Support Vector Machines and Word2vec for Text Classification with Semantic Features

Joseph Lilleberg
Computer Science Department
Southwest Minnesota State University
Marshall, MN 56258
USA
joseph.lilleberg@smsu.edu

Yun Zhu, Yanqing Zhang
Computer Science Department
Georgia State University
Atlanta, Georgia 30302-5060
USA
yzhu7@student.gsu.edu, yzhang@gsu.edu

Abstract—**With the rapid expansion of new available information presented to us online on a daily basis, text classification becomes imperative in order to classify and maintain it. Word2vec offers a unique perspective to the text mining community. By converting words and phrases into a vector representation, word2vec takes an entirely new approach on text classification. Based on the assumption that word2vec brings extra semantic features that helps in text classification, our work demonstrates the effectiveness of word2vec by showing that tf-idf and word2vec combined can outperform tf-idf because word2vec provides complementary features (e.g. semantics that tf-idf can't capture) to tf-idf. Our results show that the combination of word2vec weighted by tf-idf and tf-idf does not outperform tf-idf consistently. It is consistent enough to say the combination of the two can outperform either individually.**

Keywords—word2vec; tf-idf; text classification; scikit-learn; support vector machines; unsupervised learning; supervised learning, semantic features

I. INTRODUCTION

Text classification is widely seen as a supervised learning task that is defined as the identification of categories of new documents based on the probability suggested by a specified training corpus of already labelled (identified) documents. As the amount of available textual information of new documents available online increases, managing to classify them properly becomes more difficult. This is because the ability to effectively retrieve the correct categories for new documents relies heavily upon the amount of labelled documents already available for reference [11].

Traditional document representation involves classification using information retrieval techniques such as continuous bag-of-words or tf-idf. Widely used in natural language processing, these techniques help in providing a simplified representation of documents through various features. Continuous bag-of-words by disregarding grammar and word order but keeping multiplicity and tf-idf by reflecting the importance of a word to a particular document in a collection of documents or corpus [13, 21].

With Google's introduction of word2vec, a new approach to document representation emerged. In our work, we worked under the assumption that word2vec brings extra sematic features that help in text classification. Our initial approach involved classifying documents with word2vec without omitting stop words by simply summing them. Then by comparing these results against tf-idf without stop words, we observed that tf-idf with stop words was clearly out performing word2vec. So then we looked at a tf-idf weighted approach by adding weights to each word based on its frequency within the document using word2vec while omitting stop words, creating weighted sums of word vectors. The results were that word2vec weighted by tf-idf without stop words outscored word2vec with stop words. Also, word2vec weighted by tf-idf without stop words fell short in comparison to tf-idf without stop words. By combining word2vec weighted by tf-idf without stop words and tf-idf without stop words, we were able to achieve better results than tf-idf without stop words alone. With the assumption that word2vec brings extra semantic features, we propose that the combination of word2vec weighted by tf-idf without stop words and tf-idf without stop words can outperform either word2vec weighted by tf-idf without stop words and tf-idf with or without stop word.

II. WORD2VEC AND TF-IDF

The purpose of text classification is to categorize documents into a fixed number of predefined categories. Each document can be classified in multiple, exactly one, or no category at all. The classification of documents is seen as a supervised learning task because the objective is to use machine learning to automatically classify documents into categories based on previously labelled documents [11]. Some of the popular techniques in automatic text classification are Naïve Bayes classifier, SVM (support vector machines), and tf-idf (term frequency – inverse document frequency) [12]. Our work focuses on using tf-idf in conjunction with word2vec.

Tf-idf, defined as term frequency-inverse document frequency, is used to determine what words of a corpus may be favorable to use based on each word's document frequency. Tf-idf representation ranks among the best approaches for retrieving documents and labeling them. However, there is no compelling reason in preferring tf-idf to any other experiencebased techniques for problem solving [19]. Tf-idf calculates a value for each word in a document through an inverse proportion of the frequencies of the word in a certain document and to the percentage of documents to which the word appears in. The higher tf-idf values words have imply they have a stronger relationship in the document which they appear [14].

