Simone Albertini, Alessandro Zamberletti, Ignazio Gallo

Unsupervised feature learning for sentiment classification of short documents

Abstract

The rapid growth of Web information led to an increasing amount of user-generated content, such as customer reviews of products, forum posts and blogs. In this paper we face the task of assigning a sentiment polarity to user-generated short documents to determine whether each of them communicates a positive or negative judgment about a subject. The method we propose exploits a Growing Hierarchical Self-Organizing Map as feature learning algorithm to obtain a sparse encoding of the input data. The encoded documents are subsequently given as input to a Support Vector Machine classifier that assigns them a polarity label. Unlike other works on opinion mining, our model does not exploit a priori hypotheses involving special words, phrases or language constructs typical of certain domains. Using a dataset composed by customer reviews of products, our experimental results prove that the proposed method can overcome other state-of-the-art feature learning approaches.

1 Introduction

E-commerce has grown significantly over the past decade. As such, there has been a proliferation of reviews written by customers for different products and those reviews are of great value for the businesses as they convey a lot of information both about sellers and products e.g. the overall customers' satisfaction.

With *sentiment analysis* or *opinion mining* we refer to the task of assigning a sentiment polarity to text documents to determine whether the reviewer expressed a positive, neutral or negative judgment about a subject (Pang and Lee, 2008). This is an interesting and useful task that has been successfully applied to several different sources of information, e.g., movies (Zhuang et al., 2006) and product reviews (Hu and Liu, 2004; Popescu and Etzioni, 2005; Ding et al., 2008) to name a few.

Many works in literature (Kanayama and Nasukawa, 2006; Wen and Wu, 2011; Ku et al., 2011) manage to build lexicons of opinion-bearing words or phrases that can be used as dictionaries to obtain bag-of-words representations of the documents or to assign to each word some kind of prior information; different techniques are adopted to build those dictionaries and lexicons, e.g. the polarity of specific part-of-speech influenced by the context (Hatzivassiloglou and McKeown, 1997; Turney and Littman, 2002; Nakagawa et al., 2010). Some of these techniques involve heuristics, manual annotations (Das and Chen, 2001) or machine learning algorithms, in fact recent works use unsupervised (Maas et al., 2011; Turney and Littman, 2002) or semi-supervised (Socher et al., 2011) learning algorithms to generate proper vector-space representations for the documents. In general, machine learning is frequently employed to deal with the challenging problem of *sentiment analysis* (Pang and Vaithyanathan, 2002; Wilson et al., 2004; Glorot et al., 2011b).

One of the most promising approaches in machine learning is feature learning as it allows to learn expressive features for the documents directly from the raw data without manual annotations or hand-crafted heuristic rules. Feature learning algorithms aim to learn semantically rich features able to capture the recurrent characteristics of the raw data; on the opposite, hand crafted features are computed rather than learnt directly from the data: the algorithms to generate such features are fixed and do not generalize to different frameworks without modifications of the algorithm, which are time consuming and need expert knowledge. Feature learning manages to learn new spaces where it is possible to express the information in a way that enhance its peculiarities, thus facilitating any subsequent process of data analysis. Those feature learning algorithms are essential for building complex deep neural networks: subsequent layers of features are learnt from the raw data and are used to initialize the parameters of complex neural network architectures (Hinton and Salakhutdinov, 2006; Bengio, 2009) that have also been successfully employed in solving sentiment analysis tasks (Glorot et al., 2011b).

A valid alternative to deep architectures are feature learning algorithms in shallow settings, that is unsupervised algorithms like restricted boltzmann machines or autoencoders (Socher et al., 2011) with single layers of latent variables having high cardinality. While shallow architectures are not as powerful as complex deep learning architectures, as they usually have far fewer parameters, they are simplier to configure and train and are well suited to solve problems in limited domains (Coates et al., 2011).

