Arabic Text Classification Using Support Vector Machines

Tarek Fouad Gharib, Mena Badieh Habib, and Zaki Taha Fayed

tgharib@asunet.shams.edu.eg menabad@gmail.com ztfayed@hotmail.com

Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt

Summary
Text classification (TC) is the process of classifying documents into a predefined set of categories based on their content. Arabic language is highly inflectional and derivational language which makes text mining a complex task. In this paper we applied the Support Vector Machines (SVM) model in classifying Arabic text documents. The results compared with the other traditional classifiers Bayes classifier, K-Nearest Neighbor classifier and Rocchio classifier. Two experiments used to test the different classifiers. The first uses the training set as the test set, and the second uses Leave one testing method. Experimental results performed on a set of 1132 document show that Rocchio classifier gives better results when the size of feature set is small while SVM outperform the other classifiers when the size of the feature set is large enough. Classification rate exceeds 90% when using more than 4000 feature. Leave one method gives more realistic results over the use of training set as a test set.

Key words: Text mining, text categorization, Arabic language, Support Vector Machines

1. Introduction

The rapid growth of the Internet has increased the number of online documents available. This has led to the development of automated text and document classification systems that are capable of automatically organizing and classifying documents. Text classification is the process of classifying documents into a predefined set of categories based on their content. This assignment can be used for classification, filtering, and retrieval purposes. Machine learning approaches are applied to build an automatic text classifier by learning from a set of previously classified documents [1]. Many text categorization systems have been developed for English and other European languages, but according to a performed survey there are few researches for Arabic text categorization till the day of writing this paper. Arabic language is a Semitic language that has a complex and much morphology than English, it is a highly inflected language, and due to this complex morphology it needs a set of preprocessing routines to be suitable for manipulation. The document in text categorization system must pass through a set of steps: document conversion which converts different types of documents into plain text, stop word removal to remove insignificant words, stemming to group words sharing the same root, feature selection/extraction, super vector construction, feature weighting, classifier construction, classification, evaluation of the classifier. In our previous work [9] we have tested different approaches of stemming, feature selection, and feature weighting. We proposed a new stemming and feature selection approaches for Arabic documents classification. In this paper we will focus on the classification phase, different classifiers widely used in text categorization have been applied and compared.

The paper is organized as follow. Section 2 discusses the related works on Arabic text categorization. The system architecture is presented in section 3. Section 4 presents the algorithms classifiers used in this work. The experimental results are discussed in section 5. Finally, conclusion is presented in section 6.

2. Related Works

Few researches tackled the area of Arabic text classification. Hassan Sawaf et al. [2] used a statistical method called maximum entropy to classify Arabic News articles. Another statistical classifier – Naïve Bayes – was used by Mohamed El Kourdi et al. [3] to categorize Arabic web documents; using the root based stemmer to extract the roots of the words.

Rehab M. Duwairi [4] proposed a distance-based classification technique on a set of 1000 document. Root based stemmer was used to extract the root of the words. The algorithm build a feature vector for each category and the test document is compared with the categories vector to find the most similar one. The comparison is performed using the Dice similarity measure.

An intelligent system based on statistical learning for searching in Arabic text was presented by Reda A. El-Khoribi el al. [5]. The authors used the light stemmer for preprocessing, hidden markov models for feature extraction and Bayes classifier for classification.

The N-gram frequency statistics technique was used by Laila Khreisat [6] along with the dice similarity
measure. In this system an N-gram profile was generated for every document by extracting all the N-grams of the words in the document and sorting them according to their frequency on the document from most frequent to least frequent. The similarity measure used to determine the most similar document to the test document.

The first use of SVM for Arabic text classification was presented by Abdelwadood Moh’d A MESLEH [7]. The author used the CHI Square technique for feature selection. No stemming algorithms were used. The results of SVM outperform the other algorithms, k-NN and Navie Bayes classifiers.

