An Extension of the VSM Documents Representation using Word Embedding

Daniel MORARIU
“Lucian Blaga” University of Sibiu, Engineering Faculty, Sibiu, Romania
daniel.morariu@ulbsibiu.ro

Lucian VINȚAN
“Lucian Blaga” University of Sibiu, Engineering Faculty, Sibiu, Romania
lucian.vintan@ulbsibiu.ro

Radu CREȚULESCU
“Lucian Blaga” University of Sibiu, Engineering Faculty, Sibiu, Romania
radu.kretzulescu@ulbsibiu.ro

ABSTRACT

In this paper, we will present experiments that try to integrate the power of Word Embedding representation in real problems for documents classification. Word Embedding is a new tendency used in the natural language processing domain that tries to represent each word from the document in a vector format. This representation embeds the semantically context in that the word occurs more frequently. We include this new representation in a classical VSM document representation and evaluate it using a learning algorithm based on the Support Vector Machine. This new added information makes the classification to be more difficult because it increases the learning time and the memory needed. The obtained results are slightly weaker comparatively with the classical VSM document representation. By adding the WE representation to the classical VSM representation we want to improve the current educational paradigm for the computer science students which is generally limited to the VSM representation.

Keywords: Text Mining, Word Embedding, Classification, Document Representation (VSM), Computer Science Curricula

1 INTRODUCTION

Document classification has become a rather striking issue as the amount of information stored in electronic format is growing fast. It is becoming difficult to retrieve useful information from this huge amount of information. Automated classifying algorithms have been developed and tested in different contexts to get better results. Lately, the focus of automatic information retrieval is no longer on the classification algorithm; it has shifted on the improving of documents representation. The reason is simple; the better document representation augmented with more semantic information makes easier the work of classification algorithm. Unfortunately, the documents are structured to be understood by the humans not by the cars.

Early methods used for representing text documents as inputs for learning algorithms were based on frequency of the words (Vector-Space-Model - VSM), known in the literature as bag-of-words representation [Mitchell1997]. In this representation, it is counted if a word appeared or not appeared in that document, without keeping the information regarding the order of the words. Due to this representation, any semantic information transmitted by the order in which the words appear in the document is lost. Other used methods of document representation attempted to represent
documents as expressions, frequency vectors, to implicitly introduce some semantic information into the document representation. Representations that consider the orders of words typically require a great amount of memory for storing the documents representation, which makes a learning algorithm to be inefficient. In the last period, it is increasingly spoken about the representation of the words in the document as a vector depending on the context in which they appear, the so-called Word Embedding (WE) [Bengio2003], [Mikolov2013_1]. This new representation makes that the similar words belonging to a certain domain will have similar representations.

This paper presents a study of the influence of the new representation based on WE in order to improve the classification accuracy. More precisely, we augment the classical VSM representation by adding more information for each word using the WE representation of the word. For the evaluation of this enhanced representation we have used a Support Vector learning algorithm. This algorithm was also used in the evaluation of the VSM classical representation [Morariu2008]. As far as we know, in all previous experiments presented for the Word Embedding method the researchers have used as input only very small documents, working with a small number of words.

Also, the experiments described in this paper will be presented in our courses of Data Mining and Advanced Text Mining from the Computer Science and Computer Engineering study programs. The main idea is to present to the students another paradigm of document representation, other than the classical VSM. Thus, the students will be able to compare the current VSM documents representation with a paradigm enriched with supplementary semantic information. The new paradigm will not significantly change the information representation and how to be used for a classical classification algorithm. This will only introduce to the students a new way to look and work with the information. Thus, they will be able to study the influence of introducing additional semantic and syntactic information in the documents’ representation without significant changes in the document classification framework.

Section 2 contains the prerequisites for the work that we are presenting in this paper, the framework and methodology used for our experiments. In Sections 3 we present the main results of our experiments. Section 4 debates and concludes on the most important obtained results and proposes some further work, too.

2 EXPERIMENTAL FRAMEWORK

Starting from a set of text documents, in the first step we represented these documents in a vector format using the word frequency vectors representation (VSM). Each characteristic from this vector is called feature and it represents a word. Because such representation involves extraction of many distinct words (somewhere around 20,000 words in our experiments), in the next step we have selected only those features which are relevant for the document to obtain a smaller vector document representation. For the feature selection step, we used the Information Gain method, which is one of the most commonly used in this context.