In this work, we use a novel feature learning algorithm in a shallow setting to classify short documents associated with product reviews by assigning them positive or negative polarities; we explore the possibility to solve such task without exploiting prior knowledge such as assumptions on the language, linguistic patterns or idioms. Our method is composed by three main phases: data encoding, feature learning and classification. First, we encode all the text documents in a vector space model using several bag-of-words representations; we employed five different encoding functions, one at a time, to guarantee that the good performances of our feature learning algorithm occur indepentently from the chosen data representation. Next, a novel unsupervised feature learning algorithm is trained with the encoded documents: they are clustered using a Growing Hierarchical Self-Organizing Map (GHSOM) (Rauber et al., 2002) and, relying on the clusterization result, we define a new sparse encoding for the input documents in a new vector space. A Support Vector Machine (SVM) classifier (Cortes and Vapnik, 1995) is finally trained with these feature vectors to assign the correct polarity labels to the documents. Our method overcomes the baseline accuracies obtained by the bag-of-words encodings without employing any features learning algorithm. Moreover, a comparison against other state-of-the-art shallow feature learning algorithms is provided.

2 Related Works

Several works in literature face the *sentiment analysis* task using machine learning algorithms. In the following paragraphs we introduce some of the models that we consider strictly related to our method.

Pang and Vaithyanathan (2002) adopt corpus based methods using machine learning techniques rather than relying on prior intuitions; their main goal is to identify opinion-bearing words. The documents are encoded using a standard bag-of-words framework and the sentiment classification task is treated as a binary topic-based categorization task. In their work, they prove that the SVM classification algorithm outperforms the others and good results can be achieved using unigrams as features with presence/absence binary values rather than term frequency, unlike what usually happens in topic-based categorization.

Maas et al. (2011) propose an unsupervised probabilistic model based on the Latent Dirichlet Allocation (David M. Blei and Jordan, 2003) to generate vector representations for the input documents. A supervised classifier is employed to cause semantically similar words to have similar representation in the same vector space. They argue that incorporating sentiment information in Vector Space Model approaches can lead to good overall results.

Socher et al. (2011) employ a semi-supervised recursive autoencoder to obtain a new vector representation for the documents. Such representation is used during the classification task, which is performed by softmax layers of neurons. Note that this approach does not employ any language specific sentiment lexicon nor bag-of-words representations.

Glorot et al. (2011b) build a deep neural network to learn new representations for the input vectors. The network uses rectified linear units and it is pre-trained by a stack of denoising autoencoders (Vincent et al., 2008). The data cases are encoded using a binary presence/absence vector for each term in the dictionary. The network is used to map each input vector into another feature space in which each data case is finally classified using a linear Support Vector Machine. Despite the fact that this framework is applied to domain adaptation, its pipeline is essentially identical to ours; however, we use a shallow model (the GHSOM) instead of a deep architecture and we perform our experiments using both linear and non-linear classifiers.

3 Proposed Model

A detailed description of the proposed method is given in the following paragraphs. The whole training procedure is supervised: it consists of an unsupervised neural network for feature learning and a supervised classifier for document classification.

In Figure 1 we present an overview of the proposed method: it is possible to observe that the raw documents received as input by our feature learning algorithm

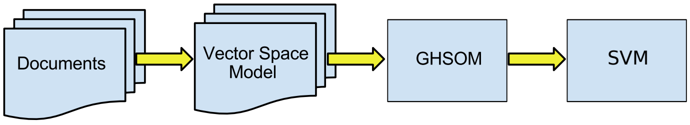

Figure 1: An overview of the proposed model. From left to right: the short documents are represented in a VSM, they are given as input to a GHSOM, the output of the GHSOM is exploited by an SVM classifier.

are represented in a Vector Space Model (VSM). The weight assigned to each term of the dictionary is computed using a weighting function w. In detail, given a set of documents D and a dictionary of terms T extracted from D, the weighting function $w_T : D \to X,\ X \subset [0,1]^{|T|}$ produces a vector representation $\vec{x} \in X$ of the document $d \in D$ in the space defined by the terms in the dictionary T. In Section 3.1 we discuss all the weighting functions applied in our experiments.

