Two-level self-organizing approach to text classification

Ioannis Triantafyllou\textsuperscript{1,2}, Iason Demiros\textsuperscript{1,2}, Stelios Piperidis\textsuperscript{1,2}

\textsuperscript{1} Institute for Language and Speech Processing, \textsuperscript{2} National Technical University of Athens
Artemidos 6 & Epidavrou, 151 25, Athens, Greece, tel: +301 6875300, fax: +301 6854270
\{yiannis, iason, spip\}@ilsp.gr

Abstract

Several text classification methods have been adopted and extensively tested in the past, yielding promising results. Related research has concentrated on the calibration of the main attributes of each method. However, no essential effort has been made to improve the discrimination capacity of the learning space, prior to classification, using text-data mining techniques. We propose a two level text classification model. The first level consists in clustering the classes using a self-organizing map. Subsequently, a text classifier completes the task, by surveying a reduced space, but with enhanced discrimination capacity.

1. Introduction

Classification comprises the kernel mechanism of any document retrieval, routing or filtering model. That is, given a set of two or more classes, an IR system assigns each document to the appropriate class(es), as well as estimating a degree for each particular assignment. In this respect, overall performance of any IR system heavily depends on the efficiency of the underlying classification model. A number of text classification approaches have been presented and extensively tested in the past. The most common ones are based on k-nearest-neighbor algorithms (Masand \textit{et al.} 92; Larkey & Croft 96), linear text classification models (Lewis \textit{et al.} 96; Dagan \textit{et al.} 97), Bayesian independence classifiers (Lewis 92; McCallum & Nigam 98), relevance feedback techniques (Masand \textit{et al.} 92; Larkey & Croft 96) as well as rule-induction algorithms borrowed from machine learning, such as decision trees (Apté \textit{et al.} 94). Extensive evaluation of these techniques reported scores of approximately 75-80\% on a break-even points comparison (Dagan \textit{et al.} 97).

However, all methods presented so far are applied directly to the training space without any attempt to examine and possibly improve its discrimination capacity. We argue that by clustering training sequences we can significantly improve discrimination capacity, thus improving overall performance.

Self-Organizing Maps (SOMs) have been proposed as the main clustering model. A project that aims at constructing methods for exploring full-text document collections (Honkela \textit{et al.} 96; Kohonen 98; Merkl & Rauber 99; Kohonen \textit{et al.} 00) sprung from the suggestion of using SOMs as a preprocessing stage for encoding documents. These maps are used to automatically cluster documents according to the features that they contain. Documents are organized, during a preprocessing stage, based on a map, in such a way that similar documents are projected into nearby locations. This type of ordering facilitates the extraction of clusters by using intuitive neighborhood relations.

2. System Architecture

The overall architecture of the proposed text classifier is divided into two levels:

At the first level a SOM has been trained aiming at the organization of the entire document collection into smaller clusters. Clustering is based on the classes of the documents rather than the documents themselves. In this respect, clusters contain classes with related class-labels. Each new document is first filtered by the trained SOM and assigned to a specific cluster of classes.

Each cluster is associated with a respective traditional classifier trained over this particular cluster. Since each traditional classifier deals with only a part of the training space, more specifically the number of classes belonging to the respective cluster, discrimination capacity is improved.
As a traditional classifier we can use any of the models mentioned earlier in section 1 (KNN, Linear, Bayesian, decision trees, ...). The proposed framework was evaluated using two of the previously mentioned traditional models, namely a KNN and a linear classifier.

3. Self-Organizing Map

Inspired by the possibility that some high level organization in the brain may be created during learning through self-organization, and that “the representation of knowledge in a particular category of things in general might assume the form of a feature map that is geometrically organized over the corresponding piece of the brain” (Kohonen 89), Kohonen proposed a self-organizing learning algorithm, the Self-Organizing Map. The SOM is a method for producing ordered ow-dimensional (usually two-dimensional) representations of an input data space. Typically such input data is complex and high-dimensional with data elements being related to each other in a nonlinear fashion.

3.1. Properties

Two main properties of the SOM, indicating that it would constitute a very efficient clustering model, are (Haykin 99):

1. Approximation of the Input Space. The produced map represented by the set of synaptic weight vectors provides a good approximation to the input space.
2. Topological Ordering. The produced map computed by the SOM algorithm is topologically ordered in the sense that the spatial location of a neuron in the lattice corresponds to a particular domain or cluster of input patterns.

