Master Computer Science

[Analysis and Improvement of Document Classification based on Word2vec and TF-IDF]

Name:	Jihui Liu
Student ID:	s1987445
Date:	30,05,2019
Specialisation:	Computer Science and Business Studies
1st supervisor:	Wessel Kraaij
2nd supervisor:	Cor Veenman

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Analysis and Improvement of Document Classification based on Word2vec and TF-IDF

Abstract. Recently, deep learning approaches have been proposed for document classification. A common approach is to combine word2vec embeddings with CNNs. However, there is no clear foundation how much of the success of the combination can be attributed to the word embeddings, and the choices of different pre-trained vectors and classifiers. Meanwhile, word2vec embeddings ignore the importance of the word to a document among the corpus. To solve this problem, an improved weighting method based on TF-IDF is proposed in this paper. Several experiments are conducted in order to assess the robustness and effectiveness of different variants including the proposed method.

1 Introduction

In order to put words into a machine learning algorithm, the text data should be converted into some machine-readable numeric or vector representation. The simplest technique is one-hot representation. As for the unique word, the one-hot representation is to create a vector with binary values of corpus size. And the vector is filled with all 0 except the only corresponding index, which is replaced by 1. Then a document will be represented by bag-of-words(BoW) model. The BoW method treats a document as a collection of words but regardless of the order, grammar, and semantics of the word. Therefore, in the BoW vector of a document, the value of each index will no longer be binary(either 0 or 1) but the number of times the word appears in this document. But one disadvantage with BoW model is that the importance of words are evenly weighted as the times they occur. However, the real situation could be different since some words may be more relevant than others.

TF-IDF, which is short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in corpus[1]. Then the document representation could be a vector which provides high value for a given term if it occurs frequently in certain document and rarely anywhere else. The full explanation of TF-IDF will be discussed in section 3.2. However even though TF-IDF representations strengthen the ability of a vector to represent a document, the vector itself is still easy to be very sparse, also known as dimension explosion phenomenon.

Word embedding is later proposed to transform a word vector from a space with high dimensions to a low-dimensional, dense real-vector space. There are a series of methods to generate word embeddings, such as neural network[2] and word co-occurrence matrix[3][4][5]. By making use of the shallow(two-layer) neural network, word2vec was proposed by Mikolov et al.[6] in 2013. It generates word representations with the relations between a word and its contextual words in a simple, scalable and fast way. As for the word-word co-occurrence matrix, it was applied by Pennington[7] to generate Global Vectors(GloVe) in 2014.

Even though the lower-dimension and neat word representations are obtained by word embeddings, how to use a word vector to effectively represent a document is still a challenge at present. Normally, a document can be represented by averaging the sum of all word vectors included, or generating document matrix by all word representations. However, these methods usually ignore the importance of a word to a document in the whole collection. Hence, we try to propose a hybrid method that applies TF-IDF(discussed in section 3.2.4) algorithm to weight the word in each document, and test it on different experimental corpus. The involved experiments will cover various classifiers such as Support Vector Machine(SVM) and Convolutional Neural Network(CNN), and different classification type like binary and multi-class classification. The main contributions of this paper are:

1. Verify the effectiveness of the combined method based on TF-IDF and word embeddings.
2. Analyze the effects of different factors, including word vector size and methods(word2vec or GloVe) on the classification effect.
3. Analyze the classification performance of different classifiers(SVM and CNN) with different input representations.

2 Problem Statement

Concerning the inspiration from previous empirical analyses of Yoon Kim[8], Mounir Hader[9] and Ye Zhang[10], many achievements have been obtained by them. In the beginning, Yoon Kim[8] proposed a simple architecture of CNN that obtained satisfying results on the classification tasks. In addition, based on the model proposed by Kim[8], Ye Zhang[10] analyzed the sensitivity of CNN model performance by tuning hyperparameters for the task of classification. According to the conclusion of Mounir Hader[9], it was approved that CNN with a simple architecture(shown in Figure 2) could be sufficient to achieve a satisfying result comparing with Naive Bayes and linear SVM. In addition, Linear SVM yielded similar results as CNN model on 2 out of 3 datasets. At last, considering the lower computational time and complexity(without tedious parameter tuning), SVMs could be recommended to be the most practical method for the basic classification task.

On the other hand, based on the accomplished experiments by them[8][10][9], we conclude that there are many aspects which can be further supplemented. Thus, we aim to design the experiments based on the following problems.

- It is unclear if the success of the CNN is attributed to the model itself or input representation(dimension reduction, and word embedding).
- Most chosen datasets are intended for the binary classification task. Those datasets are all well-organized and distributed, which means the positive and negative samples are almost even. Besides the classification type, an unevenly distributed dataset can be also necessary to test the performance of specific models.
- Experiments of Kim[8] and Zhang[10] only applied word embedding as the input of CNNs. This means the influence of traditional model like SVM with word embeddings stay unclear.
- As for the word embedding collection, only the vectors trained on Google News was applied. However, other kinds of word vectors like GloVe and self-trained(based on the datasets themselves) vectors were rarely considered in the work of Kim[8] and Hader[9].
- All the involved word vectors were fixed to 300-dimension. It is likely that 300-dimension vectors are sufficient for different models. But the influence of vector length is still unknown.
- The word vectors that have been trained are consistent for different documents. But the truth is a bit different, even if the same word may be different for different documents with respect to the importance. Then, applying a fixed vector to all documents in general can lose some specific information.

3 Method

Referring to the problems draw in Problem Statement section, a series of experiments will be designed in order to solve those problems as following:

- More models with different kinds of input representations(BoW with TF-IDF values, and word embedding) needs to be involved.
- Extending the involved datasets from binary to multi-class classification can be required. Besides the classification type, the comparison experiments of approaches on the unevenly distributed dataset are necessary to test the performance of specific models.
- Word embedding generated with word2vec needs to be applied on both SVM and CNN models in order to find a certain model that may obtain a better performance relatively.
- Instead of using vectors trained on Google News only, different word vectors like GloVe and self-trained vectors should also be tried to observe whether the influence is because of learning knowledge on external corpus or on the training data.
- Different vector size will also be involved in order to observe the influence on the final performance.
- By combining TF-IDF and word embedding, a hybrid enhancement technique will be proposed to increase the performance.

3.1 Preprocessing Data

3.1.1 Word segmentation

Text segmentation is an essential operation in the preprocessing process. Because subsequent classification operations require the use of words in the text to represent the document. During the segmentation procedure, the punctuation(commas, parentheses, exclamation marks and so on) and case-sensitivity(*Where* is equal to *where*) of words will be ignored.

3.1.2 Stopwords

Aiming at saving storage space and increasing efficiency, stopwords are the words will be automatically filtered when processing the raw data[1]. In fact, most of these words that are filtered out are words that have no practical meaning or are too frequent. For example, *a*, *an* , *the* and etc. The list of stop words is easy to find online. When crawling or indexing of a huge amount of data, stopwords are also dedicated to decrease the computational time and save space[11].

3.2 TF-IDF Weighting

3.2.1 Term frequency

With respect to a certain document, the term frequency (TF) means how many times a given word occurs in that document. The following equation displays how to express the importance of a certain word i in a particular document:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{1}$$

$n_{i,j}$ in the above expression is the times that the word appears in the document d_j, and the denominator is the sum of the occurrences of all words in the document d_j.

3.2.2 Inverse document frequency

The inverse document frequency (IDF) is a method applied to express the universal importance of a word in the collections[12]. The IDF value of a certain word can be obtained by dividing the amount of whole documents by the number of documents that the word occurs, and then taking the resulting quotient from the base 10 logarithm:

$$idf_i = \log \frac{|D+1|}{\{j : t_i \in d_j + 1\}} + 1 \tag{2}$$

where |D| refers to the amount of all documents in the corpus. $\{j : t_i \in d_j\}$ means the document frequency(DF), which is used to describe how many documents that the word occurs. Simply speaking, it is the amount of documents that meet $n_{i,j} \neq 0$. However, if the term is the out-of-vocabulary(OOV) word, the denominator will become zero. In order to prevent zero divisions, the constant "1" will be added to the numerator, denominator and the complete idf_i formula as well.

3.2.3 TF-IDF

Given by the formula 1 and 2, we can obtain the TF-IDF equation which goes to:

$$tf - idf_i = tf_{i,j} \times idf_i \tag{3}$$

However, it is unlikely that a term of 10 occurrences within a document means 10 times the importance of one individual occurrence indeed. Therefore, we use another common scaling named *sublinear* tf scaling $(1 + \log(tf_{i,j}))$ to replace original $tf_{i,j}$.

3.2.4 Improved Representation based on TF-IDF

TF-IDF is known as a numerical statistic which is dedicated to illustrating the importance of a term towards a document among the whole dataset[13]. The importance of a word increases proportionally with the occurrence in the certain document, but it also decreases inversely with the times that it appears in the whole collection or corpus. Apparently, the TF-IDF technique is in the spirit of the assumption that the importance of words for distinguishing documents is quite related to the frequency that they occur in both one specific document and entire collections. Hence, the BoW vectors with TF-IDF values would only represent documents in a count-based way. However, word embedding(wor2vec or GloVe) is different, especially for the word vectors pre-trained by the large corpus. The word vector represents the meaning of a word by its contextual words, and even the relationship between words can be easily represented by vector itself. The explanation of word2vec and GloVe can be retrieved in the Appendix section. But the challenge with using word embeddings to represent a document is that the same word holds the same vector in different documents. Obviously, even for the same word, its importance will vary concerning different documents. But using the fixed word vector is not able to illustrate these differences. Therefore, by combining TF-IDF and word embedding, an improved word vector that can express both the importance and inherent meaning of the word simultaneously can be generated.

Below we will briefly explain how to improve the representation of a document based on TF-IDF. Assume there is the collection D containing M documents, where $D_i(i = 1, 2, ..., M)$, first those documents will be pre-processed by the methods mentioned in Section 3.1. And then we train them through the word2vec model(or use the existing pre-trained models: Google or GloVe) to get the N-dimensional words corresponding to each unique word t, where $W_t = (v_1, v_2, ..., v_N)$. For each word t, the TF-IDF method(Equation 3) is intended to achieve its importance value $tf - idf(t, D_i)$ in the document, which is expressed as the weight of word t occurs in the document $D_i(i = 1, 2, ..., M)$. Thus, the improved word representation could be reformed by weighting by its TF-IDF value, which is:

$$W_t^* = W_t \times tf - idf(t, D_i) \tag{4}$$

Compared with the traditional word representation method as BoW vector with TF-IDF values, the length of Equation 4 is obviously less than BoW vector. The vector of word embedding will be N-dimensional while the BoW vector can be the size the number of words in the whole collection or corpus.