The vector space representation X for the set of input documents D is given as training data to a GHSOM that learns a new representation for the input data, as described in Section 3.2. The GHSOM generates maps that hierarchically represent the distribution of the training data. Note that, after the initial training phase, the topology of each map is fixed. At the end of the training phase, we assign a progressive numerical identifier to each k leaf units in the maps generated by the trained GHSOM and we define the learned k-dimensional feature space as F. Each vector $\vec{x} \in X$ used to train the GHSOM is mapped into a sparse feature vector by a function $feat : X \to F,\ F \subset [0,1]^k$. For each feature vector $\vec{f} \in F$ the following holds:

$$\vec{f}(i) = \begin{cases} 1 & \text{if } \vec{x} \text{ activates } u_i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where u_i is the i-th leaf unit of the GHSOM and $0 < i \leq k$. All the training vectors are mapped to obtain a set of corresponding feature vectors in F. This new set of feature vectors, along with their respective labels, constitutes the training data for an SVM classifier. Once the training phase ends, the classifier is able to assign a positive or negative label to each of its input vectors. In our experiments we evaluate the performances achieved by our model using both linear and radial basis function kernels. The linear kernel is used to evaluate the ability of the proposed model to generate a non-linear feature representation of the input vectors in a new space where the points of different classes are linearly separable. The radial basis function kernel is adopted to obtain a non-linear separating plane. In Algorithm 1 we summarize the previously described steps.

In the following subsections we introduce the weighting functions used to obtain the vector representations for the documents (Sec. 3.1), a description of the GHSOM (Sec.

Algorithm 1 Overview of the Proposed Model.

```
Training
    1. Build the dictionary of terms $T$ from the set of all documents $D$.
    2. Map all the training documents $d \in D$ in the VSM representation $w_T(d) = \vec{x},\ \vec{x} \in X$ using the dictionary $T$.
    3. Train a GHSOM with the vectors in $X$. Once the training phase ends, the number of maps generated by the GHSOM is $k$.
    4. Each $\vec{x} \in X$ is mapped in the $k$-dimensional feature space $F$ using the function $feat(\vec{x}) = \vec{f}$. Let $Y$ be the set of all the feature vectors computed in this way.
    5. Train a SVM classifier using the feature vectors in $Y$ along with their respective labels.
Prediction of a document $\bar{d}$
    1. Get the VSM representation $\vec{x} = w_T(\bar{d})$.
    2. Compute the corresponding feature vector $\vec{f} = feat(\vec{x})$ using the trained GHSOM.
    3. Predict the polarity of $\bar{d}$ by classifying the pattern $f$ using the trained SVM.
```

3.2) and other shallow unsupervised feature learning algorithms used for comparison (Sec. 3.3).

3.1 Short Texts Representation

Here we describe how the short documents are represented in a VSM using a bag-of-words approach. Let D be the set of all documents and V be a vector space whose number of dimensions is equal to the number of terms extracted from the corpus. Using an encoding function, we assign to each document $d \in D$ a vector $v_d \in V$, where $v_d(i) \in [0, 1]$ is the weight assigned to the i-th term of the dictionary for the document d. In our experiments we compare the results achieved by our model using five different encoding functions that are presented in the following paragraphs.

Binary Term Frequency. It produces a simple and sparse representation of a short document. Such representation lacks of representative power but acts as an information bottleneck when provided as input to a classifier. It has also been adopted by Glorot et al. (2011b). Given a term $t \in T$ and a document $d \in D$, Equation 2 is used to compute the value of each weight.

$$binary\_score(d,t) = \begin{cases} 1 & \text{if } t \in d \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

TF-IDF. It is a well-known method usually employed to compute the weights in a VSM. Using Equation 3, the weights assigned to a document $d \in D$ is proportional to the frequency of the term t in d (called tf) and it is inversely proportional to the frequency of t in the corpus D (called df).

$$TF \cdot IDF(d,t) = tf(d,t) \cdot \log(\frac{|D|}{df(D,t)}) \tag{3}$$

In our experiments we compare the results obtained using the TF-IDF approach applied both to unigrams and unigrams plus bigrams.

Specific against Generic and One against All. In the following Equation we present a generic way to assign a weight to each term t in a document d:

$$score(t,sc,gc) = 1 - \frac{1}{log_2(2 + \frac{F_{t,sc} \cdot D_{t,sc}}{F_{t,gc}})} \tag{4}$$

sc and gc are two sets of documents representing the specific corpus and the generic corpus respectively. We refer to the specific corpus as a set of documents that we want to consider different from the ones in the generic corpus, as such this formula intends to assign high scores to the terms of the documents in sc that are distinctive. $F_{t,sc}$ and $F_{t,gc}$ are the frequencies of the term t in sc and gc respectively. The number of documents in sc containing the term t is defined as $D_{t,sc}$.