The novelty of this article consists in the fact that in the document representation vector each word is augmented by its Word Embedding representation. Considering the idea presented by the authors in [Vintan2017] the document is represented as a vector of vectors (hyper-vector) in which for each document we have a vector representation where each word has its own vector representation given by the Word Embedding. In our experiments, each word was represented first by a 10-dimensional WE vector, then by a WE 5-dimensional vector. For obtaining the Word Embedding representation of a specific word we use the Continuous-Bag-of-Words (CBOW) with negative sampling training algorithm presented in [Mikolov2013_1] [Mikolov2013_2]. The package is called Gensim [Rehurek2010] and it is implemented in Python. This package contains the proper corpus for training the Word Embedding. The obtained Word Embedding model produced by this framework will be further used in our VSM representation module.
2.1 The used data sets

Our experiments presented in this article are performed on the Reuters-2000 collections [Reut00], which are newspapers articles published by Reuters Press in a compressed format. Due to the huge dimension of the database we will present here, results obtained using only a subset of this dataset. From all documents, we selected the documents for which the industry code value is equal to “System software”. We obtained 7083 files that are represented using 19038 features and 68 different topics. We represented documents as vectors of words, applying a stop-word filter (from a standard set of 510 stop-words) and extracting the stem of the word. From these 68 topics, we have eliminated those topics which are poorly or excessively represented. Thus, we eliminated those topics that contain less than 1% documents from all 7083 documents of the entire set. We also eliminated topics that contain more than 99% samples from the entire set, as being excessively represented. The elimination was necessary because with these topics we have the risk to use only a single decision function for classifying all documents ignoring the rest of the decision functions. After doing so we obtained 24 different topics and 7053 documents that were split randomly in a training set (4702 samples) and an evaluation set (2531 samples).

2.2 Document representation

Documents are typically represented as vectors in a features space. Each word in the vocabulary is represented as a separate dimension. The number of occurrences of a word in a document represents the value of the corresponding component in the document’s vector. This document representation results in a huge dimensionality of the feature space, which poses a major problem to text classification. The native feature space consists of the unique terms that occur into the documents, which can be tens or hundreds of thousands of terms for even a moderate-sized text collection. Due to the large dimensionality, much time and memory are needed for training a classifier on a large collection of documents.

Because there are many ways to define the feature-weight, we represent the input data in three classical different formats (called normalizations), and we try to analyze their influence on the classification accuracy. We use for normalization the Binary, the Nominal and the Cornell Smart normalizations presented in [Morariu2017].

2.3 Information Gain

Information Gain and Entropy [Mitchell1997] are functions of the probability distribution that underlie the process of communications. The entropy is a measure of uncertainty of a random variable. Based on entropy, as attribute effectiveness, a measure is defined in features selection, called “Information Gain”, and it represents the expected reduction in Entropy caused by partitioning the samples according to this attribute. The Information Gain of an attribute relative to a collection of samples S, is defined as:

\[
Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \cdot Entropy(S_v)
\]

where Values(A) is the set of all possible values for attribute A, and S_v is the subset of S for which attribute A has the value v.

Forman reported in [Forman2004] that Information Gain failed to produce good results on an industrial text classification problem, as Reuter’s database. He attributed this to the property of many feature scoring methods which ignore or remove features needed to discriminate difficult classes. Also, the information gain method favors attributes that have more distinct values than those with few distinct values.

2.4 Word Embedding

Word embedding is one of the most exciting areas of research in deep learning, although they were
originally introduced by Bengio, et al. [Bengio2003] more than a decade ago. The idea of distributed representations for symbols is even older [Hinton1986].

Word embedding refers to a recently developed family of language models supposed to learn linguistic and semantic features from natural language content by embedding the words (or composite language elements, such as phrases) in a dense, low-dimensional vector space, called embedding space. Basically, training such a model corresponds to learning a mapping from words to real-valued vectors in a vector space.