As for representing a document by using word representations, it will differ from models. For example, the SVM model uses a vector to represent a document. Therefore, for each document $D_i(i = 1, 2, ..., M)$, its document representation can be expressed as follows, where V is the number of words in D_i, and W refers to the word vector of word t, so the document vector d is also an N-dimensional vector.:

$$d_i = \frac{\sum_{t \in D_i} W_t \times tf - idf(t, D_i)}{V} \tag{5}$$

However, the CNN model is intended to use the document matrix as a representation of the document. According to the structure proposed by Kim[8](details can be found in section 3.5.2), we also used a matrix composed of word vectors to represent a document. But the word vector was replaced by a simple pre-training vector with a vector weighted by TF-IDF, which is illustrated by Equation 4 as well.

3.3 N-grams

To represent a document, in addition to using a unique word as a feature, it is also possible to use n-grams as features. N-gram technique is referring to a contiguous sequence of n items based on a certain document. Here, Latin numerical prefixes will be applied to express n-grams. Therefore, unigram, bigram, and trigram are referring to the n-gram of size 1, 2 and 3 respectively. As for the repetition experiments(reproduce others achievements to be our baselines), N-grams is used to conduct experiments in the initial phase in order to obtain a relatively better result as baseline. On the other hand, since the word vector itself corresponds to a single word, the sequences of words are not considered in the pre-trained word vector. Therefore, in experiments involving the use of word embedding, it is not wise to construct a vocabulary through bigram or trigram. In addition, it will also increase the size of the vocabulary and the computational time. Therefore, only the repetition experiments adopt n-grams, while the remaining experiments does not consider it.

3.4 Classifiers

3.4.1 Linear SVM

Assuming there is a form of n point test set:$(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$, where y_i is 1 or -1, reflecting the label where $\vec{x}_i$ should belong. The

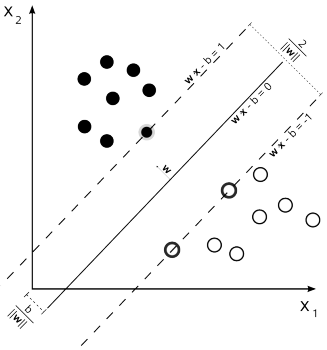


Figure 1: Linear separability: Let the sample belong to two classes, and use this sample to train the maximum margin hyperplane obtained by the SVM. The sample points on the hyperplane are also known as support vectors. Decision boundary (solid line), margin boundary (dashed line), support vector (bold dot).

dimension of each $\vec{x}_i$ vector is fixed as size p. The final target is to require the maximal margin hyperplane which separates the point set of $y_i = 1$ and the point set of $y_i = -1$. And this hyperplane maximizes the distance between hyperplane and the closest point $\vec{x}_i$. In addition, points that satisfy the following equation: $\vec{w} \cdot \vec{x} - b = 0$, where $\vec{w}$ is the normal vector, can be used to express any hyperplanes. The $\frac{b}{\|\vec{w}\|}$ determines the offset from the origin along the normal vector $\vec{w}$ to the hyperplane. Figure 1 will give a more comprehensive illustration.

3.4.2 Convolution Neural Network

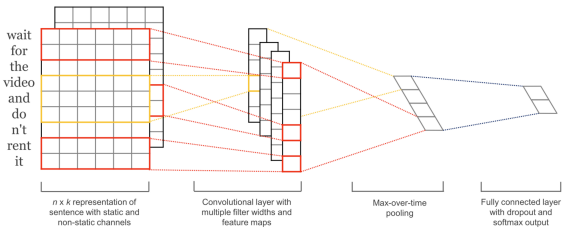


Figure 2: The simple CNN architecture with two channels for an example sentence. Figure is quoted from Yoon Kim[8].

As for the architecture of the CNN model, we chose to use the same structure as Kim[8] and Zhang[10]. Figure 2 illustrates the instance of how it works for binary classification. CNN models have a wide range of tuneable hyperparameters. It is very demanding to do the grid search with respect to all accompanying parameters. Therefore, based on the results of Kim[8] and Zhang[10], the most significant parameters are explained in Table 7 in Appendix section. Since the tun-

ing of neural networks is too time-consuming, and our experimental purpose is not primarily to study the effects of parameters, we decided to use their ready-made parameter values as some of our basic parameters.

3.5 Flow Chart

Figure 3 is the brief flow chart of how the experiment runs.

1. The raw datasets will be processed by splitting words, removing punctuation and stopwords.
2. Considering the method of word representations, the simplest way is to use BoW vectors. This method is only applied by part of experiments on Naive Bayes and SVM models. With respect to the huge length size of BoW vector, CNN model abandon using them as the input in order to save the computational time. Another option is to use word embedding, then we read words one by one and check whether a certain word exists in the pre-trained vocabularies. If positive, the pre-trained vector will be adopted. Otherwise, we randomly initialize a vector from [-1,1].
3. In order to take the importance of a certain word into consideration, word vectors should be weighted by the TF-IDF values. For the words that exist in the testing document but do not appear in the training document(OOV, short for out of vocabulary), their TF-IDF values should be zero. In the case of the traditional BoW model, OOV will be ignored since each index in the vector represents a token that occurs in the vocabulary. However, as for the experiments apply word embeddings, the pre-trained word vectors will be multiplied by their own TF-IDF values(mentioned in section 3.2.4). Then, if we multiply 0 by the pre-trained word vector, the word vector will also become zero. It means that the meaning of the word for this document is erased. Therefore, we choose to use a TF-IDF value of 1 for OOV words. This will ensure that these words will still be applied to represent a document without any changes. For those words that already exist, the value of TF-IDF should be between 0 and 1 (after normalization) in a certain document. In order to distinguish between them and OOV words, we decide to use (TF-IDF+1) as their value. Because if we still use the value of the original TF-IDF (from 0 to 1) to weight the word vectors, then we are equivalent to weakening the importance of these words compared to OOV words which use 1 instead.
4. Documents can be represented by the word representations. Given by different models, various document representations will be applied. For the SVM model, the mean of the aggregation of all word vectors will be used as the document vector. On the other hand, the CNN model will use the word matrix as the representation of a certain document.

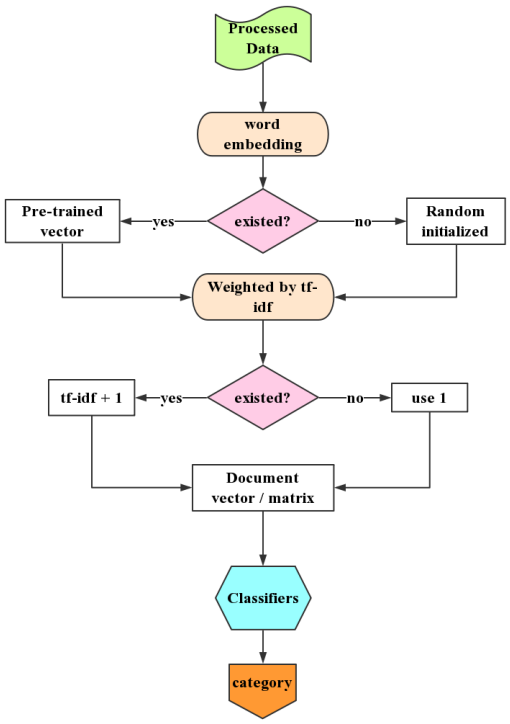


Figure 3: Part of the flow chart of experiments.

3.6 Hypotheses

Based on the given theory[8][10][9], we believe that the weighting scheme we propose(mentioned in section 3.2.4), as well as other factors(dimensional size, pre-trained vector characteristics and so on), will influence the classification effectiveness. However, since different classifiers may have different characteristics, the final classification performance may vary widely. Therefore, here we temporarily list the following hypotheses:

- Our weighting scheme(mentioned in section 3.2.4) will help classifiers get a higher accuracy relatively.
- Higher dimensional size will increase the final accuracy.
- Taking the OOV words into consideration(randomly initializing) may improve the classification performance.
- Vectors trained by itself may works worse than those learning from outside resources(Google or GloVe).

4 Experiments Setup

All the experiments involved in this paper were conducted on different datasets. Given the apparent differences in class quantity, document length, vocabulary size, and document complexity, these datasets are more proper for comparing the performance of various classification types(binary or multi-class classification). We have chosen a variety of datasets for a more informed comparison of our methods. In this way, we hope to assess the robustness of different methods. Samples of different datasets will be present in the Appendix section.

4.1 Datasets

- IMDB: the dataset contains movie review from IMDB[1][14], labeled by sentiment(positive or negative). This dataset includes 50,000 severely polarized reviews from the Internet. And those comments are split and organized into half for positive and a half for negative. Therefore, both training and testing sets will contain 50% positive reviews and 50% negative reviews.
- 20 News Group: the dataset is a corpus of approximately 20,000 newsgroup files, distributed evenly across 20 different newsgroups[2][15]. This dataset is containing 20 various newsgroups, which will be assigned to a different topic. Some of the topics are related to each other, while others are kind of unrelated to some extents.
- Consumer Complaints: the dataset is the complaints received about financial products and services[3]. There are 341,301 documents belonging to 18 different topics respectively. Besides the considerable size, this dataset has another feature which is that the number of complaints per product is imbalanced. Conventional algorithms often focus more on the majority class, not considering the data distribution. In the worst case, minority classes are treated as accidents and ignored. Therefore, this dataset can be a tough task for testing the ability of different classification methods(different representative schemes or classifiers).