Tf-idf is the composite weight of two statistics, term frequency and inverse document frequency. Term frequency is when each term is given a weight by assigning a weight to the term t based on the number of occurrences in document d denoted by tft,d [15]. Since there exists various ways to determine term frequency, we will use the raw frequency of a term in a document which is given as follows:

$$\text{tf}(t,d) = 0.5 + \frac{0.5\ x\ f(t,d)}{\max\{f(w,d): w \in d\}} \qquad [13]$$

Other ways include Boolean "frequencies" or logarithmically scaled frequencies [13].

By denoting the total number of documents in a collection as N, we can define inverse document frequency of a term t as:

$$\text{idf}_t = \log\frac{N}{dft} \qquad [15]$$

In word2vec, there are two main learning algorithms, continuous bag-of-words and continuous skip-gram. [4] With continuous bag-of-words, the order of the words in the history does not influence the projection. It predicts the current word based on the context. Skip-gram predicts the surrounding words given the current word. Unlike the standard bag-of-words model, continuous bag-of-words uses a distributed representation of the context. It's also important to state that the weight matrix between input and the projection layer is shared for all word positions [3]. Our work uses the Skip-Gram model by default which has the training complexity architecture of

$$Q = C \text{ x } (D + D \text{ x } \log_2(V)),$$

where the maximum distance for words is C, D are word representations, and V is dimensionality [3]. This means that for each training word, we will randomly select a number R in range $< 1;C >$ and use R words from history and R words from the future of the selected word as correct labels. This requires us to do two R word classifications with the selected word as input and each of the R+R words as output. Using binary tree representations of the vocab the number of output units that are need evaluation can go down to approximately $\log_2(V)$ [3, 10].

For our work, vocab = $\{t_i \mid i \in 1 \ldots N\}$ where N is the vector size and documents $d_i = <w_i \ldots w_j>$. Let $w2v(t_i)$ denote the vector representation of our approach. Then

1. $R(d_i) = \sum_t w2v(t)$ where $t \in d_i$
2. $w\_R(d_i) = \sum_t w_t\ w2v(t)$ where w_t = tf-idf weight of t
3. $C(d_i)$ = concatenate(tf-idf(di), $w\_R(d_i)$)

Mathematically speaking, our first step is the summation of vector representation using word2vec. We then applied weights using tf-idf weighting with word2vec. Then we concatenated tf-idf with our word2vec weighted by tf-idf. The concatenation merges the vectors (i.e. suppose that tf-idf has a size of 2000 and w_R has size 200, then the concatenation representation would have a size of 2200).

When dealing with text classification, it's imperative to remove stop words. Stop words are common words such as "the, a, of, for, he, she, etc." that provide no significant importance to classification. It is common for text classification to completely ignore stop words. This is because they usually slow down the process without any improvement in accuracy or performance. The elimination or reduction of stop words is likely to provide more beneficial results [16]. Our work demonstrates this effect below.

III. Our work

Word2vec has garnered a lot of interest in the text mining community [3, 4, 10]. Unlike most text classification, word2vec can be seen as both supervised and unsupervised. It is supervised in the sense that the model derives a supervised learning task from the corpus itself using either the continuous bag-of-words model or continuous skip-gram model. It is considered unsupervised in the sense that you can provide any large corpus of your choice [2]. Since word2vec is unable to distinguish the importance of each word in respect to the document being classified as it treats each word equally, it becomes difficult to extract which words hold higher value over others in respect to a particular document.