3.2. Training Phase: Algorithm

The algorithm responsible for the formation of the SOM starts by randomly initializing the synaptic weights in the network. Three essential processes run iteratively until the map converges:

1. Competition. For each input pattern the neurons in the network compute their respective values of a discriminant function (inner product) and the neuron with the largest value is declared winner.
2. Cooperation. The winning neuron determines the spatial location of a topological neighborhood of excited neurons.
3. Synaptic Adaptation. The last mechanism enables the excited neurons to increase their individual values in relation to the input pattern through suitable adjustments.
Let $n$ denote the dimension of the input space, $x(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^T$ denote the input vector selected at time $t$, $m$ denote the number of total neurons in the produced map and $w_j(t) = [w_{j1}(t), w_{j2}(t), \ldots, w_{jm}(t)]^T$ denote the weights for node $j$ at time $t$. The winning node $s$ is selected so that,

$$||x(t) - w_s(t)|| = \min_{j=1,2,\ldots,m} ||x(t) - w_j(t)||$$

After finding the winning node, adaptation of the weights of the nodes within a defined neighborhood is performed as shown,

$$w_j(t+1) = w_j(t) + a(t) [x_i(t) - w_j(t)], i=1,2,\ldots,n$$

where $a(t)$ is a gain term. The $a(t)$ is selected to guarantee that the SOM converges by introducing two control mechanisms. The first one is that $a(t)$ decreases in time and converges to 0 allowing the SOM to converge. The second one is that, $a(t)$ implicitly defines the neighborhood area of the winning node. A gain term including both of the previous mechanisms (Haykin 99), is adopted:

$$a(t) = A_1 \times \exp(-t/A_2) \times \exp(-\frac{||x(t) - r_j||^2}{2\sigma^2(t)})$$

where $\sigma(t) = A_3 \times \exp(-t/A_4)$, $r_j$ and $r_s$ define the position of excited neuron $j$ and the position of winning neuron $s$ respectively, and finally $A_1,A_2,A_3,A_4 \in \mathbb{R}$.

### 3.3. Producing Clusters of Classes

After constructing the map, nodes are labeled and then clusters are composed based on these labels. The labeling process, called “simulated electrode penetration mapping” (Haykin 99; Lin et al. 91), consists in finding for each neuron the most related input vector (the input vector for which the neuron produced the best response) and assigning the respective label to this neuron. In order to further assist the clustering process we assign two labels to each neuron, the labels of the two most related input vectors, which we call basic labels. In addition, supplementary labels are assigned to the neurons as follows: for each input vector, the neuron with the strongest response is assigned to the respective input label. Some neurons are left without supplementary labels.

The clustering process consists of three subsequent phases. At the first phase, areas are spotted based on the first basic label. Area continuity is ensured by the cooperative behavior of neighboring neurons. Then, each area is enriched with all second basic and supplementary labels. Finally, any adjacent areas one being a subset of the other are merged, iteratively, until no other merging is possible. Overlapping between final clusters is allowed, meaning that certain labels can appear in more than one clusters.

### 3.4. Supervised Calibration

The map produced by the SOM reflects variations in the statistical distribution of the input space: regions of the input space from which sample vectors are drawn with high probability of occurrence are mapped onto larger domains of the output space, and therefore with better resolution than regions from which sample vectors are drawn with a low probability of occurrence (Haykin 99). This property implies that high density input regions, will be represented by larger areas in the produced map than low density regions.

To overcome this problem we calibrate the produced map with a mistake-driven supervised learning technique:

1. Let $x_i$ denote the input vector at time $t$ while $w_s$ denotes the weight vector of the winning node $s$. In addition, let $C_s$ denote the set of classes that $x_i$ belongs to, and $C_w$ denote the set of classes that $w_s$ is marked with.

2. If $C_s \subseteq C_w$ do nothing, otherwise update the weight vectors of the winning node and all its direct-neighbors (up to six neighbors), as follows:

$$w_s(t+1) = w_s(t) + I(n) [x_i(t) - w_s(t)]$$

where $j$ is direct-neighbors of $s$.

**Figure 5.** A clustering example. Labels are: 1: grain, 2: sugar, 3: coconut, 4: cocoa, 5: coffee.
The learning constant \( l(n) \) decreases linearly with the number of iterations \( n \). An iteration is completed when all the input vectors are fed to the SOM. The learning constant \( l(n) \) must take diminutive positive values and converges to 0 very rapidly, reducing the number of iterations.

After calibration, the new map is re-labeled and re-clustered as described in section 3.3.

### 4. Traditional text classifiers

#### 4.1. KNN classifier

The most commonly used learning technique is the k-nearest neighbor (KNN). Let \( \mathbf{x} \) denote the input vector, \( n \) denote the dimension of the input space and \( x_1, x_2, \ldots, x_k \) denote its k-nearest neighbors. Nearest neighbors are defined in terms of the standard Euclidean distance,

\[
d(\mathbf{x}, \mathbf{x}_i) = \sqrt{\sum_{d=1}^{n} (x^d - x_i^d)^2}
\]

If \( y \) is the function that maps each known vector to a specific class, then \( \hat{y} \) is the estimate function of \( y \).

