English & Persian Document Categorization based on a Novel Hybrid Algorithm

Vahid Behravesh[1], S.M.R. Farshchi[2]

[*] Department of Electrical Engineering, Islamic Azad University, Bardaskan Branch, Bardaskan, Iran
[**] Department of Electrical Engineering, Islamic Azad University, Bardaskan Branch, Bardaskan, Iran

Abstract- This paper describes automatic document categorization based on large text hierarchy. We handle the large number of features and training examples by taking into account hierarchical structure of examples and using feature selection for large text data. We experimentally evaluate feature subset selection on real-world text data collected from the existing Web hierarchy named Yahoo. In our learning experiments naive Bayesian classifier was used on text data using feature vector document representation that includes word sequences (n-grams) instead of just single words (unigrams). Experimental evaluation on real-world data collected form the Web shows that our approach gives promising results and can potentially be used for document categorization on the Web. Additionally the best result on our data is achieved for relatively small feature subset, while for larger subset the performance substantially drops. The best performance among six tested feature scoring measure was achieved by the feature scoring measure called Odds ratio that is known from information retrieval.

Index Terms- text mining; text related mining; vector based normalization; text categorizer

I. Introduction

In the last 10 years content-based document management tasks (collectively known as information retrieval—IR) have gained a prominent status in the information systems field, due to the increased availability of documents in digital form and the ensuing need to access them in flexible ways. Text categorization (TC — a.k.a. Text classification, or topic spotting), the activity of labeling natural language hierarchical catalogues of Web resources, and in general any application requiring document organization or selective and adaptive document dispatching.

Until the late 80's the most popular approach to TC, at least in the "operational" (i.e., real-world applications) community, was a knowledge engineering (KE) one, consisting in manually defining a set of rules encoding expert knowledge on how to classify documents under the given categories. In the 90's this approach has increasingly lost popularity (especially in the research community) in favor of the machine learning (ML) paradigm, according to which a general inductive process automatically builds an automatic text classifier by learning, from a set of preclassified documents, the characteristics of the categories of interest. The advantages of this approach are accuracy comparable to that achieved by human experts, and a considerable savings in terms of expert labor power, since no intervention from either knowledge engineers or domain experts is needed for the construction of the classifier or for its porting to a different set of categories. It is the ML approach to TC that this paper concentrates on.

Current-day TC is thus a discipline at the crossroads of ML and IR, and as such it shares a number of characteristics with other tasks such as information / knowledge extraction from texts and text mining [1]. There is still considerable debate on where the exact border between these disciplines lies, and the terminology is still evolving. "Text mining" is increasingly being used to denote all the tasks that, by analyzing large quantities of text and detecting usage patterns, try to extract probably useful (although only probably correct) information. According to this view, TC is an instance of text mining.

TC enjoys quite a rich literature now, but this is still fairly scattered. Although two international journals have devoted special issues to this topic [2-3], there are no systematic treatments of the subject: there are neither textbooks nor journals entirely devoted to TC yet, and Manning and Schutze [4] is the only chapter-length treatment of the subject. As a note, we should warn the reader that the term "automatic text classification" has sometimes been used in the literature to mean things quite different from the ones discussed here. Aside from (i) the automatic assignment of documents to a predefined set of categories, which is the main topic of this paper, the term has also been used to mean (ii) the automatic identification of such a set of categories (e.g.,[5]), or (iii) the automatic identification of such a set of categories and the grouping of documents under them (e.g., Merkl at [6]), a task usually called text clustering, or (iv) any activity of placing text items into groups, a task that has thus both TC and text clustering as particular instances [7].

A number of statistical classification and machine learning techniques has been applied to text categorization, including regression models, nearest neighbor classifiers, decision trees, Bayesian classifiers, Support Vector Machines (SVM), rule learning algorithms, relevance feedback, voted classification, and neural networks.

Document classification, requires encoding Persian & English documents into numerical vectors. A corpus which is a collection of documents is mapped into a list of words as the feature candidates. Among the candidates, only some are selected as the features. For each document, a numerical value is assigned to each of the selected features, depending on the importance and presence of each feature. However, encoding documents so causes the two main problems: huge dimensionality and sparse distribution [6].

In order to solve the two main problems, this research uses the novel method that documents should be encoded into string vectors. A string vector refers to a finite set of strings which are words in context of a natural language. In numerical vectors representing documents, words are given as features, while in string vectors, words are given as feature values. Features of string vectors are defined very variously as properties of words with respect to their posting, lexical category, and statistical properties, but in this research, the highest frequent word, the second highest frequent one, and so on are defined as features of string vectors for easy implementation.

By encoding documents into string vectors, we can avoid completely the two main problems: huge dimensionality and sparse distribution.

We proposed the competitive neural text categorizer, as the approach to text categorization and proposed the application of it to multi-language documents categorization. Before creating the proposed neural network, traditional neural networks, such as MLP (Multi Layers Perceptron) with BP (Back Propagation) receives numerical vectors as its input data. Differently from the traditional neural networks, the proposed neural network receives string vectors. It has the two layers as its architecture: the input layer and the competitive layer. It is expected for the proposed model to improve the performance of multi-language text categorization by solving the two main problems.

