, and the lowest value (0.22) for the Internet category. The second lowest

<table>
<thead>
<tr>
<th>Category</th>
<th>Recall</th>
<th>Precision</th>
<th>Fallout</th>
<th>Error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Animal</td>
<td>0.6600</td>
<td>0.9167</td>
<td>0.0067</td>
<td>0.0400</td>
</tr>
<tr>
<td>Art</td>
<td>0.8400</td>
<td>0.3925</td>
<td>0.1444</td>
<td>0.1460</td>
</tr>
<tr>
<td>Economics</td>
<td>0.9800</td>
<td>0.3952</td>
<td>0.1667</td>
<td>0.1520</td>
</tr>
<tr>
<td>Internet</td>
<td>0.2200</td>
<td>0.6471</td>
<td>0.0133</td>
<td>0.0900</td>
</tr>
<tr>
<td>Medicine</td>
<td>0.6800</td>
<td>0.7727</td>
<td>0.0222</td>
<td>0.0520</td>
</tr>
<tr>
<td>Plant</td>
<td>0.5800</td>
<td>0.9355</td>
<td>0.0044</td>
<td>0.0460</td>
</tr>
<tr>
<td>Politics</td>
<td>0.4400</td>
<td>1.0000</td>
<td>0.0000</td>
<td>0.0560</td>
</tr>
<tr>
<td>Religion</td>
<td>0.5800</td>
<td>0.8529</td>
<td>0.0111</td>
<td>0.0520</td>
</tr>
<tr>
<td>Sport</td>
<td>0.9200</td>
<td>1.0000</td>
<td>0.0000</td>
<td>0.0080</td>
</tr>
<tr>
<td>Technology</td>
<td>0.3800</td>
<td>0.4872</td>
<td>0.0444</td>
<td>0.1020</td>
</tr>
<tr>
<td>Microaverage</td>
<td>0.6280</td>
<td>0.7400</td>
<td>0.0413</td>
<td>0.0744</td>
</tr>
</tbody>
</table>
value (0.38) for recall was for the technology category. When
the classifier’s output was reexamined, a large percentage of
the misclassified documents in the Internet category were
categorized under the Technology category, and vice versa.
The reason for this misclassification is that documents that
belong to the Internet category and those that belong to the
Technology category share many common words. We assume
that a hierarchical classifier is more appropriate as Internet
would be placed as a subcategory of Technology. Precision
reaches its highest value (1.00) for the Politics and Sport cat-
gerories and its lowest value for the Art category (0.3925).
The rest of the table is self-explanatory.

Figures 1 through 4 depict the variations of recall, pre-
cision, fallout, and error rate over the 10 categories, re-
spectively.

Conclusions and Future Work
This article has presented a distance-based classifier for
Arabic text categorization. The classifier collects informa-
tion about the categories under consideration in its learning
phase. A category is described by a set of keywords that form
its feature vector, which is constructed by scanning a set of
previously categorized documents (called positive examples).

To categorize an uncategorized document, it is scanned to
extract its feature vector, then is compared with the pre-
defined set of feature vectors for the categories at hand. The
winning category is the category that has the maximal simi-
larity (using the Dice measure) with the document’s feature
vector. The uncategorized document is assigned to only one
category.

Processing text written in Arabic was a nontrivial task
because of the richness of the language. A word can have
many forms according to its location in a sentence, letters
can have several styles according to their position in a word
and their neighboring letters, and Arabic uses diacritics. The
sizes of the feature vectors can grow very large for large
documents, and therefore stemming was used as a filtering
mechanism. The accuracy of the classifier was measured by
using recall, precision, fallout, and error rate. The classifier’s
accuracy was very good, as indicated in the Experimentation
and Results Analysis section. From the experimentation that
was carried out on an in-house collected corpus, it was con-
cluded that the classifier is robust.

We plan to investigate the suitability of other classifiers
for Arabic text categorization such as Bayesian networks
and k-nearest-neighbor. We also plan to investigate the suit-
ability of filtering mechanisms such as the information gain
of words or word clustering.

Acknowledgments
The author would like to thank Iyad Al-azzam and
Mohammed Al-Refai for the help they provided in collecting
the corpus and running the classifier. The author also thanks
the anonymous reviewers, whose useful comments helped in
shaping the final draft.

References
matic text categorization. Paper presented at the International Arab Con-
ference on Information Technology (ACIT 2003), Alexandria, Egypt.
system for Arabic. Paper presented at the Workshop in Semitic Language
Processing, COLING-ACL’98, University of Montreal, Montreal, Canada.