4.2 Word Embedding

According to the work of Kim[8] and Zhang[10], both of them used a fixed pre-trained vector of word2vec based on Google news with 300 dimensions. However, in order to study the influential factors of word embedding(external knowledge or dimension reduction), we added some more different pre-trained vectors models. For more specific information about word2vec and GloVe, please refer to the Appendix section. There is a list of four options using word embeddings:

- Google News: includes 300-dimensional vectors of 3 million words and phrases trained based on Google News of nearly 100 billion words on word2vec[4].
- GloVe: is trained on Common Crawl with 840B unique words, 2.2M vocabulary and corpus in 300 dimensions[5].
- Self-trained Skip-gram(SG): is the vectors trained on the given datasets by applying Skip-gram model of word2vec(dataset would be divided into training and testing sets, and the self-trained vectors were only trained by training set, excluding testing set).
- Self-trained Continuous Bag-of-Words(CBOW): is the vectors trained on the given datasets by adopting Continuous Bag-of-Words model of word2vec(training procedure was the same as Skip-gram model).

With respect to the vector dimension, depending on the dataset higher dimensional word embeddings may be required. And 300 dimensions(chosen by both Kim[8] and Zhang[10]) may not be sufficient for the huge work. In addition, we tried to use different vector dimensions as inputs, for instance, 300, 600, 1200 and 2400. Secondly, some existing pre-trained vectors like Google news and GloVe, they have a fixed dimension which can not be changed. Consider this situation, those two kinds of vectors would be concatenated to form a new vector in 600 dimensions instead.

Because of the variety of different classifiers, the document length should be considered as well. For SVM which used words vectors aggregation, this issue could be ignored. However, when it referred to CNN, truncating mechanism (fixed document length 1200) should be applied in order to achieve efficiency. According to the statement of Hader[9], this should affect nothing since most texts were shorter than this limitation(on IMDB dataset).

4.3 Cross-validation

Cross-validation, also known as rotation estimation, is any of various similar model validation techniques for assessing how the results of a statistical analysis will generalize to an independent dataset[16]. Normally, it will be applied to estimate the accuracy of a predictive model in reality[17]. According to the paper of Yoon Kim[8], he applied 10-fold CV on his datasets. However, given by the differences in size and complexity of our datasets, we chose to use 5-fold CV on datasets in order to save computational time. In addition, we also kept the folds(train and test data randomly) the same

[1] http://www.cs.cornell.edu/people/pabo/movie-review-data/
[2] http://qwone.com/ jason/20Newsgroups/
[3] https://catalog.data.gov/dataset/consumer-complaint-database
[4] https://code.google.com/archive/p/word2vec/
[5] http://nlp.stanford.edu/data/glove.840B.300d.zip

across all the experiments. In that way, we could observe the result of different experiments on the same fold.

4.4 Evaluation Metrics

In our experiments, *accuracy*, *precision*, *recall*, and *F1-score* would be used as the evaluation metrics. Both of the detailed definitions and some fundamental notion can be found in the Appendix section. Since not only the binary classification but also multi-class classification was involved, therefore we also treated them individually as a binary issue, which could be c and non-c class for any specific class of multi-class classification.

There are three kinds of result averaging methods including *micro average*, *macro average*, and *weighted average*. The macro-average is intended to calculate the metric individually for each class and then calculate the mean value, which means it treats all classes fairly. While the micro-average normally averages the metrics globally by counting the total true positives, false negatives and false positives. In our experiments, we chose to utilize *weighted average*. It calculates metrics for each class and generates their average weighted by the support which is the number of true instances for each class. *Weighted average* could alter macro average to account for class imbalance and return a reasonable result relatively for all kinds of datasets.

4.5 Implementation Details

Duranium	Tritanium
126GB RAM 20 Intel Xeon E5-2650 v3 CPUs @ 2.30GHz (40 threads) 6 NVIDIA GTX 980 Ti GPUs each with 6GB memory. 2 NVIDIA Titanium GPUs each with 12GB memory.	**3TB RAM 20 Intel Xeon E5-2650 v3 CPUs @ 2.30GHz (40 threads) 16 NVIDIA Tesla K80 GPUs each with 11.5GB memory**

Table 1: The specifications of the DSLab machines.

Python 3.6 will be applied to implement all the models involved in the experiments. Part of the most significant libraries will be listed as follows:

- TensorFlow v1.3.1: it is a symbolic mathematics system based on dataflow programming. It is widely used in the programming implementation of various machine learning algorithms. Its predecessor is Google's neural network algorithm library DistBelief[18].
- Keras v2.2.4: it is a deep learning library based on Theano and TensorFlow[19].
- Gensim v3.4.0: it is a tool for mining the semantic structure of documents by measuring phrases (whole sentences or documents)[20]. This library is used to train our own word vectors.
- Scikit-learn v0.20.1: it is a free software machine learning library for the Python programming language[21]. This library is used for creating traditional models, like SVMs.
- Numpy v1.15.4: it is a mathematical computing library of Python. Numpy is usually applied to store and process large matrices. Compared with the embedded list structure in Python, it is much more efficient. Also, this structure is beneficial to express matrices in a more efficient way as well[22].

All experiments were run on the DSLab machines. Table 1 shows their configuration in details.

5 Results and Discussion

All results shown in this paper are the average value of 5-fold cross-validation, except for the dataset which has a standard train/test split(20 Newsgroups). More specifically, the dataset is randomly divided into five parts, training four of them each time, and the rest is used to test. Then the mean value of all times results will be recorded as the eventual result. Furthermore, in order to keep a scientific comparison result, we train the vector on the training part during each fold for the datasets which applies 5-fold technique. However, for the dataset that already has a standard train/test split like 20 Newsgroups, we just train once for the training set. As for the hyperparameters involved in CNN model, Zhang[10] gave a very comprehensive analysis of model variants (e.g., filter widths, k-max pooling, etc.) and their effect on performance in his work. In order to achieve efficiency, we decide to apply those suggested values(listed in Table 7) in our experiments instead of tuning them by ourselves.

5.1 Naive Bayes & Linear SVM classification

In order to set a baseline for our experiments, we just reproduced those results in the spirit of the related work of Hader[9]. The results for Naive Bayes and Linear SVM classification can be found in Table 8. For Multinomial Naive Bayes, applying a bag-of-uni+bi+trigrams led to the highest accuracy on IMDB and Consumer Complaints datasets, while uni+bigrams brought higher performance on 20 Newsgroups dataset. Concerning Linear SVM classification,

uni+bigrams achieved both the highest accuracy on IMDB and 20 New groups, whereas uni+bi+trigrams obtained better achievement on Consumer Complaints dataset.

Comparing the variants that achieve the highest accuracy of N.B. and SVM models, it is obvious that SVM with sublinear tf-idf feature weighting brought better performance on all three datasets than all variants of Naive Bayes model. Furthermore, all SVM setups achieve much higher accuracy than N.B. at least 2-6 percent. These results are also similar to the achievements obtained by Hader[9]. For instance, as for IMDB dataset, Hader[9] got the highest accuracy of 87.18% and 90.05% on Multinomial Naive Bayes(uni+bi+trigrams) and Linear SVM(uni+bigrams), while we achieve 87.22% and 90.01% respectively.

Model		IMDB	2N.G.	C.C.
SVM	CBOW	84.81	70.50	67.62
	SG	**86.31**	**78.28**	**68.36**
	GloVe	85.89	76.36	63.69
	Google	85.34	73.68	62.62
CNN	CBOW	89.39	84.26	72.14
	SG	**90.47**	88.00	**74.71**
	GloVe	88.19	85.70	73.92
	Google	90.03	**89.66**	74.57
N.B.	BoW	87.22	82.46	69.53
SVM	BoW	90.01	87.00	71.52

Table 2: Accuracy of input word representations on three datasets. All embeddings are fixed to be 300 dimensions. The last two rows refer to the baselines with traditional models such as Naive Bayes and SVM.

5.2 Linear SVM & CNN classification

This part is the key research part of our experiment, which can be mainly divided into two directions. To supplement the existing experiments based on the work of Zhang[10], and also to explore the influence of different word vectors on the classification effect, we use different pre-trained word vectors for different classifiers. At the same time, we also integrate the idea of combining word2vec and TF-IDF that we proposed into the experiment and design a sub-experiment. Hence, there are two potential choices: only word2vec, or the word2vec vector weighted by TF-IDF. Other than that, the pre-trained vectors we use as input may not always be available for certain words (either in word2vec or GloVe, or trained by itself). In such cases, we choose to whether to randomly initialize the vectors between -1 to 1, calling it *fullwords* in our experiments. Combining with the TF-IDF weighted vectors, there can be four different variants in each main experiment(word2vec, word2vec with fullwords, word2vec with TF-IDF weighting, and word2vec with fullwords and TF-IDF weighting).

Results of Linear SVM can be found in Table 9 and 11, while Table 10 and 12 show the results based on CNN classification. The concrete results can be retrieved in the Appendix section. Considering the space constraints, in this section, we only extract some key-points to illustrate and present. But those points are generally illustrative.

5.2.1 Effect of input word representations

A basic feature of the classification model is that the distributed representation of the words is used as input. The flexibility of this structure allows different pre-trained word vectors to be exchanged during model initialization[10]. Hence, we plan to study the sensitivity of classifiers based on the different input representation used. In the spirit of Ye Zhang[10], we replace word2vec with GloVe. The main differences between those two representations are listed in the Appendix section. Except for Google and GloVe vectors, we also train our own vectors based on CBOW and SG models, which are also in 300-dimension(same as Google and GloVe). Table 2 shows the results based on different resources of word vectors, while the last two rows illustrate the best baseline results of Table 8. According to Table 2, we can find that for Linear SVM, SG vector achieved the highest accuracy on the three datasets. However, for the CNN model, Google achieved the best performance on one dataset, while SG still obtained the highest accuracy on the rest. It seems that without any enhancement skills(like weighting mechanism or *fullwords* technique) self-trained vector may have a stronger ability to represent a document instead of vectors derived from external knowledge.

Referring to the difference between CBOW and SG models(seen in Appendix section), the results clarify that the SG model will bring a better performance than CBOW model on all three datasets. The specific theory can be illustrated in Figure 10 in the Appendix section. From a more general point of view, in the SG model, with the help of the context words, the central word will be trained more "professional". So that the vector result will be relatively accurate, but it will certainly take longer time. However, CBOW model is quite different. Actually, the context words will share the knowledge of one central word. As for how much the context words have learned, it depends on how many time they appear in the window of a certain central word. If they are still in the window during later rounds, they will have the opportunity to learn

more from other central words, then they can make a little progress. Therefore, comparing to SG model, the output result is definitely not good enough, but for the entire training process, CBOW model is definitely more efficient and faster. It can be seen that the number of CBOW's prediction behaviors is almost equivalent to the amount of words in the entire document, then the complexity is probably $O(V)$. Meanwhile, the SG model predicts more times than CBOW. Because each word is used as a central word, it must be predicted once using context words. This is equivalent to K times more than the CBOW method (assuming K is the window size), so the time complexity is $O(KV)$.