This paper is organized as follows. In Section 2 we formally define TC and we review its most important and popular applications. Section 3 describes the main ideas of proposed neural network. The simulation result and experiment was mentioned in section 4. Section 5 concludes, discussing open issues and possible avenues of further research for TC.

II. Text Categorization

a) Formal Description

Text Categorization is the task of assigning a Boolean value to each pair $< d_j, c_i > \in D \times C$, where D is a domain of documents and $C = \{c_1, c_2, ..., c_{|c|}\}$ is a set of predefined categories. A value of T assigned to $< d_j, c_i >$ indicates a decision to file d_j under c_i while a value of F indicates a decision not to file d_j under c_i. More formally the task is to approximate the unknown target function $\Phi := D \times C \to \{T, F\}$ (that describes how documents ought to be classified) by means of a function $\Phi := D \times C \to \{T, F\}$, called the classifier.

b) Related Work

In this section, we will survey previous works relevant to this research, and point out their limitations. There exist other kinds of approaches to text categorization than machine learning based ones: heuristic and rule based approaches. Heuristic approaches were already applied to early commercial text categorization systems [9]. However, we count out the kind of approaches in our exploration, since they are rule of thumbs. Since rule based approaches have poor recall and require a time consuming job of building rules manually as mentioned in the previous section, they are not covered in this article, either. Therefore, this article counts only machine learning based approaches to text categorization considered as state of the art ones.

Typical machine learning algorithms applied traditionally to text categorization are *KNN* (K Nearest Neighbor), *NB* (Naïve Bayes), *SVM* (Support Vector Machine), and *BP* (Back Propagation). The four approaches to text categorization have been used more popularly in previous literatures on text categorization than any other traditional approaches. Among them, the simplest approach is *KNN*. *KNN* is a classification algorithm where objects are classified by voting several labeled training examples with their smallest distance from each object. *KNN* was initially applied to classification of news articles by Massand et al, in 1992 [13]. Yang compared 12 approaches to text categorization with each other, and judged that *KNN* is one of recommendable approaches, in 1999 [21]. *KNN* is evaluated as a simple and competitive algorithm with Support Vector Machine for implementing text categorization systems by Sebastiani in 2002 [19]. Its disadvantage is that *KNN* costs very much time for classifying objects, given a large number of training examples because it should select some of them by computing the distance of each test object with all of the training examples.

Another popular and traditional approach to text categorization is *NB*. Differently from *KNN*, it learns training examples in advance before given unseen examples. It classifies documents based on prior probabilities of categories and probabilities that attribute values belong to categories. The assumption that attributes are independent of each other underlies on this approach. Although this assumption violates the fact that attributes are dependent on each other, its performance is feasible in text categorization [14]. Naïve Bayes is used popularly not only for text categorization, but also for any other classification problems, since its learning is fast and simple [4].

In 1997, Mitchell presented a case of applying *NB* to text categorization in his textbook [14]. He asserted that *NB* was a feasible approach to text categorization, although attributes of numerical vectors representing documents were dependent on each other; this fact contradicts with the assumption underlying in *NB*. In 1999, Mladenic and Grobellink evaluated feature selection methods within the application of Naïve Bayes to text categorization [15]. Their work implied that *NB* is one of standard and popular approaches to text categorization. Androutsopoulos et al adopted *NB* for implementing a Spam mail filtering system as a real system based on text categorization in 2000 [1]. It requires encoding documents into numerical vectors for using *NB* to text categorization.

Another popular and traditional approach to text categorization is *SVM*. Recently, this machine learning algorithm becomes more popular than the two previous machine learning algorithms. Its idea is derived from a linear classifier, Perceptron, which is an early neural network. Since the neural network classifies objects by defining a hyper-plane as a boundary of classes, it is applicable to only linearly separable distribution of training examples. The idea of *SVM* is that if a distribution of training examples is not linearly separable, these examples are mapped into another space where their distribution is linearly separable, as illustrated in the left side of figure 1. *SVM* optimizes the weights of the inner products of training examples and its input vector, called Lagrange multipliers [2], instead of those of its input vector, itself, as its learning process. It defines two hyper-planes as a boundary of two classes with a maximal margin, as illustrated in the left side of figure 1. Refer to [8] or [2], for more detail description on *SVM*.

The advantage of *SVM* is that it is tolerant to huge dimensionality of numerical vectors; it addresses the first problem. Its advantage leads to make it very popular not only in text categorization, but also any other classification problems [2]. In 1998, it was initially applied to text categorization by Joachims [10]. He validated the classification performance of *SVM* in text categorization by comparing it with *KNN* and *NB*. Drucker et al adopted *SVM* for implementing a Spam mail filtering system and compared it with *NB* in implementing the system in 1999 [3]. They asserted empirically that *SVM* was the better approach to Spam mail filtering than *NB*. In 2000, Cristianini and Shawe-Taylor presented a case of applying *SVM* to text categorization in their textbook [2]. In 2002, Sebastiani asserted in his survey paper that SVM is most recommendable approach to text categorization by collecting experimental results on the comparison of *SVM* with other approaches from previous works [19]. In spite of the advantage of *SVM*, it has two demerits. One is that it is applicable to only binary classification; if a multiple classification problem is given, it should be decomposed into several binary classification problems for using *SVM*. The other is that it is fragile to the problem in representing documents into numerical vectors, sparse distribution, since the inner products of its input vector and training examples generates zero values very frequently.