There have been various ways in the approach others have taken with word2vec. Christopher Moody, a Data Scientist at UC Santa Cruz, has developed his own search site, ThisPlusThat, using word2vec. Using word2vec's algorithm to understand concepts of words and being able to add or subtract like vectors, Moody was able to disambiguate the words in the text using Hadoop's mapping function and develop a search site that lets you 'add' words as vectors [17]. Another approach taken was to expand the word2vec framework to capture meaning across languages. This approach was taken by Lior Wolf and colleagues from Tel Aviv University. By representing a word in two languages using a common "semantic" vector space, the result can be used to improve lexicons of under-resourced languages [18].

Our approach to word2vec based under the assumption that word2vec brings extra sematic features that help in text classification is a new approach because most work involving word2vec, to our knowledge, doesn't involve tf-idf. By adding weights to each word based on its frequency within the document in word2vec and omitting stop words, we created weighted sums of word vectors. Using the weighted sums of word vectors to represent documents combined with tf-idf without stop words, we propose that word2vec weighted by tf-idf combined with tf-idf will outperform tf-idf alone. We used skip-gram by default because skip-gram has the highest semantic accuracy but at the cost of time efficiency [3]. Our work also settled on a default vector size of 100 for the majority of our experiments for time efficiency. The result of one case is given in Table 1. When looking at these results, it's important to note that the categories play a huge role in the scores. Categories such as sports and agriculture are more likely to have higher scores because they are unlikely to have similar key terms in them as opposed to agriculture and nutrition. Another important factor is the set of documents used to train the model. Different sets of documents used to train the model can change the values significantly depending on the topics chosen.

Table 1*: Accuracy of different techniques using word2vec and tf-idf.*

2 Different Categories of different topics	*Score*
Word2vec with stop words	0.841892
Tf-idf with stop words	0.894595
Word2vec weighted by tf-idf w/o stopwords	0.895946
Tf-idf without stop words	0.881081
Word2vec weighted by tf-idf w/o stop words + Tf-idf without stop words	**0.897297**

Using word2vec, our initial approach was a summation of vectors in a particular document and then using a linear support vector machine (linearSVM) to help classify them. We then tried a summation but without stop words and weighted them by tf-idf. We then used tf-idf with and without stop words to indicate how well our approach compared.

By weighting word2vec with tf-idf without stop words and then combining it with tf-idf without stop words, we achieved scores that resulted better than tf-idf. We ran several test cases and we concluded word2vec weighted by tf-idf without stop words combined with tf-idf without stop words will not always perform better than tf-idf. This is because although in most instances the combination word2vec with tf-idf without stop words and tf-idf without stop words performed better, there a few cases that it did not concluding that it will not always perform better. However, in cases where tf-idf outperformed the combination, the difference was almost insignificant so it's reasonable to say that the results of the combination are reliable to use.

IV. RESULTS

Since word2vec is only able to convert words or phrases into vector space representation, our approach was to use the weighted sum of the word vectors to represent each document. With a 20 newsgroup text dataset provided by [1], we acquired 18,000 newsgroup posts on 20 topics that were split into 2 subsets: one for training (training data) and one for testing (test data). We used Scikit-learn.org's module to load the posts as a list of raw texts [6]. We then used word2vec to train our own model. Since word2vec's training is an unsupervised task, there's no way to truly evaluate the result. The evaluation depends on the end application. Using the training data and test data, we cleaned the documents (e.g. removing capitalization, removing punctuation, tokenization, etc.) Once the documents were cleaned, we converted them into vector space representation giving us a word2vec training and test model. Using Scikit-learn.org's LinearSVC (Linear Support Vector Classification), we fit our word2vec training model against the training data target. We then used the predict feature of LinearSVC to predict the class labels in our word2vec test model. By comparing the test data target against our word2vec test model, we acquired the accuracy score for word2vec with stop words [7].