5.2.2 Effect of input word dimensions

Besides the effect of pre-trained word vectors on model performance, we are also curious about the factor of word vectors length size. Since previous experiments[8][10] used fixed 300-dimensional word vectors, it is difficult to know if this size of word vector is sufficient for different classifiers(SVM and CNN) on different classification tasks. Thus, in addition to the pre-trained Google news and GloVe word vectors, we also train 300, 600, 1200-dimensional self-trained word vectors to meet our sensitivity analysis to different models on different vector dimensions. Meanwhile, we also try to do some concatenating operations to compare some of the word vectors deriving from different external knowledge in the same dimension. For example: 600-dimensional vector = 300-dimensional Google + 300-dimensional GloVe = 300-dimensional SG + 300-dimensional CBOW = 600-dimensional self-trained SG or CBOW. Aiming to make a scientific comparison result, we train our own word vectors on SG and CBOW models for each fold we applies. However, vectors of Google and GloVe were fixed which cannot be further retrained. Then, we only retrain the self-trained vectors during each experiment.

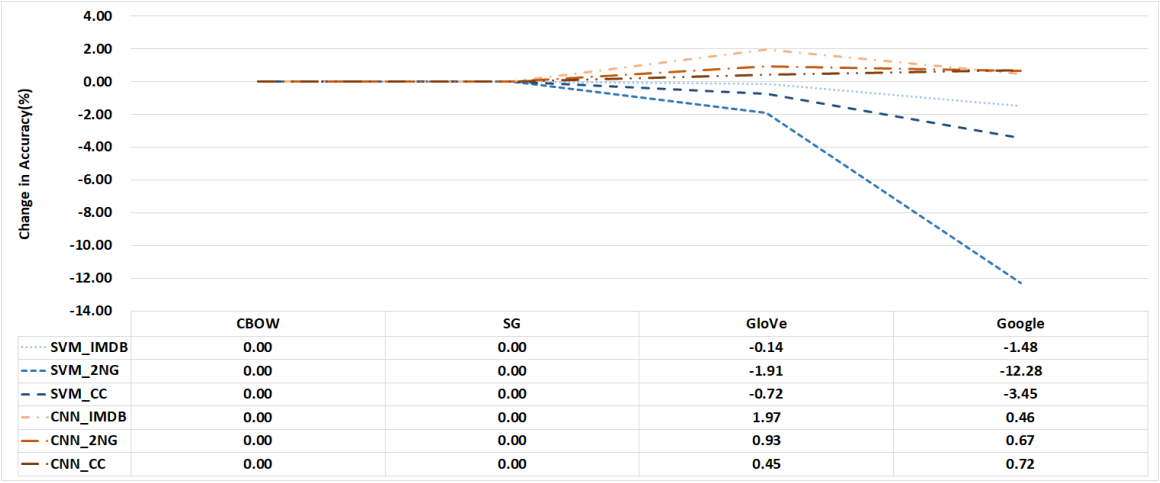
Table 11 and 12 in Appendix section mainly illustrate the performance of different dimensions on different models. Given by the extracted result of Table 3, what we can observe is that simply increasing the vector dimension seems to improve the accuracy to some extent, and the poorer performance the dataset is before dimension increment, the more obvious improvement it achieves eventually. For example, for the Consumer Complaints dataset, large-dimensional word vectors can help the classifier better distinguish between different documents. But for IMDB, it is possible that the performance of a 300-dimensional word vector is good enough. Blindly increasing the word vector may have negative effects (such as over-fitting) and even reduce accuracy. For example, in the CNN model, the 300-dimensional SG can get 90.47%(Table 2), but the 1200-dimensional SG can only get 89.55%(Table 3). On the other hand, the effect of concatenating word vectors also depends on the word vector itself. Assume there are two kinds of word vectors with similar ability, such as Google and GloVe, then the concatenating operation will enhance their ability. But if the performance of the two word vectors is very different, such as SG and CBOW of 20 Newsgroups(we can see from Table 2), then simply concatenating them may only bring deteriorate results. Take 20 Newsgroups as an example, the 600-dimensional CBOW and SG can get 73.76% and 82.20% respectively, but the concatenated 1200-dimensional vector can only get 78.65% accuracy.

Model		IMDB	2N.G.	C.C.
SVM	CBOW_600	87.10	73.76	69.34
	CBOW_1200	87.16	74.52	70.31
	SG_600	87.11	82.20	69.45
	SG_1200	**87.38**	**82.58**	**70.61**
	Google&GloVe_600	86.80	76.66	65.50
	CBOW&SG_600	86.91	74.89	69.64
	CBOW&SG_1200	**88.73**	78.65	70.62
	CBOW&SG_2400	88.14	**79.41**	**71.36**
CNN	CBOW_600	85.74	84.00	73.55
	CBOW_1200	86.80	86.72	73.88
	SG_600	89.53	88.35	**76.16**
	SG_1200	**89.55**	**89.29**	75.41

Table 3: Accuracy of input word dimensions on three datasets.

5.2.3 Effect of fullwords technique

As Figure 4 shows, the final performance of SVM model suffers from the introduction of randomly initializing vectors on Google and GloVe models. We attribute this to the case which was equivalent to introducing a certain amount of noise for SVM. Because the word vector is randomly generated and the vector does not change during the whole process, then this vector does not have any textual meaning compared to the pre-trained word vector. Therefore, if these vectors are introduced blindly, the classification performance will only be greatly reduced. However, for those models that use self-trained vectors as input, their achieved performance will stay almost the same as before. Reasons can be straight forward, when we train our own word2vec vectors(no matter SG or CBOW), the train set we use every fold will cover most words of the whole vocabulary. Therefore, those rare and unusual words that may not exist in Google or Glove will still be included in self-trained vectors. There is no doubt that there must be OOV(out-of-vocabulary)



	CBOW	SG	GloVe	Google
SVM_IMDB	0.00	0.00	-0.14	-1.48
SVM_2NG	0.00	0.00	-1.91	-12.28
SVM_CC	0.00	0.00	-0.72	-3.45
CNN_IMDB	0.00	0.00	1.97	0.46
CNN_2NG	0.00	0.00	0.93	0.67
CNN_CC	0.00	0.00	0.45	0.72

Figure 4: Effect of *Fullwords* technique on three datasets. Y-axis reflects the change in Accuracy before and after applying *fullwords* technique, while X-axis shows the results of different input representations.

words in the test set, but for self-trained word vectors, their influence will be much less than other kinds of pre-trained vectors(Google and GloVe). The specific word distribution can be seen in Figure 5. In general, whether to consider those nonexistent words can have merely no influence on SG and CBOW, while Google and GloVe can not. And the influence is closely related to the pre-trained vector itself. For example, for dataset 20 Newsgroups after introducing fullwords, the accuracy of the model using Google has dropped from 74 to 62 percent, while the model using GloVe has only dropped from 76 to 74 percent. As for Consumer Complaints dataset, the situation is similar. Concerning the IMDB dataset, both Google and GloVe show some stability (resistance) for the introduction of random vectors. This also illustrates that different types of pre-trained vectors have different coverage of words due to different training corpus. And Figure 5 will prove this point.

As for the CNN model, the situation is quite different. Since the model we applies is non-static(the pre-trained vectors will be re-tuned for each task), randomly initializing a word that does not exist in the pre-trained vocabulary is equivalent to a complement to the word matrix. For (randomly initialized) vectors not in pre-trained vector collections, fine-tuning allows them to learn more meaningful representations. Since the classifier is provided with more information, also the random vector itself is not static and fixed, the final performance will be better than SVM (IMDB: increase by 0.46 to 1.97 percent; 20 Newsgroup: 0.67 to 0.93 percent; Consumer Complaints: 0.45 to 0.72 percent). Notably, the result is only derived from variants using Google and Glove. For variants with SG and CBOW as input, the situation is the same as for SVM (basically unchanged).

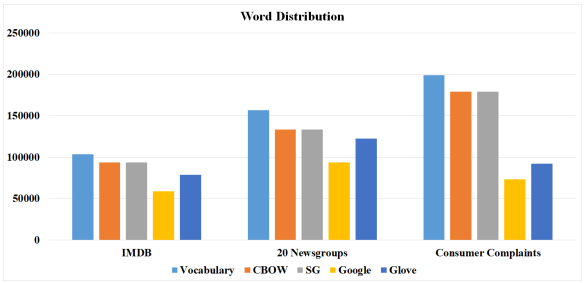
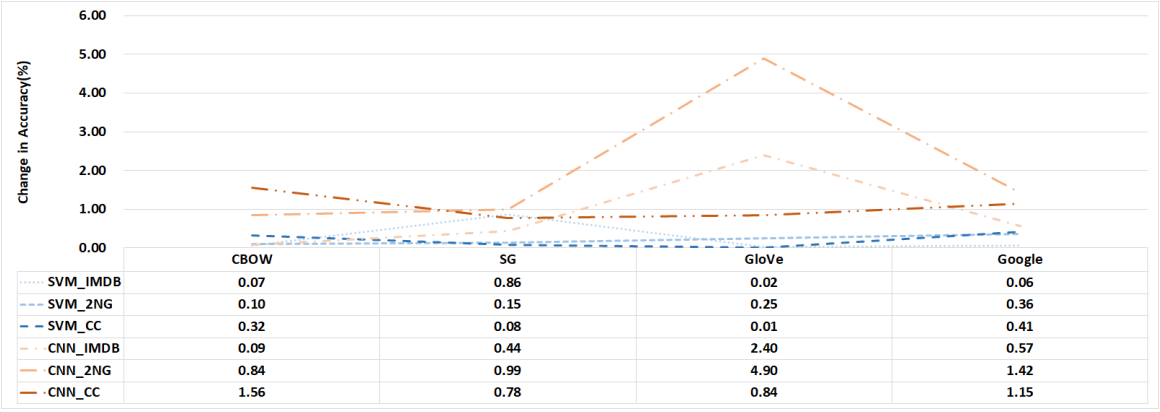


Figure 5: Sample of word distributions in different pre-trained vectors collections. Y-axis reflects the amount of words, while X-axis shows the results of different datasets.