The third popular and traditional approach to text categorization is *BP*. It is most popular supervised neural network and used for not only classification tasks but also nonlinear regression tasks [6]. It is also derived Perceptron, together with *SVM*. When a distribution of training examples is not linearly separable, in *SVM*, the given space is changed into another space where the distribution is linearly separable, whereas in back propagation, a quadratic boundary is defined by adding one more layer, called hidden layer [7][6], as illustrated in the right side of figure 1. More detail explanation about back propagation is included in [7] or [6].

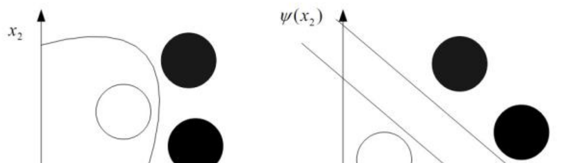

Figure 1. Mapping Vector Space in SVM.

In 1995, *BP* was initially applied to text categorization by Wiener in his master thesis [20]. He used Reuter 21578 [24] as the test bed for evaluating the approach to text categorization and shown that back propagation is better than *KNN* in the context of classification performance. In 2002, Ruiz and Srinivasan applied continually back propagation to text categorization [18]. They used a hierarchical combination of *BP*s, called *HME* (Hierarchical Mixture of Experts), to text categorization, instead of a single *BP*. They compared *HME* of *BP*s with a flat combination of *BP*s, and observed that *HME* is the better combination of *BP*s. Since *BP* learns training examples very slowly, it is not practical, in spite of its broad applicability and high accuracy, for implementing a text categorization system where training time is critical.

Research on machine learning based approaches to text categorization has been progressed very much, and they have been surveyed and evaluated systematically. In 1999, Yang evaluated 12 approaches to text categorization including machine learning based approaches directly or indirectly in text categorization [21]. She judged the three approaches, *LLSF* (Linear Least Square Fit), *K* Nearest Neighbor, and Perceptron, worked best for text categorization. In 2002, Sebastiani surveyed and evaluated more than ten machine learning based approaches to text categorization [19]. He asserted that Support Vector Machine is best approach to text categorization with respect to classification performance. All approaches which were surveyed and evaluated in these literatures require encoding documents into numerical vectors in spite of the two problems.

We explored and presented previous cases of applying one of the four traditional machine learning algorithms to text categorization. Although the traditional approaches are feasible to text categorization, they accompany with the two main problems from representing documents into numerical vectors. In the previous works, dimension of numerical vectors should reserve, at least, several hundred for the robustness of text categorization systems. In order to mitigate the second problem, sparse distribution, a task of text categorization was decomposed into binary classification tasks in applying one of the traditional approaches. This requires classifiers as many as predefined categories, and each classifier judges whether an unseen document belongs to its corresponding category or not.

There is a previous trial to solve the two problems. In 2002, Lodhi et al proposed a string kernel for applying Support Vector Machine to text categorization [11]. In their solution, documents as raw data are used directly for text categorization without representing them into numerical vectors. String kernel is a function computing an inner product between two documents given as two long strings. An additional advantage of the solution is to process documents independently of a natural language in which documents are written. However, their solution was not successful in that it took far more time for computing string kernel of two documents and the version of *SVM* using the string kernel was not better than the traditional version.

As presented in section 5, this research will be a successful attempt to solve the two problems by proposing competitive text classifier with string vectors.

III. Strategies of Encoding Documents

Since documents are unstructured data by themselves, they cannot be processed directly by computers. They need to be encoded into structured data for processing them for text categorization. This section will describe the two strategies of encoding documents with the two subsections: the traditional strategy and the proposed strategy. The first subsection describes the former and points out its demerits, and the second subsection describes the latter and mentions its merits.

a) Numerical Vector

A traditional strategy of encoding documents for tasks of text mining, such as text categorization is to represent them into numerical vectors. Since input vectors and weight vectors of traditional neural networks such as back propagation and RBF (Radial Basis Function) are given as numerical vectors, each document should be transformed into a numerical vector for using them for text categorization. Therefore, this subsection will describe the process of encoding documents into numerical vectors and what are their attributes and values.

Figure 2 illustrates the process of extracting feature candidates for numerical vectors from documents. If more than two documents are given as the input, all strings of documents are concatenated into a long string. The first step of this process is tokenization where the string is segmented into tokens by white space and punctuations. In the second step, each token is stemmed into its root form; for example, a verb in its past is transformed into its root form, and a noun in its plural form is transformed into its singular form. Words which function only grammatically with regardless of a content are called stop words [5], and they correspond to articles, conjunctions, or pronouns. In the third step, stop words are removed for processing documents more efficiently and reliably for text categorization. Through the three steps illustrated in figure 2, a collection of words are generated as feature candidates.