The weight assigned to each term t in d by Equation 4 is proportional to $F_{t,sc}$ and inversely proportional to $F_{t,gc}$; when $t \notin gc$, $score(t,sc,gc) = 1$ and when $t \notin sc$, $score(t,sc,gc) = 0$. Therefore, the value of the *score* function is proportional to the ratio $\frac{F_{t,sc}}{F_{t,gc}}$ and it is close to 0 when t is very frequent in gc (thus t is not a domain-specific term).

Using Equation 4, two weighting strategies are defined: (i) the *Specific against Generic* (SaG), where sc is the set of positive-oriented documents and gc is the set of negative-oriented documents, (ii) the *One against All* (OaA), where sc is the set of all the documents of our domain (both positive-oriented and negative-oriented documents) and gc is a set of documents that do not belong to the domain and semantically unrelated to the ones in sc.

3.2 GHSOM

In this section we describe the Growing Hierarchical Self-Organizing Map (GHSOM) model (Rauber et al., 2002).

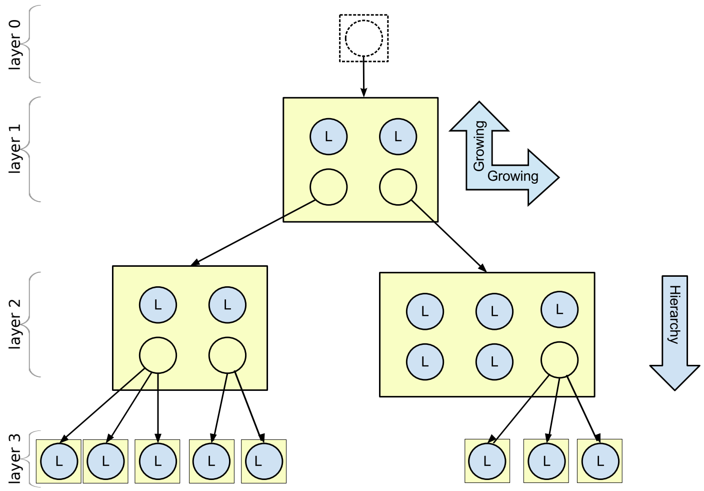

Figure 2: An example showing a GHSOM model. Each layer in the hierarchical structure is composed by several independent SOMs; the units with high mqe are expanded to form a new SOM in their subsequent layers; the units L that represent an homogeneous set of data do not require any expansion.

The GHSOM model is an evolution of the Self Organizing Map (SOM) (Kohonen, 2001). The latter is an unsupervised neural network composed by a two dimensional grid of neurons. A SOM aims to learn a quantized representation of the training vectors in their space by adjusting the weights associated to each neuron in order to fit the distribution of the input data. By doing so, a SOM operates a sort of clusterization of the input data, where the weight vectors assigned to each neuron are centroids.

In Figure 2 we show an example of GHSOM: it consists of a set of SOMs organized in a hierarchical structure built by an iterative procedure that starts from a single map and, when convenient, increases its size by adding rows and columns of neurons or by expanding a single neuron into another SOM. The criterion employed to modify the topology of a GHSOM is based on the quantization error and two parameters τ_1 and τ_2; these parameters adjust the propensity of the structure to grow in width (new rows/columns are added to the SOMs) and in depth (new SOMs are added) respectively. The mean quantization error mqe is a measure of the quality of each SOM; the greater the mqe, the higher the approximation level. The quantization error can be computed for a single unit and for a whole map using Equations 5 and 6 respectively.

$$mqe_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} \|m_i - x_j\| \tag{5}$$

$$mqe_M = \frac{1}{|M|} \sum_{i \in M} mqe_i \tag{6}$$

Let u_i be the neuron of a SOM M, m_i be the weight vector for u_i and C_i be the set of the input vectors associated to u_i.