A Word Embedding $W : \text{words} \rightarrow \mathbb{R}^n$ is a parameterized function mapping words in some language to high-dimensional vectors (perhaps 50 to 500 dimensions). For example, we might find for the word cat in the WE representation:

$$W(\text{cat}) = (0.2, -0.4, 0.7, ...)$$

Learning to represent words as vectors is a form of feature learning (the central topic of the deep learning movement), as latent (hidden) linguistic and semantic features of the words are discovered (but remain innominate) from the training data which usually consists of massive amounts of unlabeled natural language text. The embedding is a position vector in a word space.

For representing Word Embedding we use Gensim [Rehurek2010] which started off as a collection of various Python scripts for the Czech Digital Mathematics Library, where it is used to generate a short list of the most similar articles to a given article (gensim = “generate similar”). Gensim is now, one of the most robust, efficient and hassle-free piece of software for performing unsupervised semantic modelling from plain texts. We used this framework in order to build the vector representation of words. The implemented model is proposed by Mikolov [Mikolov2013_1] and provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words. These representations can be subsequently used in many natural language processing applications and for further research.

2.5 Support Vector Machine

The Support Vector Machine (SVM) is a classification technique based on statistical learning theory [Nello2000], [Schoelkopf2002] that was applied with great success in many challenging non-linear classification problems and was successfully applied to large data sets.

The SVM algorithm finds a hyperplane that optimally splits the training set. The optimal hyperplane can be distinguished by the maximum margin of separation between all training points and the hyperplane. Looking at a two-dimensional problem the algorithm want to find a line that “best” separates points in the positive class from points in the negative class. The hyperplane is characterized by a decision function like:

$$f(x) = \text{sgn}\left( \langle w, \Phi(x) \rangle + b \right)$$

where $w$ is the weight vector, orthogonal to the hyperplane, “$b$” is a scalar that represents the margin of the hyperplane, “$x$” is the current sample tested, “$\Phi(x)$” is a function that transforms the input data into a higher dimensional feature space and $\langle \cdot, \cdot \rangle$ representing the dot product. $\text{sgn}$ is the signum function. If $w$ has unit length, then $\langle w, \Phi(x) \rangle$ is the length of $\Phi(x)$ along the direction of $w$. Generally, $w$ will be scaled by $\|w\|$. The training part the algorithm needs to find the normal vector “$w$” that leads to the largest “$b$” of the hyperplane.

3 EXPERIMENTAL OBTAINED RESULTS

In the Word Embedding representation used in our experiments a word is represented as a vector in the real numbers field having positive and negative values. In the classic VSM representation of documents, where each element in the vector represents the frequency of occurrence of the word in the current document, the values are just positive. That is why most data normalization formulas
work only for positive values. In the current approach, we want to add to the classic VSM representation the new Word Embedding representation for each word. The idea is that besides syntactic information represented by the normalized frequency of a certain word in the current document, we also add semantic information related to the context in which the word occurs frequently. Each word is no longer an axis in the orthogonal space of representation of documents, it is an own space of representation, following the idea presented by us in the article [Vintan2017].

In order to use all the normalization methods listed in the Data Representation section, we performed a linear transformation of the WE representation of each word from the real numbers set \( \mathbb{R} \) to the real positive numbers set \( \mathbb{R}^+ \) values by adding the minimum value existing in that vector to each element in the vector (note \( \text{min}_\text{value} \)) as

\[
\text{new}_\text{value} = \text{old}_\text{value} - \text{min}_\text{value} + 1
\]

We added a unit to each value for making the difference between the 0 value, that means in the VSM document representation that the word in the current document does not appear, and a value greater than 1, if it occurs. This transformation is useful for the Cornell-Smart normalization where the logarithm is used. For the Binary or Nominal normalization, this transformation is not necessary.

We performed the experiments in which we developed the linear transformation to the \( \mathbb{R}^+ \) for all three normalization methods. Also, we designed experiments in which we kept the representation of WE in the real numbers set without doing that transformation. Subsequently, for the representation of a document, starting from the VSM standard representation of that document, for each word we multiplied the corresponding elements in its WE representation vector with the frequency of its occurrence in the document (thus, a product between a scalar and a vector). After this step, we applied the proposed normalization formulas.