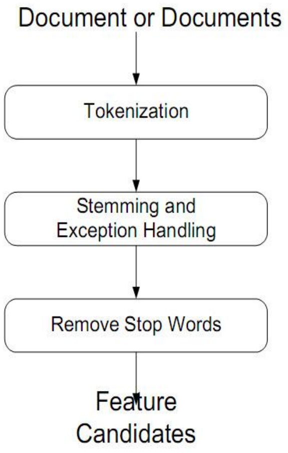

Figure 2. Flowchart of feature extraction of documents.

Since the number of the generated feature candidates is usually too big, using all of them is not feasible as features of numerical vectors. Therefore, only some of them are used as features of numerical vectors for efficiency. A scheme of defining criteria for selecting some of them as features is called feature selection method [15]. Generally, features are selected from the generated collection by their frequencies in the corpus. Therefore, candidates with highest frequencies are used as features of numerical vectors. The number of selected candidates as features becomes the dimension of numerical vectors. There are other feature selection methods than the frequency based one, and they are described in detail in [15] and [19]. However, although only some of the candidates are used as features, the number of features is still large for robust text categorization.

The selected features are given as attributes of numerical vectors and numerical information about attributes become elements of numerical vectors. In this article, we mention the three ways of defining elements as the representative ones, although others may exist. The first way is to assign a binary value indicating absence or presence of the corresponding word in the given document; one indicates its presence and zero indicates its absence. The second way is to define elements as frequencies of corresponding words in the given document; the elements become integers which are greater than or equal to zero. The third way is to assign weights computed from equation (1) to elements of numerical vectors; elements are real values.

$$W(w_k) = tf_i(wk)(\log_2 D - \text{Log}_2 df(w_k) + 1) \quad (1)$$

Where $tf_i(wk)$ is the frequency of words, w_k, D is the total number of document categories in corpus.

As we mentioned above, the process, indexing, is the conversion of text into a list of words as structured form. In this process, a text is given as the input. A string of the text is partitioned into tokens by white space and punctuation mark. Each token is to the basic form based on stemming rules; the word, “studied” is transformed to the basic form, “study”, and the plural form of a noun is changed to its singular form. Among them, stop words, which function only grammatically and are irrelevant with the content of the text, are removed after stemming step.

The collection of texts is also transformed into a bag of words by applying the union operation to all texts. Among union of bags of words, words with higher frequency are selected as attributes of numerical vector, since stop words with higher frequency are removed in the process of indexing texts. If a text is represented into a numerical vector, its attributes are selected words and their values are binary value indicating the presence of the word corresponding to the attribute, integer indicating its frequency in the text, or real value indicating its weight. This article adopted the numerical vector representing a text with binary values.

Note that numerical vectors encoding documents have two main problems as mentioned in section 1. The first problem is that the dimension of numerical vectors is still large. This problems leads to high cost of time for processing each encoded document for training a classifier and to requirement of a very large number of training examples proportionally to the dimension. The second problem is that each numerical vector includes zero values, dominantly. Since the discrimination among numerical vectors over categories is lost, categorization performance is degraded.

b) String Vector

An alternative strategy of encoding documents for text categorization is to represent them into string vectors. In this subsection, we describe this strategy and its advantage in detail. However, this strategy is applicable to only proposed competitive neural text categorizer, while the previous one is applicable to any traditional machine learning algorithm.

A string vector is defined as a finite ordered set of words. In other words, a string vector is a vector whose elements are words, instead of numerical values. Note that a string vector is different from a bag of words, although both of them are similar as each other in their appearance. A bag of words is an infinite unordered set of words; the number of words is variable and they are independent of their positions. In string vectors, words are dependent on their positions as elements, since words correspond to their features. Features of string vectors will be described in detail in the next paragraph.

Features of string vectors are defined as properties of words to the given document. The features are classified into the three types: linguistic features, statistical features, and positional features. Linguistic features are features defined based on linguistic knowledge about words in the given document: the first or last noun, verb, and adjective, in a paragraph, title, or full text. Statistical features are features defined based statistical properties of words in the given documents; the highest frequent word and the highest weighted word using equation (1). Positional features are features defined based on positions of words in a paragraph or the full text: a random word in the first or last sentence or paragraph, or the full text. We can define features of string vectors by combining some of the three types, such as the first noun in the first sentence, the highest frequent noun in the first paragraph, and so on.

We can define features of string vectors in various ways as mentioned above, but in this work, features of string vectors are defined based on only frequencies of words for implementing easily and simply the module of encoding documents into string vectors (see follow). A formal description of string vector is defined as a set of words which is ordered and has its fixed size. It is denoted by where denotes by $[s_1, s_2, ..., s_d]$ where s_i denotes a string, and there are *d* elements. For example, [computer system information] is an instance of a three dimensional string vector. Note that the string vector, [computer system information] is different from the string vector [system computer information], since elements are dependent on their positions like the case in every numerical vector.

Properties of words may be set as features of string vectors. Features of string vectors are defined in one or combined one of three views. In the first views, features are defined based on posting information of words: a random word in the first sentence, a random word in the last sentence, and a random word in the first paragraph. In the second view, they are defined based on linguistic properties of words, such as first noun, first verb, last noun, and last verb. In the third view, they are defined based on their frequencies, such as the most frequent word, the second most frequent word, and the third most frequent word, and so on. As we mentioned above, in this research, the third way of defining features of string vectors is adopted; a strong vector consists of words in the descending order of their frequencies. The reason of defining features of string vectors so is to implement easily and simply the encoder of a text clustering system.