The training process begins with the creation of an initial map constituted by only one unit whose weight vector is computed as the mean of all the training vectors. This map constitutes the layer 0 of the GHSOM and its mean quantization error is defined as mqe_0. In the subsequent layer, a new SOM M_1 is created and trained using the standard SOM training algorithm (Kohonen, 2001). After a fixed set of iterations, the mean square error mqe_{M_1} is computed and the unit u_e having the maximum square error is identified by computing $e = \text{argmax}_i \{mqe_i\}$. Depending both on the dissimilarity of its neighbours and τ_1, a new row or column of neurons is inserted at the coordinates of the unit u_e. M_1 is allowed to grow while the following condition holds:

$$mqe_{M_1} \geq (\tau_1 \cdot mqe_{M_0}) \tag{7}$$

When Equation 7 is no longer satisfied, the units of M_1 having high mqe may add a new SOM in the next layer of the GHSOM. The parameter τ_2 is used to control whether a unit is expanded in a new SOM. A unit $u_i \in M_1$ is subject to hierarchical expansion if $mqe_i \geq (\tau_2 \cdot mqe_0)$.

The previously described procedure is recursively repeated to iteratively expand the SOMs both in depth and width. Note that each map in a layer is trained using only the training vectors clustered by its parent unit. The training process ends when no further expansions are allowed.

3.3 Other feature learning algorithms

The following algorithms can be used to learn a mapping for the input data from a vector space to another and they are commonly used in literature to learn features from raw data in an unsupervised manner (Coates et al., 2011), as well as pre-train deep architectures (Glorot et al., 2011b; Hinton and Salakhutdinov, 2006; Hinton et al., 2006).

3.3.1 Restricted Boltzmann Machine

A Restricted Boltzmann Machine (RBM) (Smolensky, 1986) is an undirected graphical model composed by a visible layer and an hidden layer of neurons. The connections among the units form a bipartite graph as each neuron of a layer is only connected to the neurons in the other layer. The learned weights and biases can be used to obtain a feature mapping of the input vectors and this new representation may be provided to a classifier.

The training algorithm adopted in this work is the Contrastive Divergence (Hinton, 2002). This approximation of the gradient descent method has been employed using momentum and a L2 weight decay penalty. Both the neurons in the visible layer and the ones in the hidden layer use the logistic sigmoid activation function. We followed the guidelines available in literature to easily implement and use a RBM (Hinton, 2010).

3.3.2 Autoencoder

Let X be the set of training vectors, an autoencoder is a neural network composed by an encoder function $f(\cdot)$ and a decoder function $g(\cdot)$ such that, given $\vec{x} \in X$, the composition of the two functions gives the reconstructed input $g(f(\vec{x})) = r(\vec{x})$. The network is trained to minimize the reconstruction error using the backpropagation algorithm (Rumelhart et al., 1986). In this work we employ shallow autoencoders with three layers (input, code and output). The code layer is used to generate a new representation $f(\vec{x})$ for the input vectors that is provided as input to the classifier in the same way a vector is generated using a RBM.

In our experiments, we evaluate three different activation functions for the neurons in the code layer: logistic sigmoid, linear and rectified linear functions (Nair and Hinton, 2010). The rectified linear function units are reported to work well on sentiment analysis tasks (Glorot et al., 2011a). The momentum method has been used along with a L2 weight decay penalty for regularization.

A variant consists in training the autoencoder to remove noise from the input vectors: gaussian noise with zero mean is added to the training set so that $X + N(0, \sigma) = \tilde{X}$; hence, the network is trained to reconstruct the data in X from $\tilde{X}$. In our experiments we also use shallow denoising autoencoders with rectified linear units (Vincent et al., 2008).

4 Experiments

In this section we present the results obtained by performing an extensive experimental analysis of the proposed model. The main goal of our experiments is to determine: (i) how the parameters of our model affect its performances, (ii) the magnitude of the contribution of the GHSOM and the SVM in the proposed model, (iii) how the GHSOM performs in comparison with other state-of-the-art feature learning algorithms.