Alaa M. El-Halees [8] proposed an Arabic text categorization system that uses the maximum entropy method for classification. The paper studied the effect of stemming and words normalization accuracy. Also the authors presented a new method called part-of-speech for extracting nouns and proper nouns showing the importance of stemming and preprocessing on the classification accuracy.

Most of the above schemas provide no details about how stemming or feature selection is done. Also no comparison of classification method is provided to show which classifier is outperforming other classifiers.

3. System Architecture

Many text categorization systems have been developed for English and other European languages, however few researchers work on text categorization for Arabic language as shown in the previous section. In a previous work, Arabic text categorization systems have been proposed [9].

Figure 1 shows the different phases of the system. The document in text categorization system must pass through a set of steps: document conversion which converts different types of documents into plain text, stop word removal to remove insignificant words, stemming to group words sharing the same root, feature selection/extraction, feature weighting, and classification.

3.1 Stop words removal and stemming

3.1.1 Stop words removal

Stop words like prepositions and particles are considered insignificant words and must be removed. A list of 165 words was prepared to be eliminated from all the documents.

3.1.2 Stemming

Stemming is the process of removing all affixes from a word to extract its root. It is essential to improve performance in information retrieval tasks especially with highly inflected language like Arabic language. There are three different approaches for stemming: the root-based stemmer; the light stemmer; and the statistical stemmer.

Root-based stemmer [10] uses morphological analysis to extract the root of a given Arabic word, while the aim of the light stemming approach [11] is not to produce the root of a given Arabic word, rather is to remove the most frequent suffixes and prefixes. In statistical stemmer [12], related words are grouped based on various string similarities measures; such approach often involves n-gram which is a set of n consecutive characters extracted from a word. The main idea behind this approach is that, similar words will have a high proportion of n-grams in common.

An improvement has been performed to statistical stemmer by applying light stemmer before performing similarity measure in order to maximize the performance of the statistical stemmer. Results show that the hybrid approach of light and trigram stemming with is the most suitable stemming approach for Arabic text categorization.

3.2 Document indexing

After stop words removal and words stemming, documents are indexed and represented as a vector of weighted terms. In true information retrieval style, each document is usually represented by a vector of n weighted terms; this is often referred to as the bag of words approach to document representation [13]. In this approach the structure of a document and the order of words in the document are ignored. A global super vector is constructed. It consists of all the distinct words (also called terms) that appear in all training samples of all classes after removing the stop words and words stemming.

3.2.1 Term selection

Typically, there can be thousands of features in document classification. Hence, a major characteristic, or
difficulty of text categorization problems is the high dimensionality of the feature space. For this, term selection techniques are used to select from the super vector terms a subset of terms that are deemed most useful for compactly representing the meaning of the documents. Term selection is also beneficial in that it tends to reduce over fitting, (i.e. the phenomenon by which a classifier tends to be better at classifying the data it has been trained on than at classifying other data). Usually, term selection techniques consist of scoring each term in the super vector by means of a term evaluation function f (TEF) and then selecting a set of terms that maximize f. Many term evaluation functions have been introduced for term selection for English text categorization [14, 15, 16]. These functions are Document Frequency Threshold, Information Gain, CHI Square, Odds Ratio, NGL Coefficient and GSS Score [1].

In this phase we faced a problem in the feature selection phase. When global feature selection approach was used, most of the documents did not contain any term in the list of the selected terms (empty documents). In other words, term evaluation functions select terms with rare appearance in the data set (i.e. terms with very low document frequency). This problem motivated the use of a hybrid approach between document frequency threshold information gains. Document frequency is used to remove rare terms and information gain to select most informative terms from the remaining list. Also we used a hybrid approach [17] that combines local and global feature selection to select effective features to improve efficiency. This approach gives a high classification rate comparatively with global feature selection and, in the same time, reduces the number of empty documents.