When representing documents into string vectors, their sizes are fixed with d, and it is called the dimension of string vectors. A *d* dimensional string vector consists of *d* words in the descending order of their frequencies in the given entire full text; the first element is the highest frequent word, the second element is the second highest frequent word, and the last element is the *d* the highest frequent word. Figure 3 illustrates the process of encoding a document into its string vector with the simple definition of features. In the first step of figure 3, a document is indexed into a list of words and their frequencies. Its detail process of the first step is illustrated in figure 3. If the dimension of string vectors is set to *d*, *d* highest frequent words are selected from the list, in the second step. In the third step, the selected words are sorted in the descending order of their frequencies. This ordered list of words becomes a string vector representing the document given as the input.

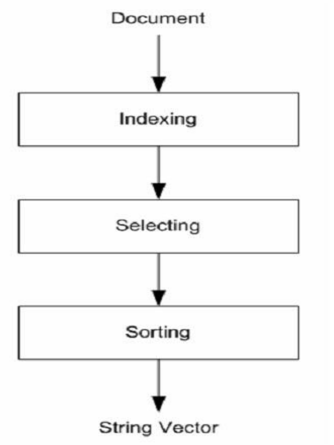

Figure 3. The process of mapping a document into a string vector [2].

Table 1 illustrate differences between string vectors and numerical vectors. The first difference is that numerical values are given as elements in numerical vectors, while strings are given as elements in string vectors. The second difference is that the similarity measure between two numerical vectors is the cosine similarity or the Euclidean distance, while that between two string vectors is the semantic average similarity. The third difference between the two types of structured data is that features for encoding documents into numerical vectors are words, while those for encoding them into string vectors are statistical linguistic and posting properties of words. Therefore, a string vector is the vector where numerical values are replaced by strings in a numerical vector.

Table 1. Numerical vectors versus string vectors

	Numerical Vector	**String Vector**
Element	Numerical value	String
Similarity Measure	Inner products, Euclidean distance	Semantics similarity
Attributes	Words	Property of words

The differences between string vectors and bags of words are illustrated in table 2. Both types of structured data have strings as their elements. As the similarity measure, cardinality of intersection of two bags of words is used while the average semantic similarity is used in string vectors. A bag of words is defined as an unordered infinite set of words, while a string vector is defined as an ordered finite set of words. Although a bag of words and a string vector look similar as each other, they should be distinguished from each other, based on table 2.

Table 2. Bag of words versus string vectors

	Numerical Vector	**String Vector**
Element	String	
Similarity measure	Number of shared words	Semantics similarity
Set	Unordered infinite set	Ordered finite set

We use an inverted index is used as the basis for the operation on string vectors as expressed in equation (4). An inverted index is defined as a list of words each of which is linked with a list of documents including it. A list of words is implemented with a hash table, while a list of documents which including a word is implemented with an array. A semantic similarity between two words is computed based on a number of documents where both words are collocated with each other. The more documents including both words, the higher semantic similarity between them is. From the inverted index, two lists of document identifiers corresponding to the two words are retrieved. The intersection is taken from the two lists of document identifiers as a list of documents including both words. In the next section we define the Semantic similarity.

c) Semantic Similarity

The proposed text categorizer needs to compute the similarity between two string vectors and update a weight vector. Weight vectors are updated by substituting its elements by inter-words (see [31]). So it is necessary to define operations on string vectors. This section will describe two operators necessary for training. A similarity matrix should be built from the given collection of texts before defining two operators on string vectors. Each entry in the matrix indicates the semantic similarity between two words based on their collocation and weights in a text.

The similarity matrix is defined word by word from the given collection of texts [33]. It is expressed by the symmetry function and square matrix shown in equation (2).

$$S = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1N} \\ s_{21} & s_{22} & \cdots & s_{2N} \\ \cdots & \cdots & \cdots & s_{3N} \\ s_{N1} & s_{N2} & \cdots & s_{NN} \end{bmatrix}$$

$$s_{ij} = s_{ji} = sim(t_i, t_j) \qquad (2)$$

An element, s_{ij}, of the similarity matrix, *S*, indicates the similarity between two words, t_i and t_j. It is computed by equation (3) (see [32]).

$$s_{ij} = \frac{\sum_{d \in D_i \cap D_j} (w_{d_i} + w_{d_j})}{\sum_{d \in D_i} w_{d_i} + \sum_{d \in D_j} w_{d_j}} \qquad (3)$$

where D_i is the set of documents including the word, t_i, D_j is the set of documents including the word t_j , w_{d_i} is the weight of the word, t_i, in document, *d* and w_{d_j} is the weight of the word, t_j, in document, *d*. As illustrated in the equation (3), the similarity between two words is based on their collocation in same document.