5.2.4 Effect of TF-IDF weighting technique

Unlike the introduction of random vector mechanism, TF-IDF weighted vector has a positive effect on results, and this effect also varies depending on the classification type. Taking Linear SVM into consideration, for the binary classification, it is possible that these pre-training word vectors are already excellent enough for SVM to deal with the classification task. Therefore, whether weighted or not has little effect on the final performance. However, for multi-class classifications (regardless of whether the categories were evenly distributed), the weighted vector will increase the final accuracy by 0.2 to 5 percent point. The TF-IDF considers the influence of a word not only to a document, but also to the whole collection of corpus. Hence, the weighted vectors can add new



	CBOW	SG	GloVe	Google
SVM_IMDB	0.07	0.86	0.02	0.06
SVM_2NG	0.10	0.15	0.25	0.36
SVM_CC	0.32	0.08	0.01	0.41
CNN_IMDB	0.09	0.44	2.40	0.57
CNN_2NG	0.84	0.99	4.90	1.42
CNN_CC	1.56	0.78	0.84	1.15

Figure 6: Effect of TF-IDF weighting on three datasets. Y-axis reflects the change in Accuracy before and after applying TF-IDF weighting technique, while X-axis shows the results of different input representations.

power to the classifying ability of the model. The experimental results also proves this, regardless of the pre-trained vector, the weighted vector can improve a little accuracy more or less.

It can be found from both Figure 6 and Table 10 that the word2vec model weighted by TF-IDF achieves a positive influence on the performance. At the same time, the benefits of weighting vectors are more pronounced than completing *fullwords*. Also, on the other hand, CNN with TF-IDF weighted technique obtains more improvement than Linear SVM model.The reason can be attributed to the way of representing a document. In the SVM model, we add the word vectors together and got average value as a document vector(as Figure 7 shows). We analyze the adding and averaging operation will lose some specific details. Assume a word vector is very significant for a document, but the final document vector is the result of all vectors' average value. Then the importance of the word may decrease and affect the classification performance, even if we have adopted the TF-IDF weighting method. However, the CNN model uses a vector matrix to represent a document. Figure 8 is quoted from Ye Zhang[10] in his paper. It illustrates the architecture Zhang[10] has applied, which is also adopted by us. The specific values in the pictures are for demonstration purposes only and do not represent the parameters used in actual experiments. Here Zhang[10] faced binary classification task and then he assumed two potential output states. The only factor that may have an influence on the result(for the input phase) will be the matrix size, while the information hidden in the vectors will be included thoroughly.

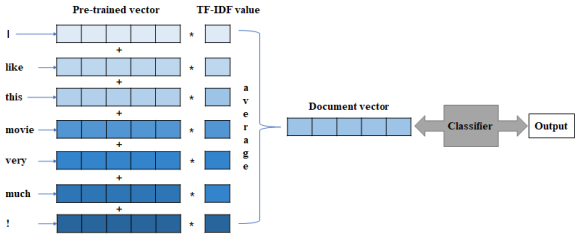


Figure 7: Representation of input on SVM model.

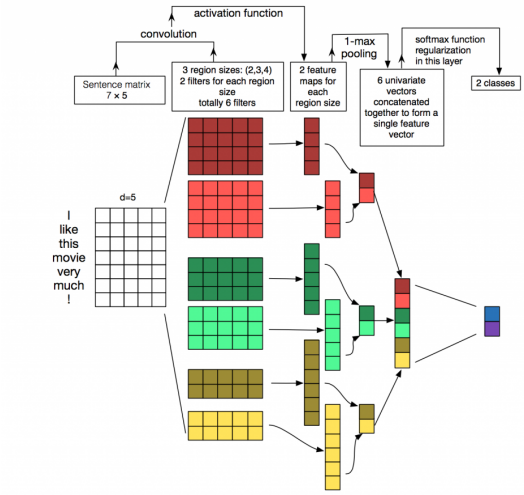


Figure 8: Representation of input on CNN model. Figure from Ye Zhang[10].

5.3 Limitations

It is obvious that tuning of CNN hyperparameters is a truly boring and demanding task. Normally, those parameters should be varied on different datasets. Therefore, it could be more complicated and demanding when referring to each task separately. Thanks

to the achievements of Kim[8] and Zhang[10], we did not waste much time doing the tuning task in our experiments. However, we still hold the view that fine-tuning will have an influence on the final performance. Moreover, we did not design experiments on SVMs with other kernels(like poly or RBF). Also, we did not focus on other kinds of neural network models like Recurrent Neural Networks(RNN). Although CNN can perform well in many tasks, it also has some disadvantages. For example, it cannot capture longer sequence information. The core idea of CNN is to obtain the feature representation of the document, while the more widely adopted in natural language processing is RNN. Compared with CNN, RNN is easier to better reflect context information. Therefore, it will be interesting to run similar experiments on different architectures, like RNN or joint CNN+RNN proposed by Wang[23] in 2016.

On the other hand, we also had certain limitations on the selection of datasets. We only selected the datasets of the binary and the multi-class classification, but we did not involve the dataset of the multi-label classification. Multi-label, also named multi-output classification is referring to the classification problem where multiple labels may be assigned to one certain sample[24]. This type of classification is much more complicated than a simple binary or n-ary classification. At the same time, it also tests the classification ability of the model. However, due to time constraints, we did not find a suitable multi-label dataset, we had no choice but to abandon this classification type temporarily.

5.4 Summary

Figure 9 shows the comparison of the best results of our experiments with our previous baseline results for three different data sets. It is apparent that the CNN model outperformed on all three datasets, while Multinominal Naive Bayes did achieve the worst results. But for the SVM model, its results were very unclear. BoW-based(using TF-IDF values) SVMs could achieve superior results in two out of three datasets, while the SVM based on word embedding only performed better in one dataset.

Basically, TF-IDF weighting would bring positive benefits to the final performance, while fullwords would only increase accuracy in the CNN model, and it brought a negative effect on the SVM model respectively. Hence the results of CNN in the above figure were all from the variants that introduce fullwords and tf-idf weighting, while the results of SVM_w2v belong to variants that applied weighting technique but not fullwords. As for the result of CNN model, most best results were obtained by 300-dimensions vectors of Google weighted by TF-IDF with fullwords technique. However, most best values of SVM model were achieved by the high dimensional(1200 or 2400) SG vectors weighted by TF-IDF but without fullwords technique. For the specific results, please refer to the Appendix section.

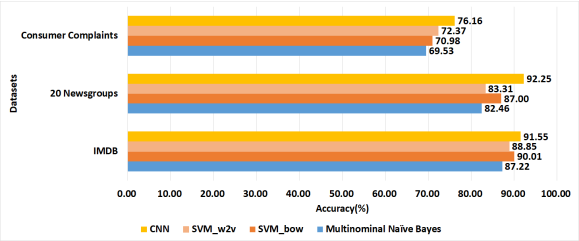


Figure 9: Comparison of the best results of our experiments and baseline with respect to four overall variants(input representations, dimensional size, fullwords technique, and TF-IDF weighting technique). Y-axis reflects the involved datasets, while X-axis shows the Accuracy they achieved.

6 Related Work

The essential purpose of our work is to verify the effectiveness of our proposed ideas(mentioned in section 3.2.4) and to extend the research on some existing work[8][10][9]. Table 4 shows the main differences between our experiments and other related work, mainly based on the research results of Yoon Kim[8] and Ye Zhang[10].

First of all, Kim[8] proposed a simple CNN architecture with little hyperparameter tuning that could achieve excellent results on sentiment analysis classification. And his work was quite similar to the architecture of Razavian et al.[25], which was designed for image classification task initially. By running various experiments with CNNs(improved upon the state of the art on 4 out of 7 datasets), Kim[8] approved that CNNs with pre-trained vector could perform remarkably well. Also, his simple architecture was applied widely by all the following works more or less.

But, there was little practical analysis to guide the decisions about setting various hyperparameters. And Zhang[10] tried to address this gap in his work in 2016. Then, a more comprehensive analysis of various hyperparameters was completed by Zhang[10] instead. His aim was to distinguish between important and comparatively inconsequential design decisions for sentence classification by applying the model proposed by Kim[8]. In addition to considering the multi-class classification, using Google's pre-trained

vector and some concatenating operations on vectors, he also introduced GloVe vector, which was proposed by Pennington et al.[7]. GloVe was the vector trained by involving nonzero elements in a word-word co-occurrence matrix, instead of the whole sparse vector matrix or the context windows in a large corpus. Meanwhile, Pennington[7] clarified that GloVe vectors could outperform other models on word analogy, word similarity, and named entity recognition tasks. In the end, Zhang[10] derived quite a few suggestions regarding CNN architecture and hyperparameters(input word vector, filter region size, number of feature maps, activation function, pooling strategy, and regularization).

On the other hand, the research of Mounir Hader[9] was also relevant to our ideas. He tried to investigate the practical use of CNN for text classification by comparing with other traditional models like SVMs and Naive Bayes. He drew the conclusion that without the tuning job, CNN was sufficient to dealing with classification tasks. Moreover, conventional methods like SVM could obtain similar achievement as his CNN model on 2 out of 3 datasets. However, his work only dealt with the binary classification task, and he did not consider the word vector representation and dimensional diversity. In addition, he did not find the impact of word embedding on other classifiers, For example, in his research, the experiment of CNN+vector(based on Google News of word2vec) was carried out, however, the possibility of SVM+vector(word2vec) was not considered. Then, it could be unclear if his achievements was attributed to the model itself or input representation(dimension reduction, and word embedding).

Combined with those existing achievement and limitations, our experiments would cover more thoroughly. Initially, we extended the word vector types to self-trained vectors by adopting word2vec proposed by Mikolov[6] (CBOW and SG). In addition, we have extended the consideration of the vector dimension from 300 to 2400. Even for vectors of the same size, we not only had vector concatenating operation but also trained vectors of this length directly. Furthermore, both Kim[8] and Zhang[10] considered initialize the OOV words randomly for CNN model but ignored the influence of OOV words for SVM model. However, we also considered the relevant experiments of the SVM model (although the final performance may not as satisfied as CNN model). The last row in Table 4 is a weighting method we proposed(mentioned in section 3.2.4). Through our experiments, we could see that the word2vec model based on TF-IDF weighting had a good performance on all classifiers. This also verifies the effectiveness of the combined method of generating improved word representations in the document classification task.