3.2.2 Term weighting
After selecting the most significant terms in the super vector, each document is represented as a weighted vector of the terms found in the super vector. Every word is given a weight in each document. There are many suggested weighting schemes [16] such as Boolean weighting, Term Frequency (TF) weighting, Term Frequency Inverse Document Frequency (TFIDF) weighting, and Normalized-TFIDF weighting. Normalized-TFIDF schema is chosen as the best schema for term weighting.

3.3 Classification algorithms
Two different non-parametric classifiers have been used; k-NN and Rocchio classifiers. A result shows that Rocchio classifier is outperforms k-NN classifier in both time and accuracy.

4. Text Classifiers
Most commonly used classifiers in the field of text categorization were used. These classifiers are k-NN classifier, Rocchio classifier, Bayes classifier and Support Vector Machines (SVM).

4.1 k-Nearest Neighbor Classifier
To classify an unknown document vector d, the k-nearest neighbour (k-NN) algorithm ranks the document's neighbours among the training document vectors, and use the class labels of the k most similar neighbours to predict the class of the input document [1,18]. The classes of these neighbours are weighted using the similarity of each neighbour to d, where similarity may be measured by for example the Euclidean distance or the cosine between the two document vectors. The Euclidean distance is used as a conventional method for measuring distance between two documents, the formula of the Euclidean distance between documents \( d1(w11,w12,…,w1n) \) and \( d2(w21,w22,…,w2n) \) is as follow:

\[
E(d1,d2) = \sqrt{\sum_{i=1}^{n} (w_{2i} - w_{1i})^2}
\]

k-NN has been applied to text categorization since the early days of its research. However, it has a set of drawbacks. k-NN is a lazy learning example-based method that does not have a off-line training phase. The main computation is the on-line scoring of training documents given a test document in order to find the k nearest neighbors, this makes k-NN not efficient because nearly all computation takes place at classification time rather than when the training examples are first encountered, k-NN time complexity is \( O(N*M) \) where N is number of training documents and M is the number terms for each document vector. Moreover, k-NN classifier has a major drawback of selecting the value of k; the success of classification is very much dependent on this value. The Rocchio method however can deal with those problems to some extent as shown in the next section.

4.2 Rocchio Classifier
Rocchio is the classic profile-based classifier used for document routing or filtering in information retrieval [19]. In this method, a prototype vector is built for each class ci, and a document vector d is classified by calculating the distance between d and each of the prototype vectors [1,18]. The prototype vector for class ci is computed as
the weighted average vector over all training document vectors that belong to class \( c_i \). This means that learning is very fast for this method compared to the k-NN classifier.

The weighted average of a category \( c_i(\text{wi}_1,\text{wi}_2,...,\text{wi}_n) \) is computed as follow:

\[
    w_i = \beta \sum_{\text{doc} \in \text{POS}_i} \frac{w_{jk}}{\text{POS}_i} - \gamma \sum_{\text{doc} \in \text{NEG}_i} \frac{w_{jk}}{\text{NEG}_i}
\]

Where \( w_{jk} \) is the weight of the term \( \text{tk} \) in document \( d_j \). \( \text{POS}_i \) is the set of documents that belongs to \( c_i \) (positive examples), and \( \text{NEG}_i \) is the set of documents that doesn’t belongs to \( c_i \) (negative examples). In this formula, \( \beta \) and \( \gamma \) are control parameters that allow setting the relative importance of positive and negative examples. For instance, if \( \beta \) is set to 1 and \( \gamma \) to 0, the profile of \( c_i \) is the centroid of its positive training examples. In general, the Rocchio classifier rewards the closeness of a test document to the centroid of the positive training examples, and its distance from the centroid of the negative training examples. The role of negative examples is usually de-emphasized, by setting \( \beta \) to a high value and \( \gamma \) to a low one (e.g. use \( \beta=1.6 \) and \( \gamma=0.4 \) [20]).