If a similarity matrix is built from the corpus with equation (2), we can compute the similarity between two string vectors, denoted by $x_i = [t_{i1}, t_{i2}, \ldots, t_{id}]$ and $x_j = [t_{j1}, t_{j2}, \ldots, t_{jd}]$. The similarity $sim(t_{ik}, t_{jk})$ between two words, t_{ik} and t_{jk} is indicated by the entry, s_{ij} of the row and column corresponding to such words or its reverse, s_{ij} in the similarity matrix. The similarity between two string vectors x_i and x_j denoted by $sim(x_i, x_j)$ is computed by equation (4).

$$sim(x_i, x_j) = \frac{1}{d} \sum_{k=1}^{d} sim(t_{ik}, t_{jk}) \qquad (4)$$

Given two words t_i and t_j. An inter-word *tk* is a word presenting a higher similarity to both t_i and t_j than the similarity between t_i and t_j. Such similarity is defined by the similarity matrix built from the given corpus. First, we find the similarity between two words from the similarity matrix to find inter-words between them. And we extract words with higher similarity with both of them from the similarity matrix. The set of inter-words between two words denoted by t_i and t_j, denoted by I_{ij} is expressed by equation (5).

$$\begin{aligned} I_{ij} &= I(t_i, t_j) = \\ &\{t_k \mid t_k \quad s(t_i, t_k) \geq s(t_i, t_j) \wedge s(t_j, t_k) \geq s(t_j, t_i) \end{aligned} \qquad (5)$$

IV. Competitive Neural Text Categorizer

This section describes the proposed competitive neural network, in detail, with respect to its architecture, training, classification, and properties.

The Self-Organizing Map (SOM), [28] proposed by Kohonen, provides a competitive learning principle of nodes such that adjacent nodes tend to have similar weight vectors. The SOM is an artificial neural network model that is well suited for mapping high- dimensional data into a two-dimensional representation space. The training process is based on the weight vector adoption with respect to the input vectors. The SOM has shown to be a highly effective tool for data visualization in a broad spectrum of application domains. Especially the utilization of the SOM for information retrieval purposes in large free form document collections has gained wide interest in the last few years. The general idea is to display the contents of a document library by representing similar documents in similar regions of the map. Without knowledge of the type of and the organization of the documents it is difficult to get satisfying results without multiple training runs using different parameter settings, which obviously is extremely time consuming given the high-dimensional data representation.

In contrast to another traditional neural network model [30], the main characteristics of SOM are two-fold, namely dimension reduction and topology preservation. Using SOM, a high-dimensional data space will be mapped to some low-dimensional space [27]. SOMs have recently been used to archive over 7 million documents [26]. Not only have SOMs been shown to scale up to very large document collections, these maps also allow for a novel mode of navigating through a large collection of text documents. As we mentioned above, the entire text collection is presented to a user as a two-dimensional map, where each node in the map is associated to a set of documents that are likely to be composed of similar terms and phrases. In addition to the classification of documents at the node level there is also classification of nodes. That is, the closer two nodes are in the map, the more similar are their associated documents.

While many studies have been devoted to automatic document clustering, our work is based on the rule learning approach. This approach generates a set of classification rules from labeled (pre-classified) training data. The greatest advantage of using rules is its comprehensibility. Rules are relatively easy to understand and modify. Thus, it is particularly helpful for end users to organize personalized URL repositories (bookmarks or hotlists).

As a baseline algorithm, we used the rule learning algorithm RIPPER (Repeated Incremental Pruning to Produce Error Reduction) [1],[2]. RIPPER is an efficient, noise-tolerant rule learning algorithm based on the separate-and-conquer strategy, and its algorithm is summarized as follows. The training data is partitioned into two subsets, a growing set and a pruning set. Using these two subsets, RIPPER builds up a rule set by repeatedly adding rules to an empty rule set. The rule-growing algorithm begins with an empty conditions, and greedily adds conditions until the rule no longer makes incorrect prediction on the growing set. Here, each condition represents the appearance of a particular word *w* in a document *d*. Next, the learned rule is simplified by deleting conditions so as to improve performance of the rule on the pruning set. All examples covered by the formed rule are then removed from the training set and a new rule is learned in the same way until all examples are covered by the rule set.

An example of a rule set constructed by RIPPER is below (using Prolog-like notation):

Painting :- WORDS ~ "watercolor".
Painting :- WORDS ~ "art", WORDS ~ "museum".
Painting :- WORDS ~ "author", WORDS ~ "picture".

This rule set means that a document *d* is considered to be in the category "Painting" if and only if
(word "watercolor" appears in *d*) OR
(word "art" appears in *d* AND word "museum" appears in *d*) OR
(word "author" appears in *d* AND word "picture" appears in *d*).

That is, rule conditions checks whether a keyword (e.g. "watercolor", "art", "museum", etc.) appears in a document.

One weakness with the RIPPER algorithm is that it does not create a condition for a keyword which appears in more than two categories. To take a simple example, let us consider document categories: "Painting", "Photography" and "Sports". In the training data, the word "gallery" may occur frequently in categories "Painting" and "Photography". Thus, the following rules are never created because these rules contradict each other.
Painting :- WORDS ~ "gallery".
Photography :- WORDS ~ "gallery".

However, an appearance of the word "gallery" strongly indicates that the document is not the "Sports" category but it is the "Painting" category or the "Photography" category. In order to achieve high-precision document categorization, it is desirable to use as many keywords as possible.