	Yoon Kim	**Ye Zhang**	**Our work**
Type	binary	multi-class	multi-class
Input	1	2	4
Dimension	300	300	300 to 2400
Fullwords	CNN	CNN	SVM & CNN
TF-IDF	No	No	Yes

Table 4: Differences between our experiments and related work. Type means the type of classification task, while Input shows the choice of trained embeddings(Google, GloVe, SG, and CBOW) as input. Dimension reflects the vector size, and Fullwords represents the technique that considers the existence of OOV words. Last, TF-IDF refers to the hybrid method that combines word embedding and TF-IDF.

7 Conclusion

We have run several extended experiments of SVM and CNN for document classification with respect to many aspects. In this section, We made some conclusions by summarizing our main findings and deriving from this practical information for those who are looking to implement classification problems.

Given by the hypotheses we proposed in section 3.6, here we summarize some main empirical findings:

- Using TF-IDF to weight pre-trained vector does help increase the performance for both SVM and CNN models. However, the improvement influence varies from model to model, and the effect of CNN in our experiments is significantly better than the SVM model. This may also be due to the way we choose to represent a document. As we discussed in section 5.2.4(Efect of TF-IDF weighting technique), we have taken the mean operation in the SVM model, which more or less interferes with the impact of the weighting method.
- Since the length of the Google and GloVe vectors is fixed, and we cannot change them. Hence we only draw this conclusion based on self-trained vectors(CBOW and SG). High-dimensional word vectors do bring higher accuracy, and this phenomenon is particularly evident in the SVM model. But unfortunately, adopting a higher-dimensional vector means longer training time and calculation process. In general, a 300-dimensional vector is sufficient for the CNN model, while the SVM model is not enough. If a better classification performance is required, we recommend not using vectors with more than 1200 dimensions. Because an oversized vector

is not only quite demanding, but it can also cause side effect like over-fitting, which reduces accuracy.

- For OOV(out-of-vocabulary) words, randomly generating their vectors will become noise interference terms for the SVM model, however not for the CNN model. Therefore, we hold the view that when using SVM for classification, it is wise to drop the OOV words, but they can be taken into account in the CNN model. If existing pre-trained vectors (Google and GloVe) are applied, it is difficult to guarantee that the word vectors they provide cover most of the words in the vocabulary. Then, when using the CNN model, it is a good choice to consider the existence of OOV words. On the other hand, if self-trained vectors are chosen, whether to consider OOV words becomes unnecessary since they are likely to cover the most words respectively.
- Vectors trained by learning from outside knowledge(Google and GloVe) seems to work better then self-trained(CBOW and SG). Word vectors trained based on the large corpus can indeed generate better vectors. More corpus information gives them more knowledge, which is why vectors learning from external resources perform better than self-trained vectors. But at the same time, the shortcomings are also obvious. It is hard to ensure that Google and GloVe can cover most of the words in the vocabulary, especially for some personal comments, movie reviews and other speech that are more casual and informal. As for this type of document, the pre-trained word vector is somewhat weak and not as satisfied as self-trained vectors. In this case, the self-trained word vector can achieve higher accuracy at the beginning(without weighting). However, since those well-trained vectors are easy to access online, which means it saves a lot of training time indeed. We still recommend using those existing vectors pre-trained by others, especially for those huge and demanding work. On the other hand, for those who want to train their own vectors, then vectors based on the SG model is recommended. However, it will cost more time than CBOW model to obtain the better vectors respectively.

7.1 Future work

Other than the lack of various selection of the model we mentioned in the Limitations section, we can also refer to more methods of document representation. For example, the doc2vec method proposed by Tomas Mikolov[26] can be the choice. Doc2vec is a method that derives from the adaptation of word2vec. Instead of generating the word vectors, doc2vec intends to represent a document directly. This algorithm simplifies the procedures since there is no need to use word vectors to encode a document, while doc2vec will output the document representation directly. And we also tried to do a comparison experiment between doc2vec and word2vec, but doc2vec is a non-deterministic vector representation. Precisely speaking, every time we use doc2vec to represent a document, we may get different vectors. Because the final result was not that stable, we gave up on this method. But in future experiments, we may wish to consider optimizing doc2vec to achieve a satisfying result. Also, our weighting method is mainly for word embeddings, but we also see that the traditional SVM+BoW model with TF-IDF values can perform well in many cases and it takes less time as well. We consider that we can try to improve the traditional BoW vector in the future. It is well known that BoW vectors are too sparse, and there are many places in the vector where the value is 0 (0 means that the word does not appear in this document). Then we can consider using word2vec to find the "existing words"(in this document) which are most similar to these "non-existing words"(values of 0). Furthermore, those meaningless 0 can be replaced with other values. And we hold the view that it will be likely to improve the expressive power of BoW vectors.

References

[1] A. Rajaraman, J.D. Ullman, *Data Mining* (Cambridge University Press, 2011), p. 1–17

[2] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, *Distributed Representations of Words and Phrases and their Compositionality*, arXiv e-prints arXiv:1310.4546 (2013), 1310.4546

[3] R. Lebret, R. Collobert, *Word Emdeddings through Hellinger PCA*, arXiv e-prints arXiv:1312.5542 (2013), 1312.5542

[4] O. Levy, Y. Goldberg, *Neural word embedding as implicit matrix factorization*, in *Advances in neural information processing systems* (2014), pp. 2177–2185

[5] Y. Li, L. Xu, F. Tian, L. Jiang, X. Zhong, E. Chen, *Word embedding revisited: A new representation learning and explicit matrix factorization perspective*, in *Twenty-Fourth International Joint Conference on Artificial Intelligence* (2015)

[6] T. Mikolov, K. Chen, G. Corrado, J. Dean, *Efficient Estimation of Word Representations in Vector Space*, arXiv e-prints arXiv:1301.3781 (2013), 1301.3781

[7] J. Pennington, R. Socher, C.D. Manning, *GloVe: Global Vectors for Word Representation*, in *Empirical Methods in Natural Language Processing (EMNLP)* (2014), pp. 1532–1543, http://www.aclweb.org/anthology/D14-1162

[8] Y. Kim, *Convolutional neural networks for sentence classification*, arXiv preprint arXiv:1408.5882 (2014)

[9] M. Hader, *A Comparative Study of Naive Bayes, SVMs and Convolutional Neural Networds for Binary Text Classification* (2017), research project of Leiden Institute of Advanced Computer Science(LIACS)

[10] Y. Zhang, B. Wallace, *A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification*, arXiv e-prints arXiv:1510.03820 (2015), 1510.03820

[11] H.P. Luhn, *Key word-in-context index for technical literature (kwic index)*, American Documentation **11**, 288 (1960)

[12] K. Sparck Jones, *A statistical interpretation of term specificity and its application in retrieval*, Journal of documentation **28**, 11 (1972)

[13] A. Rajaraman, J.D. Ullman, *Data Mining* (Cambridge University Press, 2011), p. 1–17

[14] A.L. Maas, R.E. Daly, P.T. Pham, D. Huang, A.Y. Ng, C. Potts, *Learning Word Vectors for Sentiment Analysis*, in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (Association for Computational Linguistics, Portland, Oregon, USA, 2011), pp. 142–150, http://www.aclweb.org/anthology/P11-1015

[15] K. Lang, *Newsweeder: Learning to filter netnews*, in *Proceedings of the Twelfth International Conference on Machine Learning* (1995), pp. 331–339

[16] R. Kohavi, *A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection* (Morgan Kaufmann, 1995), pp. 1137–1143

[17] S. Arlot, A. Celisse, *A survey of cross-validation procedures for model selection*, Statist. Surv. **4**, 40 (2010)

[18] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard et al., *TensorFlow: A System for Large-Scale Machine Learning*, in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (USENIX Association, Savannah, GA, 2016), pp. 265–283, ISBN 978-1-931971-33-1, https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

[19] F. Chollet, *Xception: Deep Learning with Depthwise Separable Convolutions*, arXiv e-prints arXiv:1610.02357 (2016), 1610.02357

[20] R. Řehůřek, P. Sojka, *Software Framework for Topic Modelling with Large Corpora*, in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* (ELRA, Valletta, Malta, 2010), pp. 45–50, http://is.muni.cz/publication/884893/en

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg et al., *Scikit-learn: Machine learning in Python*, Journal of machine learning research **12**, 2825 (2011)

[22] S. van der Walt, S.C. Colbert, G. Varoquaux, *The NumPy Array: A Structure for Efficient Numerical Computation*, Computing in Science Engineering **13**, 22 (2011)

[23] X. Wang, W. Jiang, Z. Luo, *Combination of convolutional and recurrent neural network for sentiment analysis of short texts*, in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers* (2016), pp. 2428–2437

[24] J. Read, B. Pfahringer, G. Holmes, E. Frank, *Classifier chains for multi-label classification*, Machine Learning **85**, 333 (2011)

[25] A. Sharif Razavian, H. Azizpour, J. Sullivan, S. Carlsson, *CNN features off-the-shelf: an astounding baseline for recognition*, in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (2014), pp. 806–813

[26] Q.V. Le, T. Mikolov, *Distributed Representations of Sentences and Documents*, arXiv e-prints arXiv:1405.4053 (2014), 1405.4053

[27] Y. LeCun, Y. Bengio, G. Hinton, *Deep learning*, nature **521**, 436 (2015)

[28] D.P. Kingma, J. Ba, *Adam: A Method for Stochastic Optimization*, arXiv e-prints arXiv:1412.6980 (2014), 1412.6980

[29] V. Nair, G.E. Hinton, *Rectified Linear Units Improve Restricted Boltzmann Machines*, in *Proceedings of the 27th International Conference on International Conference on Machine Learning* (Omnipress, USA, 2010), ICML'10, pp. 807–814, ISBN 978-1-60558-907-7, http://dl.acm.org/citation.cfm?id=3104322.3104425

A Word2Vec & GloVe

A.1 Word2Vec

In the spirit of the Neural Network Language Model(NNLM) proposed by Bengio[27], Mikolov et al.[6] came up with the word2vec model in 2013. By utilizing a very shallow neural network, word2vec is designed to generate the word embeddings in an efficient way. Generally speaking, word2vec is a prediction-based method which involves two main models. Figure 10 shows the difference between the two models. One model is named continuous bag of words(CBOW), while another is called skip-gram(SG). CBOW model uses the contextual words as input and utilizes them to predict the possibility of central word. However, SG model works in a reverse way. SG uses a central word to predict the contextual words of it. With respect to the difference of training procedure, the performance and computational time could differ from models. To some extents, CBOW model could be less professional than SG model. All the update of contextual words is shared by the knowledge of the central word, which means one specific mount of information will be divided into pieces and each contextual words will obtain some of them. However, as for the SG model, the central word will be modified by the effort of all the corresponding contextual words. Therefore, CBOW model could be more proper for small text data, while SG model could get better embeddings in a large dataset instead.