For tf-idf, we used Scikit-learn.org's TfidfVectorizer. Using TfidfVectorizer's fit and transform feature, our tf-idf training model learned the vocabulary and document frequency of each word in the training data [8]. After fitting our tf-idf training model predicting the class labels in our tf-idf test model using TfidfVectorizer's predict feature, we acquired the score for tf-idf with stop words. When fewer categories are used, the scores of word2vec and tf-idf are relatively close; however, the gap between word2vec and tf-idf increases as the number of categories increased with tf-idf placing higher as shown in Fig. 2. When we use various category sizes and topics, we can notice that the again fewer categories provides the better results as can be seen in Fig 1. As the categories increase in size, the difference between their scores becomes less significant also. For tf-idf without stop words, we replicated the process that we used for tf-idf with stop words but initialized the TfidfVectorizer to omit stop words instead. The result was an increase in performance between tf-idf and tf-idf with stop words.

Unlike tf-idf, word2vec has some pitfalls when it comes to the removal of stop words. First, word2vec neglects the value of each word in respect to the document being classified which makes accurate classification difficult. Second, word2vec is unable to distinguish the frequency at which words appear which is especially importantly because stop words can skew the results. In order to cope with these disadvantages, we developed an algorithm that would be able to acquire the top words. This was done by using the tf-idf training and test models without stop words to determine the top words by counting the document frequency of each word and selecting the least frequent words. We then placed weights on these words giving them higher importance when we took the weighted sum of vectors. After fitting word2vec weighted by tf-idf model against the training data target and predicting the class labels, our results outscored our previous word2vec model in every instance. However, it was still unable to match the success tf-idf without stop words.

By combining the vector representations of word2vec weighted by tf-idf without stop words and tf-idf without stop words, we were able to achieve results that outscored tf-idf without stop words in most cases. Although our combined vector representations were able to achieve better results, it was still capable of being outscored by tf-idf without stop words.

The results are shown in Fig. 1 and Fig. 2 where the following numbers correspond to the following techniques: 1

corresponds to Word2Vec with stop words, 2 corresponds to tf-idf with stop words, 3 corresponds to tf-idf without stop words, 4 corresponds toWord2Vec weighted by tf-idf without stop words, and 5 corresponds to Word2Vec & tf-idf combined without stop words.

Figure 1: *Accuracy scores of x number of categories using word2vec and tf-idf with various category sizes and topics.*

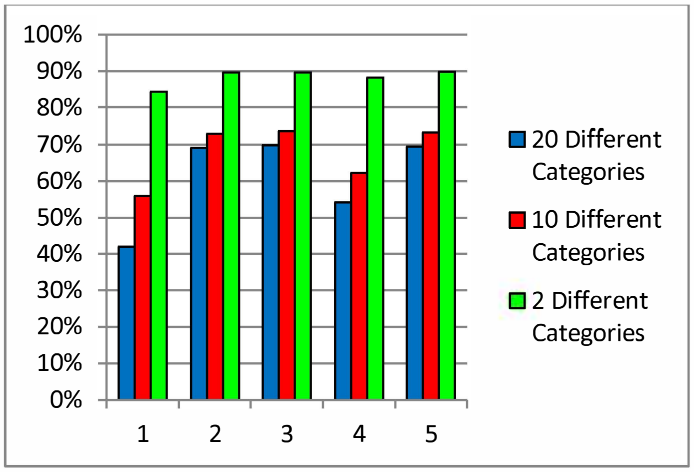
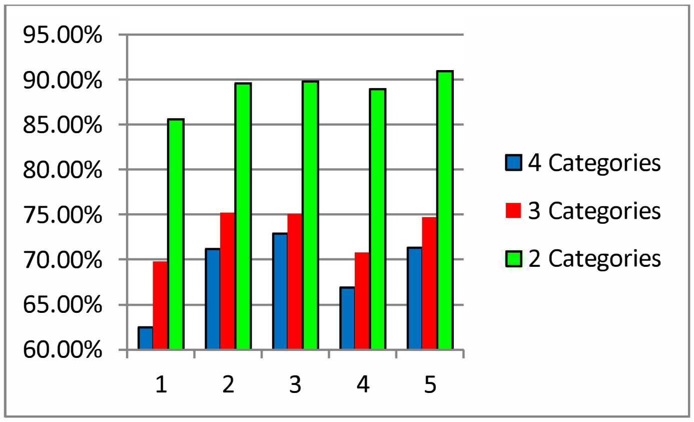