The Rocchio method deals with k-NN problems to some extent. It uses the generalized instances to replace the whole collection of training instances by summarizing the contribution of the instances belonging to each category. Besides its efficiency this method is easy to implement, since learning a classifier basically comes down to averaging weights and classifying a new instance only needs computing the Euclidean distance between the new instance and the generalized instances. It can be regarded as a similarity-based algorithm. Its time complexity is considered to be \( O(L*M) \) where \( L \) is number of generalized instances and \( M \) is the number terms for each document vector. Moreover, the Rocchio method can deal with noise to some extent via summarizing the contribution of the instances belonging to each category. For example, if a feature mainly appears in many training instances of a category, its corresponding weight in the generalized instance will have a larger magnitude for this category. Also if a feature mainly appears in training instances of other categories, its weight in the generalized instance will tend to zero. Therefore, the Rocchio classifier can distill out certain relevant features to some extent. On the other hand, one drawback of the Rocchio classifier is it restricts the hypothesis space to the set of linear separable hyperplane regions, which has less expressiveness power than that of k-NN algorithms.

4.3 Naive Bayes Classifier

The Naive Bayes (NB) classifier is a probabilistic model that uses the joint probabilities of terms and categories to estimate the probabilities of categories given a test document [13]. The naive part of the classifier comes from the simplifying assumption that all terms are conditionally independent of each other given a category. Because of this independence assumption, the parameters for each term can be learned separately and this simplifies and speeds the computation operations compared to non-naive Bayes classifiers.

There are two common event models for NB text classification, discussed by McCallum and Nigam [21], multinomial model and multivariate Bernoulli model. In both models classification of test documents is performed by applying the Bayes’ rule [13]:

\[
    P(c_i | d_j) = \frac{P(c_i) \cdot P(d_j | c_i)}{P(d_j)}
\]

Where \( d_j \) is a test document and \( c_i \) is a category. The posterior probability of each category \( c_i \) given the test document \( d_j \), i.e. \( P(c_i | d_j) \), is calculated and the category with the highest probability is assigned to \( d_j \). In order to calculate \( P(c_i | d_j) \), \( P(c_i) \) and \( P(d_j | c_i) \) have to be estimated from the training set of documents. Note that \( P(d_j) \) is same for each category so we can eliminate it from the computation. The category prior probability, \( P(c_i) \), can be estimated as follows:

\[
    P(c_i) = \frac{\sum_{j=1}^{N} y(d_j, c_i)}{N}
\]

Where, \( N \) is number of training documents and \( y(d_j, c_i) \) is defined as follows:

\[
    y(d_j, c_i) = \begin{cases} 
        1 & \text{if } d_j \in c_i \\
        0 & \text{otherwise} 
    \end{cases}
\]

So, prior probability of category \( c_i \) is estimated by the fraction of documents in the training set belonging to \( c_i \). \( P(d_j | c_i) \) parameters are estimated in different ways by the multinomial model and multivariate Bernoulli model. We present multinomial model as it is proved by McCallum [21] that the multinomial model was almost uniformly better than the multivariate Bernoulli model.
4.3.1 Multinomial Model

Multinomial model for Naive Bayes classification is the event model we used and evaluated in our study. In the multinomial model a document \(d_j\) is an ordered sequence of term events, drawn from the term space \(T\). The Naive Bayes assumption is that the probability of each term event is independent of term’s context, position in the document, and length of the document. So, each document \(d_j\) is drawn from a multinomial distribution of terms with number of independent trials equal to the length of \(d_j\). The probability of a document \(d_j\) given its category \(c_i\) can be approximated as:

\[
P(d_j \mid c_i) = \prod_{k=1}^{|d_j|} P(t_k \mid c_i)
\]