To avoid the problem, we extended the RIPPER algorithm to automatically introduce hierarchical categories in a rule set. We describe how the extended algorithm works by taking a simple example. First, in the rule growing phase, a rule is grown by simply adding conditions using the growing set. This phase may create contradictory rules. Assume here that the following rule set is created:
Painting :- WORDS ~ "gallery".
Painting :- WORDS ~ "watercolor".
Photography :- WORDS ~ "gallery".
Photography :- WORDS ~ "photo".

In this rule set, the first and third rules are contradictory. Then, we examine the frequencies
Freq(gallery, Painting)
Freq(gallery, Photography)

that word "gallery" occurs in the "Painting" or "Photography" category. If these frequencies exceeds a predetermined threshold, a new category will be created for word "gallery".

In this way, we finally obtain the following rule set:
Arts :- CATEGORY ~ Painting.
Arts :- CATEGORY ~ Photography.
Arts :- WORDS ~ "gallery".
Painting :- WORDS ~ "watercolor".
Photography :- WORDS ~ "photo".

The new category "Arts" covers both of the category "Painting" and "Photography" and a document *d* is considered to be in the category "Arts" if the word "gallery" appears in *d*. (We use the category name "Arts" for the sake of convenience. In practice, category names are automatically generated by a program.) The following figure illustrates this rule set.

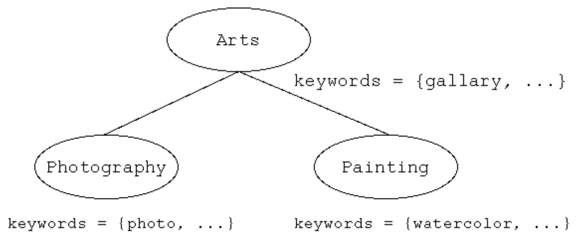

Fig.4 Hierarchical categories

V. Discussion & Conclusion

Motivated by an increased interest in automatically categorizing the World Wide Web documents, we proposed a new method for document categorization based on the RIPPER rule learning algorithm, and obtained encouraging results. As future research, we intend to elaborate the method by combining different categorization methods such as probabilistic classifiers.

This research used a full inverted index as the basis for the operation on string vectors, instead of a restricted sized similarity matrix. It was cheaper to build an inverted index from a corpus than a similarity matrix, as mentioned in section 1. In the previous attempt, a restricted sized similarity matrix was used as the basis for the operation on string vectors. Therefore, information loss from the similarity matrix degraded the performance of the modified version. This research addresses the information loss by using a full inverted index, instead of a restricted sized similarity matrix.

Note that there is trade-off between the two bases for the operation on string vectors. Although it is cheaper to build an inverted index from a corpus, note that it costs more time interactively for doing the operation expressed in equation (3). Let's the numbers of words, documents, and elements in each string vector be N ,M , and d . In using the inverted index, the complexity for doing the operation is $O(M^2d)$ in worst case, while in using the similarity matrix, the complexity is $O(d)$. When we try to compute semantic similarities of all possible pairs, the complexity is $O(N^2M^2d)$, whether we use a similarity matrix or an inverted index.

Experiments in the previous section showed that the proposed method works better than traditional networks, with respect to classification performance and classification time on binary or multi label classification corpus. This study implies that the representation of texts into string vectors is more appropriate than the representation into numerical vectors for text classification. The significance of this study is to address two main problems from the traditional representation of texts, by proposing a new unsupervised neural network using string vectors as its weight vectors and input vectors.

Other machine learning algorithms such as Naïve Bayes and back propagation are considered to be modified into their adaptable versions to string vectors. The operation may be insufficient for modifying other machine learning algorithms. For example, it requires the definition of a string vector which is representative of string vectors corresponding to a mean vector in numerical vectors for modifying a *k*-means algorithm into the adaptable version. Various operations on string vectors should be defined in a future research for modifying other machine learning algorithms.

On the other hand as mentioned in previous sections, the proposed method requires the construction of a similarity matrix to perform operations on string vectors. The experiment for the evaluation of classification time did not count the time for building the similarity matrix. Actually, it took very much time to build it.

To make our proposed method more practical, it is necessary to address the high cost of building the similarity matrix from the collection of texts. If all the words are used to build it other than stop words, it costs very much time to do that. We can consider three solutions to this problem for future research. The first solution consists of building a similarity matrix with only keywords from texts. The second solution consists of replacing the construction of the similarity matrix by word sense acquisition. The third solution consists of performing such operations on string vectors directly in the collection of texts. As a result, firstly, we need define the string vector for another neural network, and then address the problem of similarity matrix building with one of these solutions.

VI. References

[1] G. Attardi, S. Dimarco, D. Salvi, “Categorization by context”, Journal of Universal Computer Science, Vol.4, pp.719-736, 1998.

[2] N. Cristianini, J. Shawe-Taylor, Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press, 2000.

[3] H. Drucker, D. Wu, V. N. Vapnik, “Support vector machines for Spam categorization”, IEEE Transaction on Neural Networks, Vol.10, No.5, pp.1048-1054, 1999.

[4] R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification, John Wiley & Sons, Inc, 2001.