All our experiments are carried out using the Customer Review Dataset (Hu and Liu, 2004). The dataset is composed by several annotated reviews associated to 5 different products; each review consists of a set of short phrases whose lengths do not averagely exceed 30 words. All the phrases are independently annotated, thus they can be treated as short documents; moreover, their polarities can be predicted independently from the reviews they belong to. The Customer Review Dataset is composed by a total of 1095 positive and 663 negative phrases; in our experiments we balance it so that the positive and negative amounts of phrases are equal. This set of documents is splitted into a train set and a test set: 70% of the positive and negative phrases forms the training set while the remaining data cases form the test set. This split is performed once and then it is fixed and maintained during all the experiments.

Table 1: F-measure values obtained by different stages of the proposed model for the Customer Review Dataset. The columns labelled with *SVM linear* and *SVM rbf* show the baseline results; the column labelled with *GHSOM* shows the results obtained by directly using a GHSOM as classifier; the last two columns show the sparse feature vector classification (SFVC) results obtained by the SVM with a linear and a radial basis function kernels.

Encoding	**SVM linear**	**SVM rbf**	**GHSOM**	**SFVC (linear)**	**SFVC (rbf)**
Binary term frequency	0.52	0.56	0.75	0.81	0.87
TF-IDF unigrams	0.55	0.57	0.76	0.76	0.86
TF-IDF 2-grams	0.60	0.62	0.76	0.78	0.85
Specific against generic	0.54	0.76	0.76	0.76	0.88
One against all	0.56	0.56	0.77	0.81	0.90

All the meta-parameters both for the feature learning algorithms (such as τ_1 and τ_2 for the GHSOM or the parameters for the algorithms in Sec. 3.3) and the SVM are selected using 5-fold cross-validation on the training set.

We evaluate the performances achieved by the proposed model using the *F-measure* defined as in Equation 8.

$$F_1 = \frac{2 \cdot p \cdot r}{(p + r)} \tag{8}$$

where p and r represent precision and recall values respectively.

Baseline. In the first part of our experiments, we measure the classification results using the encodings described in Section 3.1; the vector representations generated by those encodings are classified by an SVM with both linear or radial basis function kernels, thus skipping the feature learning phase. As shown in Table 1, the results obtained using the linear and non-linear kernels are similar. That's because the vector space has a great dimensionality, therefore mapping the data into an higher dimensional non-linear space does not improve the classification performances.

Note that this first part of the experiments is crucial for the subsequent phases: the unsupervised feature learning algorithm aims to learn a new space to generate new feature vectors from those same input vectors. It is important to know whether the classification of the data in the new space can outperform the results obtained just by using a SVM with the same input vectors but without feature learning.

GHSOM analysis. In this second part of our experiments, we analyse the distribution of the documents in the clusters produced by a trained GHSOM.

Given a trained GHSOM, we assign a polarity to each of its leaf units. Let u_i be a leaf unit in the map M generated by an expansion of the unit u_{par} belonging to the previous layer. We define $P = P_{pos} \cup P_{neg}$ as the set of training vectors clustered by the unit u_i. The polarity assigned to u_i is computed as follows:

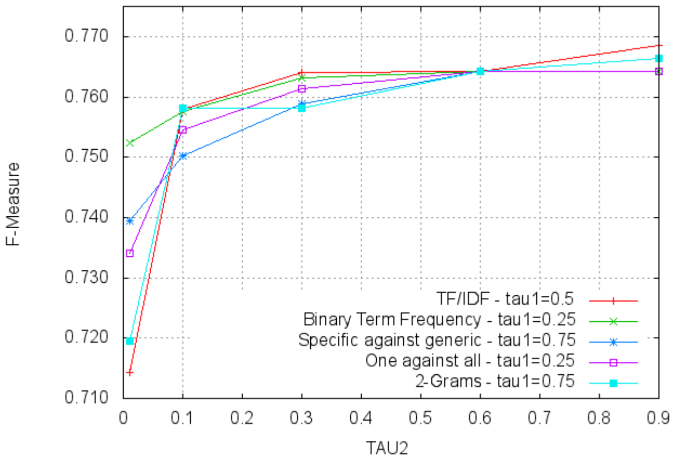

Figure 3: F-measure values achieved by a trained GHSOM for the Customer Review Dataset, while varying the parameter τ_2. For each of the five encodings of Sections 3.1 and 3.2, the optimal parameter τ_1 was found using a k-fold cross-validation technique with $k = 5$.