For the document representation in these experiments, we have used a vector with 1309 features. This vector dimension was obtained after representing all documents from the Reuters data set as vectors and after applying the Information Gain feature selection method. For this dimension in the VSM classical representation, we obtained the best results [Morariu2008] and we will compare the new representation with these results.

We present results obtained with dimensions 10 and 5 respectively, for the Word Embedding vector. This means that for WE 10 a document is represented in a dimension with 1309*10 features. We do not use higher dimensions because the dimension of documents representation increases exponentially, which leads to higher execution time and memory usage.

These new obtained results were then compared with the results obtained with a classical VSM representation of documents. The aim was only to see if some improvements occur if we include more information in the representation, especially because of the Word Embedding representation that inserts semantical information.

### 3.1 Results obtained using 10 elements in Word Embedding representation

For the classification algorithm, presented in Section 2.5, we used our implementation of the Support Vector Machine classifier with the Polynomial and Gaussian kernels. The formula of kernels and parameters were presented in [Morariu2007]. For the polynomial kernel, we used in our experiments five values of degree for each type of representation. For the Gaussian kernel, we performed our experiments with six different degrees for Binary and for Cornell-Smart representations.

In these experiments for each selected word we have used the embedding representation by length 10, note WE10. This means that for each word we introduce ten new dimensions. Thus, a document can be seen as one that is represented as a vector having 1309 features where each feature is represented by 10 different WE dimensions. In reality, we do not use this huge dimension in documents representation. We make all the computations on each dimension separately, without representing the documents in their huge dimension using sparse vectors representation.
In Table 1 are represented comparatively the results from the classification accuracy point of view obtained for a classical VSM representation with 1309 features and a representation using 1309 features, each of them having 10 different dimensions (with WE10).

### Table 1. Classification accuracy results for WE equal with 10, only positive domain

| Kernel Type | Kernel degree | Accuracy for 1309 features | Accuracy for 1309 feature and WE10 | Accuracy for 1309 features | Accuracy for 1309 feature and WE10 |
|-------------|---------------|----------------------------|------------------------------------|-----------------------------|------------------------------------|
| **Cornel SMART** | D1.0           | 80.99%                     | 79.33%                             | 81.45%                      | 79.63%                             |
|             | D2.0           | 87.11%                     | 86.52%                             | 86.64%                      | 58.61%                             |
|             | D3.0           | 86.51%                     | 77.12%                             | 85.79%                      | 68.06%                             |
|             | D4.0           | -                          | -                                  | 74.61%                      | 72.61%                             |
|             | D5.0           | -                          | -                                  | -                           | 72.22%                             |
| **RBF**     | C1.0           | 82.99%                     | 65.84%                             | 82.99%                      | 59.21%                             |
|             | C1.3           | 83.57%                     | 70.01%                             | 83.57%                      | 52.11%                             |
|             | C1.8           | 84.30%                     | 68.01%                             | 84.30%                      | 60.70%                             |
|             | C2.1           | 83.83%                     | 77.12%                             | 83.83%                      | 63.89%                             |
|             | C2.8           | 83.66%                     | 68.31%                             | 83.66%                      | 64.40%                             |
|             | C3.1           | 83.66%                     | 71.25%                             | 71.25%                      | 71.25%                             |

As it can be observed, the obtained results using the Word Embedding representation are close to the obtained results without WE but, however, are constantly weaker. Under no circumstances, the results could be improved. This can mean that by increasing the vector dimension we make the documents more difficult to be classified. Or maybe, increasing the vector we also increased the noise that is send to the learning algorithm making the learning process weaker.

These results were obtained after we applied the linear transformation from real numbers WE to $\mathbb{R}^+$. The Word Embedding vector generated by the Gensim framework contains positive and negative values in the real numbers set. To see if there is an influence when using positive and negative numbers, we performed experiments also without any transformation. For the Binary normalization, the results are the same because in fact we take in consideration only the occurrence or absence of the word. In the Cornell-Smart normalization, we use the logarithm function that obviously does not work for negative numbers, so we could not test it. Only for the Nominal normalization, we obtained different results that are presented in Table 2.