One obvious refinement to the KNN algorithm is the distance-weighted KNN algorithm,

\[
\hat{y}(\mathbf{x}) = \text{argmax}_{c \in C} \sum_{i=1}^{k} w_i \delta(c, y(x_i))
\]

where \( \delta(a,b) = 1 \) if \( a=b \), otherwise \( \delta(a,b) = 0 \),

\[
w_i = 1/d(\mathbf{x}, \mathbf{x}_i)^2,
\]

and \( C \) is the set of all classes.

The adopted text classification model was based on the distance-weighted KNN algorithm.

One practical issue in applying KNN algorithms is that the distance between instances is calculated based on all their attributes. As less attributes are relevant to the classification process, the more misleading assignments of class labels occur. This difficulty, which arises when many irrelevant attributes are present, is sometimes referred to as the curse of dimensionality (Mitchell 97).

One way to overcome this problem is rejecting as many irrelevant attributes as possible. This is achieved by clustering similar instances.

#### 4.2. Linear classifier

A linear text classifier represents a class \( c \) as a weight vector

\[
\mathbf{w}(c) = [w_{f_1}(c), w_{f_2}(c), \ldots, w_{f_n}(c)]^T,
\]

where \( f_1, f_2, \ldots, f_n \) is the set of all features and \( n \) is the size of this set (the dimension of the input space). Decision is made through evaluating the score of a document \( d \) with every class \( c \), by computing the dot product:

\[
F_c(d) = \mathbf{s}(d) \cdot \mathbf{w}(c),
\]

where \( \mathbf{s}(d) = [s_{f_1}(d), s_{f_2}(d), \ldots, s_{f_n}(d)]^T \) is the strength of each feature in document \( d \).

Multiplicative weight-updating algorithms such as Winnow (Littlestone 88) have been studied extensively in the theoretical learning literature. They perform exceptionally well in domains with very high dimensionality and particularly in the presence of irrelevant attributes, noise (Dagan et al. 97).

During the training phase of multiplicative weight-updating algorithms, adaptation of weight vector is mistake-driven. If the algorithm predicts 0 (no assignment in this class) and the correct value is 1, then we have a positive example and the weight vector is promoted: multiplied by a factor \( a \), where \( a>1 \). If the algorithm predicts 1 (assignment in this class) and the correct value is 0, then we have a negative example and the weight vector is demoted: multiplied by a factor \( b \), where \( 0<b<1 \).

The adopted linear classifier is the one referred to as Balanced Winnow. In this case the algorithm keeps two weight vectors \( \mathbf{w}^+, \mathbf{w}^- \). The overall weight vector is then computed as the difference between these two vectors: \( \mathbf{w} = \mathbf{w}^+ - \mathbf{w}^- \).

During training, a positive example alerts promotion to the positive weight vector and demotion to the negative weight vector, while a negative example alerts demotion to the positive weight vector and promotion to the negative weight vector.

### 5. Feature Selection

Special consideration has been given to the selection of the feature space, a crucial aspect in the performance of any text classification model as well as of the results of the SOM training phase. Any n-word in the training corpus constitutes a candidate feature. Functional words are excluded based on stop-lists.

#### 5.1. Average Mutual Information (MI)

Feature selection consists in reducing the vocabulary size of the training corpus by selecting n-words with the highest average mutual information over the set of values of the class variable. Average mutual information is the difference between the entropy of the class variable, \( H(C) \), and the entropy of the class variable conditioned on the presence or absence of the feature (word or phrase) variable, \( H(C|F) \). This method has been adopted by several researchers (Yang & Pederson 97; Joachims 97; McCallum & Nigam 98).
Let \( C \) denote the random variable over all classes and \( F \) the random variable over the presence or absence of feature \( f \) in a document, the Average Mutual Information is given by the following formula:

\[
I(C;F) = H(C) - H(C|F) = \sum_{c \in \{0,1\}} P(c \mid f) \log \left( \frac{P(c \mid f)}{P(c)} \right)
\]

where, \( P(c) \) is the number of documents with class label \( c \) divided by the total number of documents, \( P(f) \) is the number of documents containing (or not) the feature \( f \) divided by the total number of documents and \( P(c,f) \) is the number of documents with class label \( c \) that also contain (or not) feature \( f \), divided by the total number of documents.

The most dominant features are selected based on their deviation from the highest scoring feature.

### 5.2. TF.IDF

Feature selection is also performed based on the TF.IDF measure (Lewis et al. 96; Joachims 97). In our case we have classes rather than documents, so TF is the frequency of feature \( f \) within a class \( c \), and IDF is the logarithm of \( N/N_f \), where \( N \) is the total number of classes and \( N_f \) is the number of classes containing the feature \( f \).