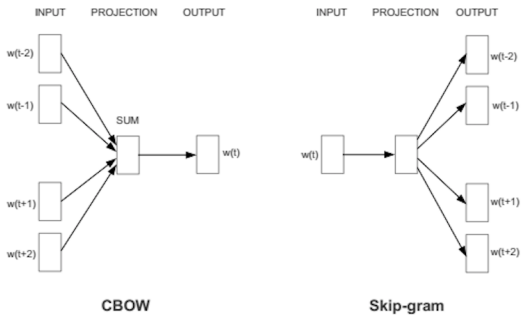


Figure 10: CBOW and Skip-gram models of Word2Vec

With respect to the output of word2vec, also known as word embeddings, they make the word vectors meaningful comparing with the traditional BoW vectors. With the help of word2vec, the vector itself can reflect some relationship between words. For instance, it can be used to find the most similar word in the vector space. Even for the word formula, such as *King - Man + Woman = Queen*, can be realized by word2vec as well. Therefore, it is obvious that the word vector is very useful for expressing the semantic features of the word.

A.2 GloVe

GloVe is an unsupervised learning algorithm for obtaining vector representations for words[7]. Unlike the theory of word2vec mentioned in Appendix A.1, GloVe is a method based on the word-word co-occurrence matrix. There is an example of word-word co-occurrence matrix shown in Table 5. As for the notation of matrix, $P_{ij} = P(j|i) = X_{ij}/X_i$ where X_{ij} shows the number of times word j occurs in the context of word i, and X_i illustrates the overall time of occurrence of all corresponding contextual words of i. Then the ratio of two different words and one mutual word can also reflect some interesting relationships between words. For instance, assume *solid* is the mutual word which seems relevant to *ice* than *steam*, the ratio can be greater than 1. However, if the mutual word is *gas* which is closer to *steam* instead of *ice*, then the ratio can be smaller than 1 relatively. But if the mutual word is *water* or *fashion*, then the ratio can be around 1. This means that words like *water* or *fashion* is either relevant to both *ice* and *steam* or irrelevant to them. Based on this situation, Pennington[7] tried to propose a model to reflect it. This is the theory of Global Vectors for Word Representation, also known as GloVe. And according to the statement of Jeffrey Pennington[7], for the same corpus with similar training time, GloVe consistently outperforms word2vec.

B Datasets

Given the apparent differences in class quantity, document length, vocabulary size, and document complexity, these datasets are more proper for comparing various classification types(binary or multi-class classification). Samples of different datasets will be listed as followings, where the order is: IMDB, 20 Newsgroups, and Consumer Complaints. Due to the space limitation, some contents are omitted which is replaced by an ellipsis.

In 1974, the teenager Martha Moxley (Maggie Grace) moves to the high-class area of Belle Haven, Greenwich, Connecticut. On the Mischief Night, eve of Halloween, she was murdered in the backyard of her house and her murder remained unsolved. Twenty-two years later, the writer Mark Fuhrman (Christopher Meloni), who is a former LA detective that has fallen in disgrace for them in O.J. Simpson trial and moved to Idaho, decides to investigate the case with his partner Stephen Weeks (Andrew Mitchell) with the purpose of writing a book. The locals squirm and do not welcome them, but with support of the retired detective Steve Carroll (Robert Forster) that was in charge of the investigation in the 70s they discover the criminal and a net of power and money to cover the murder."Murder in Greenwich" is a good TV ::: ::: ::: ::: ::: :::

Probability and Ratio	k = solid	k = gas	k = water	k = fashion
$P(k\|ice)$	$1.9\text{x}10^{-4}$	$6.6\text{x}10^{-5}$	$3.0\text{x}10^{-3}$	$1.7\text{x}10^{-5}$
$P(k\|steam)$	$2.2\text{x}10^{-5}$	$7.8\text{x}10^{-4}$	$2.2\text{x}10^{-3}$	$1.8\text{x}10^{-5}$
$P(k\|ice)/P(k\|steam)$	8.9	$8.5\text{x}10^{-2}$	1.36	0.96

Table 5: Co-occurrence probabilities based on a 6 billion token corpus. Table is quoted from Pennington[7].

From: Mamatha Devineni Ratnam<mr@andrew.edu>
Subject: Pens fans reactions
Organization:
Post Office, Carnegie Mellon, Pittsburgh, PA
Lines: 12 NNTP-Posting-Host: po4.andrew.cmu.edu
I am sure some bashers of Pens fans are pretty confused about the lack of any kind of posts about the recent Pens massacre of the Devils. Actually, I am bit puzzled too and a bit relieved. However, I am going to put an end to non-PIttsburghers' relief with a bit of praise for the Pens. Man, they are killing those Devils worse than I thought. Jagr just showed you why he is much better than his regular season stats. He is also a lot of fun to watch in the playoffs. Bowman should let JAgr have a lot of fun in the next couple of games since the Pens are going to beat the pulp out of Jersey anyway.
::: ::: :::
::: ::: :::

"I have been trying to open a small XXXX business XX/XX/XXXX and had worked with XXXX of Chase Bank in XXXX XXXX, WI for business funding.
our relations she promised me she was diligently going see if she could get my business funding with Chase, I was asking for a XXXX loan. After waiting weeks for a return call regarding her promise, I never heard back her to this day. Without funding from Chase, we were with no choice to get a loan, WI (formerly known as Bank) for 33 % interest and our 2 motorcycles as collect Our thoughts are to pay that loan in the shortest time possible once we are able to open our doors for revenue. This money now was not only the money we needed but it also has $ XXXX of our property tied to it. 4 days before we were to secure the loan I opted to open my business checking account with XXXX at the 2nd Chase branch, WI. We did not want to sign for that loan but we needed to, to complete my business plan buy
::: ::: :::
::: ::: :::

C Evaluation Metrics

In order to make it easier to understand those equations, we first use Table 6 to illustrate some essential notations. The first metric we use is the *accuracy*. It is the ratio of the number of all correct classes to the number of all the wrong classes.

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{6}$$

Then we use the *precision* to represent the ratio of the number of samples with the correct classification to the number of all classified texts in this predicted class.

$$precision = \frac{tp}{tp + fp} \tag{7}$$

Similar to *precision*, *recall* is the ratio of the number of samples with the correct classification to the actual number of texts in this real class.

$$recall = \frac{tp}{tp + fn} \tag{8}$$

As for *F1-score*, it is an indicator that considers both *precision* and *recall*.

$$F1 - score = 2 * \frac{precision * recall}{precision + recall} \tag{9}$$

		Predicted	
		pos/c class	*neg/non-c class*
True	*pos/c class*	tp	fn
	neg/non-c class	fp	tn

Table 6: Definition of TP,TN,FP and FN.

D Hyperparameters of CNN model

Table 7 represents the most significant parameters set in the CNN model. Those parameters with symbol * are the new tuneable parameters we introduce. As for other parameters, most of them are the same in the mentioned papers written by Yoon Kim[8], Ye Zhang[10], and Mounir Hader[9]. But, all the related work is set for binary classification and small datasets. Therefore, we decide to increase several values in order to obtain a relatively higher performance on larger and multi-class classification issue. For instance, we add **HD**, **NF** and **BS** from (0,100,64) to (128,128,128). And we set cross-entropy loss as our objective function which should be minimized. Adam(short for Adaptive Moment Estimation)[28] is used as the optimization algorithm. We also applied early stopping scheme in order to save the computational time. This mechanism means the machine will stop training if the model gets No optimization over 1000 steps. However, aiming to avoid the accident, we will save the best-observed model after each epoch, which will be used for evaluation later. In addition, different from Kim[8], our model will contain a hidden layer. Therefore, we choose ReLu(short for Rectified linear units)[29] as activation function for convolutional layer.

Param.	Definition	Baseline
HD	*Hidden Dimensions*: is the number of neurons in hidden layer	128
FS	*Filter Size*: the heights of the kernels	3,4,5
DP	*Dropout Probability*: drop out the neurons with a certain probability	0.5
NF	*Number of Filters*: the number of filters applied for each filter region	128
BS	*Batch Size*: size of the mini-batches	128
PM	*Pooling method*: scheme to apply to the feature map	global max
UB	*Use Bias*: whether to add a bias term after the convolutional layer	yes
LR	*Learning Rate*: determines to what extent the latest information overrides the previous one	1e-3
DR	*Decay Rate*: indicates the extent of reducing the learning rate	0.9
AF	*Activation Function*: is used to add nonlinear factors	ReLu
BW	*Balance Weight*: whether to adjust class weights inversely proportional to the class frequencies	no
***WV**	*Word Vector*: resources of input word vectors	pre-trained
***VD**	*Vector Dimension*: dimension of vectors as input	300
***TW**	*Tf-idf Weighted*: whether to use the weighted vector by tf-idf values	no
***FW**	*Fullwords*: whether to randomly initialize an nonexistent word	no
***VF**	*Vocabulary Filter*: whether to trim the vocabulary	yes

Table 7: Part of Tuneable Hperparameters in CNN model

E Tables of Results

Table 8 reflects the results of Multinomial Naive Bayes and Linear SVM classifications based on sublinear tf-idf. These results are also obtained by Mounir Hader[9] in his work. Table 9 and 10 represent the performance of both Linear SVM and CNN based on different input representations and variants(*fullwords* and TF-IDF weighting techniques). As for the results of different models based on different input dimensions, they can be found in Table 11 and 12 respectively.