$$pol(u_i) = \begin{cases} pos & \text{if } |P_{pos}| > |P_{neg}| \\ neg & \text{if } |P_{neg}| > |P_{pos}| \\ pol(u_{par}) & \text{otherwise} \end{cases} \tag{9}$$

As previously stated, it is possible to exploit the GHSOM as a clustering algorithm: each leaf unit is a centroid in the input vectors space and each unseen document is assigned the polarity of its closest centroid. Given an unseen document $\bar{d}$, we compute its closest leaf unit $u_{\bar{d}}$ as described in Section 3.2 and its polarity as $pol(u_{\bar{d}})$. The results obtained by this simple clusterization algorithm are presented in Table 1; we observe a general improvement over the baseline classification results.

In Figure 3 we present the development of the *F-measure* F_1 while varying the parameter τ_2; the value assigned to τ_1 is determined using k-fold cross-validation as previously stated. Note that, as the GHSOM grows in depth, the classification results obtained using the 5 different encodings improve. We argue that, as the number of leaf units increases, the centroids in the vector space become more specialized and precise.

Sparse feature vectors classification. In our final experiments we measure the results obtained when the sparse feature vectors generated by the trained GHSOM are given as input to both a linear and a non-linear SVM classifiers. These feature vectors are the vectors produced by the $feat$ function in Section 3. The results are presented in Table 1 and they prove that: (i) the classification by a SVM of the feature vectors

Table 2: Comparison with other feature learning methods.

Method	full (linear)	full (rbm)
RBM	0.83	0.85
Autoencoder (linear)	0.68	0.70
Autoencoder (logistic)	0.71	0.71
Autoencoder (ReLu)	0.71	0.74
Denoising autoencoder	0.72	0.74
GHSOM	0.81	0.90

obtained using out feature learning algorithm always outperforms the baseline, (ii) the encoding generated by the GHSOM defines a vector space that is better (in terms of separability) than the ones defined by the encodings discussed in Section 3.1. Note that the vectors generated by the $feat$ function are not well linearly separable: in fact, a non-linear classifier trained using the sparse feature vectors generated by the GHSOM performs better than a linear one.

Comparison. We provide comparisons with the feature learning algorithms introduced in Section 3.3. We used those shallow feature learning algorithms in the same pipeline described in Figure 1 in place of our GHSOM based feature learning algorithm and no one was able to outperform our method. Table 2 shows the best results obtained by the algorithms while trying all the short text representation strategies listed in Section 3.1 and setting the values for all the meta parameters, such as the number of latent variables, using 5-fold cross-validation.

We tried four different settings for the autoencoders. The first one uses linear activation units in the code layer, as it is usually employed when working with text (Hinton and Salakhutdinov, 2006). Next, we tried autoencoders with logistic sigmoid activation units which is a standard non-linear activation function for learning features; it is usually employed as it is considered biologically plausible for learning. We also tried autoencoders with rectified linear activation functions (ReLu) in the code layer as a recent work (Glorot et al., 2011a) argues that rectified units may improve the quality of the learned features as they provide a natural way to produce a sparse representation since a lot of components are assigned values exactly equal to 0. Our results show that the autoencoder with ReLu obtains better results than the autoencoders with linear and logistic sigmoid units.

Finally, we trained a denoising autoencoder with ReLu; this kind of autoencoder is expected to learn better features as it cannot just copy the input to the code layer because it is corrupted by noise. The denoising autoencoder produced results in line with the ReLu autoencoder when using the radial basis function SVM. However, the vectors produced by the denoising autoencoder are easier to separate using a linear classifier, therefore we assess that this autoencoder is able to learn a feature space that allows to obtain better performances in a linear classification setting than the other autoencoders.

The RBM led to better results than the ones achieved by the autoencoders. More importantly, the vector representation produced by the RBM led to better classification results than the ones obtained by our method when using a linear classifier; this means that the RBM learns a feature space that is better than the one of our GHSOM-based algorithm when the subsequent classification task is performed in a linear setting. However, as shown in Tables 1 and 2, it is not good enough to let the non-linear SVM overcome the best classification performance obtained by the proposed model, which learns a space that produces very effective feature vectors when classifying in a non-linear setting.