Figure 2: *Accuracy scores of x number of categories sharing the same topic.*



V. Conclusion

Our work shows that the combination of word2vec weighted by tf-idf without stop words and tf-idf without stop words can outperform either word2vec weighted by tf-idf without stop words or tf-idf with or without stop words However, the combination does not outperform tf-idf with or without stop words every time. It even fails to outperform the existing methods in some instances. However, the difference when the combination of word2vec weighted by tf-idf without stop words and tf-idf without stop word is outscored is negligible being less than a hundredth. Therefore, it is reasonable enough to say that word2vec weighted by tf-idf without stop words and tf-idf without stop words is reliable.

With our assumption that word2vec brings extra sematic features, we can conclude that the combination of word2vec weighted by tf-idf without stop words and tf-idf without stop words can result in better scores as can be seen in Table 1 between the comparison of word2vec weighted by tf-idf without stop words against the combination and then the comparison between tf-idf without stop words against the combination. In Figure 1 we can also see that as we increase the number of different categories, the difference in scores decreases meaning adding more categories is insignificant and doesn't offer much , if any, new information. We can also conclude with our approach in Figure 2 that having fewer categories can result in higher scores.

Our work with word2vec is only one of many approaches word2vec in text classification. It's also a different approach taken by others in the sense that it incorporates tf-idf in conjunction with word2vec instead of disambiguating the words which, to our knowledge, hasn't been done yet [17]. Although our approach only scratches the surface, better results can still be achieved. Word2vec brings extra tools that aren't offered anywhere else in text classification currently making word2vec very beneficial. The potential it brings to text classification is still unknown. However, we believe that word2vec has a lot to offer to text classification based our evidence and results. Future work includes ways to improve the consistency of word2vec weighted by tf-idf without stop words and tf-idf without stop words so that it'll outscore word2vec weighted by tf-idf without stop words and tf-idf with or without stop words in every instance. This can be achieved many different ways such as modifying the stop word list or modifying the weights we add to words such as what determines the weighting. As well as using other methods in conjunction with the combination of tf-idf such as mapping, or using a combination of techniques, like word2vec and tf-idf, with the meaning of the words captured [17, 18].

Acknowledgment

This research is supported by the NSF REU grant # 1156733.

References

[1] Feature extraction, Scikit-learn.org, 2011 [Online] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. Available: http://scikit-learn.org/stable/modules/feature_extraction.html . [Accessed: 19 July 2014].

[2] C. S. Perone "Machine learning :: Text feature extraction (tf-idf) – Part 1," pyevolve.sourceforge.net, 18 September 2011. [Online]. Available: http://pyevolve.sourceforge.net/wordpress/?p=1589. [Accessed: 19 July 2014].

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, " Efficient Estimation of Word Representations in Vector Space," arxiv.org, , 7 September 2013. [Online]. Available: http://arxiv.org/pdf/1301.3781.pdf. [Accessed: 20 July 2014].

[4] "word2vec," code.google.com, [Online] 2013. Available: https://code.google.com/p/word2vec/. [Accessed: 20 July 2014].

[5] Radim, Řehůřek. " Word2vec Tutorial", radimrehurek.com, [online] 2 Feburary 2014, http://radimrehurek.com/2014/02/word2vec-tutorial/ (Accessed: 19 July 2014).

[6] The 20 newsgroups text dataset, Scikit-learn.org, 2011 [Online] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. Available: http://scikit-learn.org/stable/datasets/twenty_newsgroups.html. [Accessed: 19 July 2014].

[7] sklearn.svm.LinearSVC, Scikit-learn.org, 2011 [Online] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. Available: http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC.predict . [Accessed: 19 July 2014].