### Table 2. Accuracy results for WE equal with 10 in Real domain

| Kernel Type | Kernel degree | Data representation | Accuracy for 1309 features | Accuracy for 1309 feature and WE10 Real |
|-------------|---------------|---------------------|----------------------------|----------------------------------------|
| **Polynomial** | D1.0           | NOMINAL             | 86.69%                     | 85.62%                                 |
|             | D2.0           | NOMINAL             | 85.03%                     | 84.52%                                 |
|             | D3.0           | NOMINAL             | 84.35%                     | 83.24%                                 |
|             | D4.0           | NOMINAL             | 81.54%                     | 80.82%                                 |
|             | D5.0           | NOMINAL             | 80.73%                     | 57.72%                                 |

Even in this case, when we keep the positive and negative values from WE, the results are not better compared to the classic VSM representation. Comparative with WE representation only in positive domain the results are approximately the same, but constantly weaker.

From the training time point of view, this time increased because the vector dimension increases also. For all performed experiments, we obtain an average training time of 4.07 hours for the
polynomial kernel and 9.35 hours for Gaussian kernel. The Gaussian kernel, because of its non-linearly transforming of the data into a higher dimension space, usually takes more time to learn. These values were obtained on a personal computer with i7 CPU working at 3.1 GHz, having 8GB DRAM memory and Windows 10 operating system.

Table 3. Classification accuracy results for WE equal with 5, only positive domain

| Kernel Type | Kernel degree | Accuracy for 1309 features | Accuracy for 1309 feature and WE5 |
|-------------|---------------|-----------------------------|----------------------------------|
| Polynomial  | D1.0          | 80.99%                      | 81.45%                           |
|             | D2.0          | 87.11%                      | 86.64%                           |
|             | D3.0          | 86.51%                      | 85.79%                           |
|             | D4.0          | 71.84%                      | 74.61%                           |
|             | D5.0          | -                           | 72.22%                           |
| RBF         | C1.0          | 82.99%                      | -                                |
|             | C1.3          | 83.57%                      | -                                |
|             | C1.8          | 84.30%                      | -                                |
|             | C2.1          | 83.83%                      | -                                |
|             | C2.8          | 83.66%                      | -                                |
|             | C3.1          | 83.66%                      | -                                |

3.2 Results obtained using 5 elements in the Word Embedding representation

Because using the Word Embedding vector with dimension 10 the results are not so good and because the training time is quite high, we repeated the experiments with the Word Embedding vector dimension equal with 5 (note WE5). The experiments were performed under the same conditions as the previous (WE10) and the results are presented in Table 3. In this case the WE vector dimension has 6545 dimensions, smaller than previous, but still huge comparatively with the classical VSM representation.

In this case, the results are better comparatively with WE10 but are still weaker than in the classical VSM representation. The training time needed in all experiments for the learning step substantially decreases and it was in average 1.2 hours for the polynomial kernel and 6 hours for the Gaussian kernel, using the same system configuration.

With a size of only 5 elements in the Word Embedding representation, the coded semantic information for a word is rather weak. In all articles that discuss about the Word Embedding the authors recommend at least 50 dimensions for a WE vector, and sometimes even 100 dimensions. With these dimensions, more semantic information is codified for each word about the context for the word, and maybe the codification is better and helps the classification algorithm.

In real issues, for document classification these dimensions lead to very large documents representation (of 65450 elements or 130900 elements) which makes this problem hard to solve using normal PC host computers.

In Table 4 are presented results obtained for only Nominal normalization for data represented in R with Word Embedding vector having positive and negative values.
his problem can be solved partially, otherwise are not.

Thus, during the Kernel Type

Polynomial

problem that need systems huge vectors information. Thus, the results from the classic VSM representation, but however, they are slightly smaller.

4 CONCLUSIONS AND FURTHER WORK

In this article we present some experiments designed on documents classification where we include in the representation of text documents some semantic information, in hope that we could obtain better classification results. For this new approach, we use one of the new method presented and used in Natural Language Processing domain, called Word Embedding. In articles that talk about Word Embedding and present some experiments, as far as we know, all examples contain only few words, thus using a very small document representation. Usually, each document contains only a phrase, and using the WE representation produces good results. When we try to represent real complex documents that contain more words (in our dataset, initially approximatively 20000 words and after feature selection it was reduced to 1309 words) the obtained results were not so good. We developed experiments with a WE vector length of 5 and 10. For higher dimensions the learning time and memory needed increase too much. The results are closer to VSM representation but in almost all cases are slightly smaller.