Where \(|d_j|\) is the number of terms in document \(d_j\); and \(t_k\) is the \(k^{th}\) term occurring in document \(d_j\). Thus the estimation of \(P(d_j \mid c_i)\) is reduced to estimating each \(P(t_k \mid c_i)\) independently. The following Bayesian estimate is used for \(P(t_k \mid c_i)\):

\[
P(t_k \mid c_i) = \frac{1 + TF(t_k, c_i)}{|T| + \sum_{l \in |T|} TF(t_l, c_i)}
\]

Here, \(TF(t_k, c_i)\) is the total number of times term \(t_k\) occurs in the training set documents belonging to category \(c_i\). The summation term in the denominator stands for the total number of term occurrences in the training set documents belonging to category \(c_i\). This estimator is called Laplace estimator and assumes that the observation of each word is a priori likely [22].

4.4 Support Vector Machine Classifier

Support Vector Machines (SVM) is a relatively new class of machine learning techniques first introduced by Vapnik [23] and has been introduced in TC by Joachims [24]. Based on the structural risk minimization principle from the computational learning theory, SVM seeks a decision surface to separate the training data points into two classes and makes decisions based on the support vectors that are selected as the only effective elements in the training set.

During classification, SVM makes decision based on the \(OSH\) instead of the whole training set. It simply finds out on which side of the \(OSH\) the test pattern is located. This property makes SVM highly competitive, compared with other traditional pattern recognition methods, in

\[
w \cdot x + b = 0
\]

for \(i = 1, 2, \ldots, N\); where the dot product operation \((\cdot)\) is defined by

\[
w \cdot x = \sum_i w_i x_i
\]

for vectors \(w\) and \(x\). Thus the goal of the SVM learning is to find the optimal separating hyper-plane (OSH) that has the maximal margin to both sides. This can be formalized as:

\[
\text{minimize} \quad \frac{1}{2} w \cdot w
\]

subject to

\[
\begin{align*}
& w \cdot x + b \geq +1 \quad \text{if} \quad y_i = +1 \\
& w \cdot x + b \leq -1 \quad \text{if} \quad y_i = -1
\end{align*}
\]

Figure 2 shows the optimal separating hyper-plane.
terms of computational efficiency and predictive accuracy [25].

The method described is applicable also to the case in which the positives and the negatives are not linearly separable. Yang and Liu [25] experimentally compared the linear case (namely, when the assumption is made that the categories are linearly separable) with the nonlinear case on a standard benchmark, and obtained slightly better results in the former case.

Support Vector Machines have been applied successfully in many text classification tasks due to their principle advantages [24]:

- They are robust in high dimensional spaces. Overfitting does not affect so much the computation of the final decision margin.
- Any feature is important. Even some features that could be considered as irrelevant ones have been found to be good when calculating the margin. 
- They are robust when there is a sparsely of samples.
- Most text categorization problems are linearly separable.

5. Experimental Results

This collection consists of 1,132 documents that contain 95138 words (22347 unique words). These documents were collected from the three main Egyptian newspapers ElAhram [http://www.ahram.org.eg/], ElAkhabar [http://www.akhbarelyom.org.eg/], and ElGomhoria [http://www.algomhuria.net.eg/] during the period from August 1998 to September 2004. These documents cover 6 topics. Table 1 shows the number of documents for each topic. Each document has average size of about 84 words before stemming and stop words removal. Document represents the first paragraph of an article, it has been chosen because it usually contains an abstract to the whole article.

<table>
<thead>
<tr>
<th>Topic</th>
<th>No. of documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arts</td>
<td>233</td>
</tr>
<tr>
<td>Economics</td>
<td>233</td>
</tr>
<tr>
<td>Politics</td>
<td>280</td>
</tr>
<tr>
<td>Sports</td>
<td>231</td>
</tr>
<tr>
<td>Woman</td>
<td>121</td>
</tr>
<tr>
<td>Information Technology</td>
<td>102</td>
</tr>
</tbody>
</table>