[8] sklearn.feature_extraction.text.TfidfVectorizer, Scikit-learn.org, 2011 [Online] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. Available: http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html . [Accessed: 19 July 2014].

[9] Wikipedia contributors, " Vector space model," en.wikipedia.org, 9 July 2014. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Document_classification&oldid=612457216. [Accessed: 18 July 2014].

[10] Y. Goldberg and O. Levy, " word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method," arxiv.org, 15 Feburary 2014. [Online]. Available: http://arxiv.org/pdf/1402.3722.pdf. [Accessed: 20 July 2014].

[11] T. Joachims, " Text Categorization with Support Vector Machines: Learning with Many Relevant Features," www.cs.cornell.edu, 1998. [Online]. Available: http://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf. [Accessed: 20 July 2014].

[12] Wikipedia contributors, "Document classification," *en.wikipedia.org*, 11 June 2014. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Vector_space_model&oldid=616216662. [Accessed: 18 July 2014].

[13] Wikipedia contributors, " Tf–idf," en.wikipedia.org, 11 July 2014. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Tf%E2%80%93idf&oldid=616537234. [Accessed: 18 July 2014].

[15] P. Fan and T. Pong, "Knowledge Representation from Lecture Videos through Multimodal Analysis"in Proceedings of the International Journal of Information and Education Technology 2013; Cambridge, United Kingdom: Cambridge University Press, June 2012. Available: http://nlp.stanford.edu/IR-book/pdf/06vect.pdf. [Accessed: 19 July 2014]

[16] B. Patel and D. Shah, " Significance of Stop Word Elimination in Meta Search Engine," in Proceedings of the 2013 International Conference on Intelligent Systems and Signal Processing (ISSP), Mar 1-2, 2013, Anand (Gujarat), India. Gujarat: IEEE, 2013. Available: IEEEXplore, http://ieeexplore.ieee.org.ezproxy.gsu.edu/stamp/stamp.jsp?tp=&arnumber=6526873. [Accessed: 19 July 2014].

[17] C. Moody, " ThisPlusThat: A Search Engine That Lets You 'Add' Words as Vectors," insightdatascience.com, 13 November 2013. [Online]. Available: http://insightdatascience.com/blog/thisplusthat_a_search_engine_that_lets_you_add_words_as_vectors.html. [Accessed: 19 July 2014]

[18] Lior Wolf, Yair Hanani, Kfir Bar, and Nachum Dershowitz, April 2014, "Joint word2vec Networks for Bilingual Semantic Representations" (Poster), Conference on Intelligent Text Processing and Computational Linguistics (CICLing), Kathmandu, Nepal.

[19] C. Elkan; "Deriving TF-IDF as a fisher kernel"in *String Processing and Information Retrieval*; Mariano Consens and Gonzalo Navarro, 12th International Conference, SPIRE in Buenos Aires, Argentina, Springer Berlin Heidelberg, pp 295-300, 2005.

[20] A. L. Maas and A. Y. Ng, " A Probabilistic Model for Semantic Word Vectors," in Proceedings of the NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning, 10 December, 2010, Palais des Congrès de Montréal/Convention and Exhibition Center, Montreal, Quebec, Canada. http://ai.stanford.edu/~ang/papers/nipsdlufl10-ProbabilisticModelSemanticWordVectors.pdf. [Accessed: 19 July 2014].Congrès de Montréal/Convention and Exhibition Center, Montreal, Quebec, Canada. http://ai.stanford.edu/~ang/papers/nipsdlufl10-ProbabilisticModelSemanticWordVectors.pdf. [Accessed: 19 July 2014].

[21] Wikipedia contributors, " Bag-of-words model," en.wikipedia.org, 9 July 2014. [Online]. Available: http://en.wikipedia.org/wiki/Bag-of-words_model. [Accessed: 18 July 2014