Thus, the open question is if the WE representation is helpfully in the large document classification. Theoretically WE representation introduces some semantic information in the document representation. This new information should help the learning algorithm to obtain better classification results but first we need to find new methods for adequately representing this huge amount of information for real text documents.

These experiments and obtained results are also helpfully in the educational part because they could expand the knowledge horizon of our undergraduate and master students in Computer Science field. In this paper, we have presented some simple examples of how to add information about the document’s semantics, such as the representation of WE, without essentially changing the classic way of documents representation and classification. Thus, during the lessons, we can present the new WE paradigm of word representation in comparison with the classical VSM approach. This paper also helps to improve the curriculum because it presents a simplified approach, through vectors of frequency vectors (hyper-vectors), for modifying the current classifiers, especially those based on learning by kernels, so the students will be able to learn other methods regarding the data representation.

Results obtained for a small Word Embedding vector dimension are not very encouraging. Perhaps higher WE dimension could help but the vector dimension for document representation needs a lot of memory. This problem can be partially solved using some programming tricks without losing information. Another disadvantage is due to the fact that the training time needed for learning these huge vectors increases exponentially. This problem can be solved partially, using some multicore systems that can run in parallel some parts (threads) of the learning algorithm. This remains an open problem that needs to be further solved.

Table 4. Accuracy results for WE equal with 5 in Real domain

| Kernel Type | Kernel degree | Data representation | Accuracy for 1309 features | Accuracy for 1309 feature and WE 5 Real |
|-------------|---------------|---------------------|---------------------------|---------------------------------------|
| Polynomial  | D1.0          | NOMINAL             | 86.69%                    | 85.41%                                |
|             | D2.0          | NOMINAL             | 85.03%                    | 84.05%                                |
|             | D3.0          | NOMINAL             | 84.35%                    | 82.99%                                |
|             | D4.0          | NOMINAL             | 81.54%                    | 81.62%                                |
|             | D5.0          | NOMINAL             | 80.73%                    | 80.77%                                |

From the classification accuracy point of view, the obtained results with VSM+WE representation are closer to the classical VSM representation but are constantly slightly smaller. This means that the VSM+WE representation introduces some noise into the data, leading to disturb the learning classification algorithm. Except for only few cases where the obtained values are better than the obtained values in VSM standard representation, in all other cases the obtained values are closer to the results from the classic VSM representation, but however, they are slightly smaller.
5 REFERENCES

Bengio, R. Ducharme, P. Vincent. (2003). A neural probabilistic language model. Journal of Machine Learning Research, 3:1137-1155, 2003.

Forman G. (2004) A Pitfall and Solution in Multi-Class Feature Selection for Text Classification, Proceedings of the 21st International Conference on Machine Learning, Banff, Canada, 2004.

Hinton. G. (1986) Learning distributed representations of concepts, Proceedings of the Eighth Annual Conference of the Cognitive Science Society, Amherst, Mass, 1 – 12, 1986.

Mikolov T., Chen K., Corrado G., and Dean J. (2013) Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR, 2013.

Mikolov T., Sutskever I., Chen K., Corrado G., and Dean J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of NIPS, 2013.

Mitchell T. (1997) Machine Learning, McGraw Hill Publishers, 1997.

Morariu D. (2008) Text Mining Methods based on Support Vector Machine, MATRIX ROM Publishing House, București, ISBN 978-973-755-343-0, 2008.

Nello C., Swawe-Taylor J.(2000) An introduction to Support Vector Machines, Cambridge University Press, 2000.

Řehůřek R., Sojka P. (2010) Software Framework for Topic Modelling with Large Corpora, Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, 2010.

Reuters Corpus, [Online]. http://about.reuters.com/researchandstandards/corpus/. Released in November 2000.

Schoelkopf B., Smola A. (2002) Learning with Kernels, Support Vector Machines, MIT Press, London, 2002.

Vintan L., Morariu D., Cretulescu R., Vintan M.. (2017) An Extension of the VSM Documents Representation, International Journal of Computers, Communications & Control, ISSN 1841–9836, Vol. 12, Issue 3, pp. 403 - 414, June 2017.