Experiments were performed using different sizes of features ranges from 2500 to 5000 feature. At each number of features different values for $\beta$ and $\gamma$ for the Rocchio algorithm are used. The $K$ in the $K$-NN classifier was set to the values {1, 3, 5, 7, ..., 19}. The SVM implementation used was the Sequential Minimal Optimization (SMO) algorithm by Platt [26] with a polynomial kernel function. SMO is a fast method to train SVMs. Training an SVM requires the solution of a very large quadratic programming (QP) optimization problem. SMO breaks this large QP problem into a series of smallest possible QP problems. These small QP problems are solved analytically, which avoids using a time-consuming numerical QP optimization as an inner loop. The amount of memory required for SMO is linear in the training set size, which allows SMO to handle very large training sets.

The Naïve Bayes model used was the multinomial model which performs better than the multivariate Bernoulli model at larger vocabulary sizes.

Two experiments used to test the different classifiers. In the first experiment we used the training set of documents as a test set. Leave One method, in the second experiment, used to test the classifiers. All experiments are performed on a computer with 2.8 GHz Pentium4 processor and 512 MB Ram.

When using the training documents as a test set, the results of SVM, Naïve Bayes and Rocchio classifiers were very high and the classification accuracy tend to be 100%. This is because the classifiers have already seen the test documents during training phase and thus it is very trivial to the classifier to classify those documents. While using the Leave One method for testing gives more realistic results. Leave one method involves using a single document from the original sample as the test data, and the remaining documents as the training data. This is repeated such that each document in the sample is used once as the test data.

Comparing the training time of the four classifiers, it seems that Bayes classifier was the most efficient classifier while SVM was the most expensive. Table 2 shows the time taken for training the four classifiers (h:mm:ss).
Table 2: Time taken for training the classifier

<table>
<thead>
<tr>
<th>No. of terms</th>
<th>K-NN</th>
<th>Rocchio</th>
<th>Bayes</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>2500</td>
<td>0:00:03</td>
<td>0:00:16</td>
<td>0:00:42</td>
<td>0:03:45</td>
</tr>
<tr>
<td>3000</td>
<td>0:00:04</td>
<td>0:00:20</td>
<td>0:00:49</td>
<td>0:04:42</td>
</tr>
<tr>
<td>3500</td>
<td>0:00:05</td>
<td>0:00:22</td>
<td>0:00:56</td>
<td>0:05:27</td>
</tr>
<tr>
<td>4000</td>
<td>0:00:08</td>
<td>0:00:28</td>
<td>0:00:63</td>
<td>0:05:59</td>
</tr>
<tr>
<td>4500</td>
<td>0:00:13</td>
<td>0:00:30</td>
<td>0:00:69</td>
<td>0:06:23</td>
</tr>
<tr>
<td>5000</td>
<td>0:00:17</td>
<td>0:00:32</td>
<td>0:00:88</td>
<td>0:06:52</td>
</tr>
</tbody>
</table>

Regarding the classification accuracy results in figures 3 and 4 shows that Rocchio classifier was competitive to the widely used classifier SVM, however the SVM classifier outperforms the other classifiers when the number of features is large. This observation affirms the conclusion of Joachims [24] that SVM has the ability to generalize well in high dimensional feature spaces. Also experimental results show that the best values for $\beta$ and $\gamma$ of the Rocchio classifier were 1.6 and 0.4 respectively.

The Rocchio and the SVM classifiers significantly outperform the Naïve Bayes classifier, while the K-NN classifier significantly underperforms all the other classifiers. The SVM classifier outperforms the Rocchio classifier when the number of features is large enough.

6. Conclusion

This paper presents the result of classifying Arabic text documents using SVM model. SVM classifier significantly outperforms the other classifiers in high dimensional feature spaces. Also using Leave one method gives more realistic results than using training set as a test set. In future work we will investigate the hierarchical classification of Arabic documents. Also we will Building a bigger Arabic Language TC Corpus will be considered as well in our future research.

References