[5] V. I. Frants, J. Shapiro, V. G. Voiskunskii, Automated Information Retrieval: Theory and Methods, Academic Press, 1997.

[6] F. Gabriel Pui Cheong, J. Yu, “Text classification without negative examples revisit”, IEEE Transaction on Knowledge and Data Engineering, Vol.18, No.1, pp.6-20, 2006.

[7] S. Haykin, Neural Networks: Comprehensive Foundation, Macmillan College Publishing Company, 1994.

[8] M. Hearst, “Support vector machines”, IEEE Intelligent Systems, Vol.13, No.4, pp.18-28, 1998.

[9] P. Jackson, I. Mouliner, Natural Language Processing for Online Applications: Text Retrieval, Extraction and Categorization, John Benjamins Publishing Company, 2002.

[10] T. Joachims, “Text categorization with support vector machines: learning with many relevant features”, The Proceedings of 10th European Conference on Machine Learning, pp.143-151, 1998.

[11] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, C. Watkins, Text Classification with String Kernels, Journal of Machine Learning Research, Vol.2, No.2, pp.419-444, 2002.

[12] T. Martin, H. B. Hagan, H. Demuth, M. Beale, Neural Network Design, PWS Publishing Company, 1995.

[13] B. Massand, G. Linoff, D. Waltz, “Classifying news stories using memory based reasoning”, The Proceedings of 15th ACM International Conference on Research and Development in Information Retrieval, pp.59-65, 1992.

[14] T. M. Mitchell, Machine Learning, McGraw-Hill, 1997.

[15] D. Mladenic, M. Grobelink, “Feature selection for unbalanced class distribution and Naïve Bayes”, The Proceedings of International Conference on Machine Learning, pp.256-267, 1999.

[16] J. C. Platt, “Sequential minimal optimization: a fast algorithm for training support vector machines”, Technical Report MSR-TR-98-14, 1998.

[17] J. Rennie, “Improving multi-class text classification with support vector machine”, Master's thesis, Massachusetts Institute of Technology, 2001.

[18] M.E. Ruiz, P. Srinivasan, “Hierarchical text categorization using neural networks”, Information Retrieval, Vol.5, No.1, pp.87-118, 2002.

[19] F. Sebastiani, “Machine learning in automated text categorization”, ACM Computing Survey, Vol.34, No.1, pp.1-47, 2002.

[20] E. D. Wiener, “A neural network approach to topic spotting in text”, The Thesis of Master of University of Colorado, 1995.

[21] Y. Yang, “An evaluation of statistical approaches to text categorization”, Information Retrieval, Vol.1, No.1-2, pp.67-88, 1999.

[22] L. Man, Chew, T. Lim, “Supervised and traditional term weighting methods for automatic text categorization”, IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol.31, No.4, pp.721-735, 2010.

[23] V. Lertnattee, T. Theeramunkong, “Multidimensional text classification for drug information”, IEEE Transaction on Information Technology in Biomedicine, Vol.8, No.3, pp.306-312, 2008.

[24] http://www.research.att.com/~lewis/Reuters21578.html.

[25] D. Isa, L. Lee, “Text document preprocessing with the Bayes formula for classification using the support vector machine”, IEEE Transaction on Knowledge and Data Engineering, Vol.20, No.9, pp.1264-1272, 2008.

[26] The GHSOM Architecture and Training Process, http://www.ifs.tuwien.ac.at/~andi/ghsom/description.html

[27] M. P¨ oll¨ a, T. Honkela, and T. Kohonen, “Bibliography of self-organizing map (SOM) papers: 2002-2005 addendum”, Information and Computer Science, Helsinki University of Technology, Espoo, Finland, Tech. Rep. TKK-ICS-R24, 2009.

[28] T. Kohonen, Self-Organizing Maps. Berlin: Springer-Verlag, 1997.

[29] Y. Liu, X. Wang, and C. Wu, “ConSOM: A conceptional self-organizing map model for text clustering”, Neurocomputing, Vol. 71, No. 4-6, pp. 857–862, 2008.

[30] Chih-Wei Hsu and Chih-Jen Lin, “A comparison of methods for multi-class support vector machines”, IEEE Transactions on Neural Networks, Vol.13, No. 2, pp. 415–425, 2002.

[31] S. Wermter, “Neural Network Agents for Learning Semantic Text Classification”, Information Retrieval, Vol. 3, No. 2, pp.87-103, 2000.

[32] A. Budanitsky, G. Hirst, “Evaluating wordnet-based measures of lexical semantic relatedness”, Computational Linguistics, Vol. 32, No. 1, pp. 13-47, 2006.

[33] C. Yu, B. Cui, S. Wang, J. Su, “Efficient index-based knn join processing for high-dimensional data”, Information Software Technol., Vol. 49, No. 4, pp. 332–344, 2007.

AUTHORS

First Author – Vahid Behravesh, Department of Electrical Engineering, Islamic Azad University, Bardaskan Branch, Bardaskan, Iran, Email: vahidbehravesh@yahoo.com.

Second Author – S.M.R. Farshchi, Department of Electrical Engineering, Islamic Azad University, Bardaskan Branch, Bardaskan, Iran, Email: shiveex@gmail.com.