The selected features are the most dominant ones in each class. Furthermore, the number of selected features for each class is related to the frequency of the class.

### 5.3. Word Conflator

Feature selection is also supported by a word conflator tool. The tool aims at capturing morphological variations of words located in the document collection, through a process called “conflation” (Frakes 84). Word conflations are performed at search time allowing for full form information to be kept in the training space. The scoring mechanism is based on a dynamic programming framework, especially designed to assign higher scores to morphological variations of the same root form.

### 5.4. Proposed Selection

By evaluating the previously described methods we have concluded that the MI measure outperforms the TF.IDF measure. Furthermore, the final selection of dominant features per class, was based on a combination of the previous measures, namely, the product of MI and TF.IDF. The combined measure improved the results of the SOM training phase as well as the results of the KNN classification model. Once again, as in the case of the TF.IDF measure, the selected features are the most dominant ones in each class and the number of selected features per class is also related to the frequency of the class.

#### 5.5. Feature Repetition

It is important to consider the frequency of a feature when determining its strength. There are cases where a feature is more indicative to the relevance of the document when it appears several times. However, this is not always true since long documents usually introduce a lot of noise. We experimented with three alternatives (Dagan et al. 97) concerning the strength of selected features: the boolean appearance in the document (0 or 1), the actual value of the feature frequency, and the square root of the feature frequency, and found that the last measure outperforms the others.

### 6. Evaluation

To provide an objective basis of comparison, we used the document collection of the Reuters newswire of 1987, properly identified as Reuters-22173 (Apté et al. 94), but hereafter referred to as Reuters. We chose to experiment with 3965 documents as training cases and 1056 documents as test cases. There were 57 class labels (topics) of interest, occurring more than twice in the training data. Stories with no class labels were left out of the process.

The first level consisted in training the self-organizing map. The number of neurons (\( N_n \)) we used was related to the average size of the classes contained in the training set, \( N_n \approx N_c \frac{\sqrt{D}}{N_c} \), where \( N_c \) is the total number of classes and \( D \) is the total number of documents in the training set. Based on our training set we estimated a value of \( N_n = 486 \approx 475.399 \).

The feature selection process was evaluated based on a subset of 970 documents of the original training set. The results of the produced SOM are presented in the table below (Figure 6).

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>MI</td>
<td>86.89 %</td>
</tr>
<tr>
<td>TF.IDF</td>
<td>73.15 %</td>
</tr>
<tr>
<td>MI × TF.IDF</td>
<td>89.53 %</td>
</tr>
</tbody>
</table>

**Figure 6.** Estimated method precision (recall 85%)
Feature selection was based on the combined MI × TF.IDF measure. The dimension of the feature space is an important parameter of the training phase of the SOM. Tests led to a 420-dimensional feature space. The dominant features per class were related to the frequency of each class. The following diagram (Figure 7) depicts the relation between the precision of the produced map and the size of the respective feature space.

We compared the results of a typical classification process and the results of a classification process filtered by the SOM, for both traditional classifiers. The results of the KNN classifier were significantly improved while the linear classifier showed a slight but noticeable improvement (Figures 8,9).

In the case of the KNN classifier the feature space was based on the combined MI × TF.IDF measure technique. The most dominant features per class were selected. The set of classes utilized for classification purposes in the typical classification process was the whole set, while in the case of the SOM filtering process, the utilized set of classes was only the subset contained in the activated cluster of the SOM.

In the case of the linear classifier the whole set of possible features (words and phrases occurring a reasonable number of times) of the entire document collection was selected, while in the case of the SOM filtering, only the set of possible features of documents contained in the activated cluster of classes was selected.

The clustering of the produced SOM resulted in 17 clusters of classes, therefore 17 KNN and 17 linear classification models.

7. Conclusions

We have demonstrated that clustering of the training space prior to classification can improve overall classification accuracy. Special emphasis must be given to the accuracy of the clustering process, since miscalculations during the first level of the proposed classification model, significantly affect the results of the second level. Soft clustering can ensure high rates of accuracy. Working within clusters rather than the whole set of classes offers one main advantage: the feature space is limited to a related subspace with increased discrimination capacity. Moreover, this feature subspace can be further improved by combining (MI × TF.IDF) the best of both worlds: the global view of the training space provided by the average mutual information metric with the broadly used and well proved technique of TF.IDF.

8. Ongoing Work

Our current work focuses on evaluating the proposed architecture using other classification models as well as other training corpora.

Furthermore, improving feature selection taking into account statistical information as well as other types of text evidence (titles, cue-phrases) is still an open issue.
References