Model	N-grams	IMDB				20 News Groups				Consumer Complaints			
		Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1
Multinomial N.B. sublinear tf-idf	uni	82.50	85.14	78.88	81.88	81.61	81.54	81.61	81.43	67.87	66.84	67.48	67.11
	uni+bi	84.87	87.45	83.46	85.11	**82.46**	**82.62**	**82.46**	**82.44**	68.11	69.23	68.88	68.50
	uni+bi+tri	**87.22**	**88.15**	**86.55**	**85.48**	80.35	80.52	80.35	80.34	**69.53**	**69.20**	**68.93**	**69.55**
Linear SVM sublinear tf-idf	uni	88.78	88.10	88.98	88.98	85.90	86.11	85.90	85.80	70.13	69.99	70.59	70.23
	uni+bi	**90.01**	**89.55**	**90.88**	**90.12**	**87.00**	**87.19**	**87.00**	**86.87**	70.67	70.53	70.89	70.46
	uni+bi+tri	89.73	88.92	90.10	89.68	86.74	86.94	86.74	86.60	**71.52**	**70.98**	**71.03**	**71.46**

Table 8: Results of Multinomial Naive Bayes and Linear SVM classifications based on sublinear tf-idf

Model	Pre-trained	IMDB				20 News Groups				Consumer Complaints			
		Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1
Linear SVM word2vec	CBOW	84.81	84.82	84.81	84.81	70.50	70.72	70.42	70.37	67.62	67.53	68.96	66.37
	SG	**86.31**	**86.31**	**86.31**	**86.31**	**78.28**	**78.35**	**78.29**	**78.19**	**68.36**	**68.28**	**69.62**	**67.17**
	GloVe	85.89	85.89	85.89	85.89	76.61	76.65	76.78	76.41	63.69	63.75	65.46	61.86
	Google	85.34	85.34	85.34	85.34	73.97	74.03	74.31	73.58	62.62	62.53	64.56	60.77
Linear SVM word2vec (full words)	CBOW	84.81	-	-	-	70.50	-	-	-	67.62	-	-	-
	SG	**86.31**	**-**	**-**	**-**	**78.28**	**-**	**-**	**-**	**68.36**	**-**	**-**	**-**
	GloVe	85.75	85.75	85.75	85.75	74.45	74.56	74.61	74.19	62.97	63.06	64.84	61.02
	Google	83.86	83.86	83.86	83.86	61.40	60.92	62.16	61.12	59.17	58.73	61.80	56.98
Linear SVM tfidf weighted word2vec	CBOW	84.88	84.89	84.88	84.88	70.60	70.77	70.57	70.47	67.94	68.00	69.16	66.67
	SG	**87.17**	**87.11**	**87.20**	**87.20**	**78.42**	**78.49**	**78.44**	**78.34**	**68.44**	**68.44**	**69.62**	**67.25**
	GloVe	85.91	85.92	85.91	85.91	77.46	77.48	77.63	77.26	63.70	63.81	65.47	61.82
	Google	85.40	85.40	85.40	85.40	74.04	74.13	74.38	73.62	63.03	63.34	64.72	61.03
Linear SVM tfidf weighted word2vec (full words)	CBOW	84.88	-	-	-	70.60	-	-	-	67.94	-	-	-
	SG	**87.17**	**-**	**-**	**-**	**78.42**	**-**	**-**	**-**	**68.44**	**-**	**-**	**-**
	GloVe	85.79	85.79	85.79	85.79	74.94	74.88	75.17	74.76	62.96	63.02	64.86	61.01
	Google	84.21	84.21	84.21	84.21	61.74	60.94	62.28	61.19	59.35	59.81	61.51	56.72

Table 9: Results of Linear SVM classification based on different input representations

Model	Pre-trained	IMDB				20 News Groups				Consumer Complaints			
		Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1
CNN word2vec	CBOW	89.39	89.40	89.38	89.38	84.26	84.65	84.01	84.13	72.14	71.91	72.64	71.84
	SG	**90.47**	**90.48**	**90.47**	**90.47**	88.00	88.06	87.96	87.97	74.71	74.53	75.08	74.51
	GloVe	88.19	88.21	88.18	88.18	85.70	85.80	85.68	85.62	73.92	73.70	74.37	73.70
	Google	90.03	90.04	90.03	90.03	**89.66**	**89.90**	**89.50**	**89.58**	**74.57**	**74.31**	**74.97**	**74.43**
CNN word2vec (full words)	CBOW	89.39	-	-	-	84.26	-	-	-	72.14	-	-	-
	SG	90.47	-	-	-	88.00	-	-	-	74.71	-	-	-
	GloVe	90.16	90.17	90.16	90.16	86.63	86.77	86.55	86.57	74.38	74.07	74.79	74.27
	Google	**90.49**	**90.50**	**90.49**	**90.49**	**90.33**	**90.74**	**90.03**	**90.21**	**75.29**	**75.02**	**75.67**	**75.19**
CNN tfidf weighted word2vec	CBOW	89.48	89.49	89.47	89.47	85.10	85.32	84.99	85.00	73.70	73.48	74.04	73.57
	SG	**90.91**	**90.95**	**90.89**	**90.89**	88.98	89.19	88.83	88.93	75.48	75.22	75.84	75.39
	GloVe	90.59	90.60	90.59	90.59	90.60	90.90	90.39	90.51	74.77	74.57	75.14	74.59
	Google	90.60	90.63	90.59	90.59	**91.08**	**91.27**	**90.95**	**91.03**	**75.72**	**75.56**	**75.99**	**75.61**
CNN tfidf weighted word2vec (full words)	CBOW	89.48	-	-	-	85.10	-	-	-	73.70	-	-	-
	SG	90.91	-	-	-	88.98	-	-	-	75.48	-	-	-
	GloVe	90.83	90.85	90.82	90.82	91.78	91.80	91.78	91.75	74.92	74.76	75.19	74.82
	Google	**91.55**	**91.56**	**91.55**	**91.55**	**92.25**	**92.34**	**92.20**	**92.21**	**75.93**	**75.76**	**76.23**	**75.80**

Table 10: Results of CNN classification based on different input representations

Model	Pre-trained	IMDB				20 News Groups				Consumer Complaints			
		Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1
Linear SVM unweighted	CBOW_600	87.10	87.10	87.10	87.10	73.76	73.93	73.69	73.65	69.34	69.11	70.46	68.45
	CBOW_1200	87.16	87.16	87.16	87.16	74.52	74.67	74.47	74.43	70.31	70.06	71.27	69.61
	SG_600	87.56	87.56	87.56	87.56	82.20	82.29	82.20	82.11	69.45	69.22	70.58	68.56
	SG_1200	**87.59**	**87.59**	**87.59**	**87.59**	**82.58**	**82.65**	**82.61**	**82.49**	**70.61**	**70.36**	**71.58**	**69.89**
	Google&GloVe_600	86.80	86.80	86.80	86.80	76.66	76.72	76.77	76.50	65.50	65.26	67.10	64.15
	CBOW&SG_600	86.91	86.91	86.91	86.91	74.89	74.90	75.13	74.64	69.64	69.56	70.68	68.68
	CBOW&SG_1200	**88.73**	**88.74**	**88.73**	**88.73**	78.65	78.81	78.61	78.54	70.62	70.36	71.56	69.94
	CBOW&SG_2400	88.14	88.15	88.13	88.13	**79.41**	**79.53**	**79.37**	**79.32**	**71.36**	**71.01**	**72.21**	**70.87**
Linear SVM weighted	CBOW_600	87.12	87.12	87.12	87.12	74.02	74.16	74.00	73.90	70.56	69.10	71.95	69.18
	CBOW_1200	87.33	87.33	87.33	87.33	74.57	74.71	74.51	74.48	71.41	70.18	71.41	70.25
	SG_600	88.32	88.32	88.32	88.32	82.63	82.71	82.65	82.53	70.72	69.34	71.59	69.43
	SG_1200	**88.53**	**88.53**	**88.53**	**88.53**	**83.31**	**83.37**	**83.35**	**83.22**	**71.63**	**71.00**	**72.41**	**71.58**
	Google&GloVe_600	87.32	87.32	87.32	87.32	77.05	77.09	77.19	76.88	67.00	66.41	67.71	64.52
	CBOW&SG_600	87.17	87.17	87.17	87.17	75.42	75.46	75.58	75.21	70.57	69.58	71.65	69.62
	CBOW&SG_1200	**88.85**	**88.85**	**88.85**	**88.85**	79.12	79.29	79.05	79.01	71.68	70.72	72.30	70.42
	CBOW&SG_2400	88.24	88.24	88.24	88.24	**79.61**	**79.71**	**79.59**	**79.53**	**72.37**	**71.87**	**72.79**	**71.83**

Table 11: Results of Linear SVM classification based on different input dimensions

Model	Pre-trained	IMDB				20 News Groups				Consumer Complaints			
		Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1	Acc.	Pre.	Rec.	F1
CNN unweighted	CBOW_600	85.74	85.75	85.73	85.73	84.00	84.15	83.93	83.92	71.29	71.19	71.94	70.73
	CBOW_1200	86.80	86.81	86.80	86.80	86.72	86.88	86.74	86.53	72.69	72.51	73.19	72.37
	SG_600	89.53	89.53	89.53	89.53	88.35	88.46	88.28	88.31	74.59	74.30	75.00	74.47
	SG_1200	**89.55**	**89.60**	**89.53**	**89.53**	**89.29**	**89.50**	**89.15**	**89.22**	**74.83**	**74.69**	**75.09**	**74.72**
CNN weighted	CBOW_600	89.23	89.24	89.23	89.23	85.06	85.19	84.99	85.00	73.55	73.41	74.06	73.17
	CBOW_1200	89.34	89.35	89.33	89.33	87.14	87.39	86.98	87.06	73.88	73.66	74.25	73.72
	SG_600	**90.75**	**90.77**	**90.74**	**90.74**	89.55	89.66	89.50	89.50	**76.16**	**75.92**	**76.48**	**76.09**
	SG_1200	90.45	90.46	90.45	90.45	**90.11**	**90.22**	**90.03**	**90.07**	75.41	75.32	75.65	75.26

Table 12: Results of CNN classification based on different input dimensions