5 Conclusion

The method presented in this work is able to generate a sparse encoding of short documents in an unsupervised manner, without using any prior knowledge related to the context of the problem. In our experiments we proved that a properly trained Growing Hierarchical Self-Organizing Map, used as clustering algorithm for feature learning and applied to different bag-of-word data representations, can provide robust results. Excellent performances can be achieved when the output of such model is provided as input to a Support Vector Machine classifier; thus, we argue the suitability of feature learning algorithms in the field of *sentiment analysis*. Our solution presents some interesting advantages: (i) it is language independent, (ii) it does not require any lexicon of opinion-bearing words nor idioms, (iii) it is domain independent, meaning that it may be applied to different contexts without further modifications. The comparison with other state-of-the-art unsupervised feature learning algorithms confirms the effectiveness of the proposed method: our feature learning model produces feature vectors that, once classified using a SVM classifier, lead to better performances compared to the state-of-the-art algorithms that are similar to ours in characteristics and complexity.

References

Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127.

Coates, A., Lee, H., and Ng, A. Y. (2011). An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Journal of Machine Learning Research*, 20(3):273–297.

Das, S. and Chen, M. (2001). Yahoo! for amazon: Extracting market sentiment from stock message boards. In *In Asia Pacific Finance Association Annual Confference (APFA)*.

David M. Blei, A. Y. N. and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Ding, X., Liu, B., and Yu, P. S. (2008). A holistic lexicon-based appraoch to opinion mining. In *Proceedings of First ACM International Conference on Web Search and Data Mining (WSDM)*.

Glorot, X., Bordes, A., and Bengio, Y. (2011a). Deep sparse rectifier neural networks. *Journal of Machine Learning Research*, 15:315–323.

Glorot, X., Bordes, A., and Bengio, Y. (2011b). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*.

Hatzivassiloglou, V. and McKeown, K. R. (1997). Predicting the semantic orientation of adjectives. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

Hinton, G. E. (2010). A practical guide to training restricted boltzmann machines. Technical report.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Hinton, G. E. and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

Kanayama, H. and Nasukawa, T. (2006). Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kohonen, T. (2001). *Self-Organizing Maps*.

Ku, L.-W., Huang, T.-H., and Chen, H.-H. (2011). Predicting opinion dependency relations for opinion analysis. In *Proceedings of 5th International Joint Conference on Natural Language Processing (IJCNLP)*.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 807–814.

Nakagawa, T., Inui, K., and Kurohashi, S. (2010). Dependency tree-based sentiment classification using crfs with hidden variables. In *Human Language Technologies (HLT)*.

Pang, B. and Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Pang, B. and Vaithyanathan, S. (2002). Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Popescu, A. M. and Etzioni, O. (2005). Extracting product features and opinions from reviews. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

Rauber, A., Merkl, D., and Dittenbach, M. (2002). The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks*, 13:1331–1341.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, pages 533–536.

Smolensky, P. (1986). Parallel distributed processing: explorations in the microstructure of cognition, vol. 1. chapter Information processing in dynamical systems: foundations of harmony theory, pages 194–281. MIT Press.

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., and Manning, C. D. (2011). Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Turney, P. D. and Littman, M. L. (2002). Unsupervised learning of semantic orientation from a hundred-billion-word corpus. Technical report, Technical Report EGB-1094, National Research Council Canada.

Vincent, P., Larochelleand, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*.

Wen, M. and Wu, Y. (2011). Mining the sentiment expectation of nouns using bootstrapping method. In *Proceedings of 5th International Joint Conference on Natural Language Processing (IJCNLP)*.

Wilson, T., Wiebe, J., and Hwa, R. (2004). Just how mad are you? finding strong and weak opinion clauses. In *Proceedings of the 19th national conference on Artifical intelligence (AAAI)*.

Zhuang, L., Jing, F., and Zhu, X.-Y. (2006). Movie review mining and summarization. In *Proceedings of the 15th ACM international Conference on Information and Knowledge Management (CIKM)*.