Movie Review Sentiment Analysis based on Word2Vec and Doc2Vec

Eric Wang
Department of Computer Science
University of Victoria
xhwang@uvic.ca

Henry Jiang
Department of Computer Science
University of Victoria
ziangjiang@uvic.com

Zhixiang Li
Department of Computer Science
University of Victoria
lizhixiang27@gmail.com

Chengxu Zhang
Department of Computer Science
University of Victoria
chengxuzhang123@gmail.com

Tong Zhang
Department of Computer Science
University of Victoria
zt55699@gmail.com

Abstract

Movie reviews are an important way to gauge the performance of a movie. While providing a numerical/stars rating to a movie tells us about the success or failure of a movie quantitatively, a collection of movie reviews is what gives us a deeper qualitative insight on different aspects of the movie. A textual movie review tells us about the the strong and weak points of the movie and deeper analysis of a movie review can tell us if the movie in general meets the expectations of the reviewer. Sentiment Analysis is a major subject in machine learning which aims to extract subjective information from the textual reviews. The field of sentiment of analysis is closely tied to natural language processing and text mining. It can be used to determine the attitude of the reviewer with respect to various topics or the overall polarity of review. Using sentiment analysis, we can find the state of mind of the reviewer while providing the review and understand if the person was "happy", "sad", "angry" and so on. In this project, we aim to use Sentiment Analysis on a set of movie reviews given by reviewers and try to understand what their overall reaction to the movie was,i.e.if they liked the movie or they hated it. We would train some machine learning models using features extracted by Word2Vec and make the predictions. Also, a detailed analysis on Word2Vec would be provided. In other words, we aim to utilize the relationships of the words in the review to predict the overall polarity of the review.

1 Introduction

Movie review is the analysis and evaluation of movie quality. A movie review implies a recommendation aimed at consumers. Most websites related to movie allow Internet users to submit reviews with or without ratings and aggregate them into an overall score, so reviews in websites like IMDB are treasures for film companies to know if they made a good movie and for movie fans to determine if they want to watch the movie. Having had a full understanding of audiences' attitude to the film from the sentiments of reviews, film companies can adjust the following issuance strategy and optimize

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020).

the issuance process, and fans can have a prediction to the movie quality. But the problem is that there exists a large amount of reviews without ratings and a binary evaluation. film makers and fans both can't extract the sentiment of reviews from words and text directly so that they need to read and analyze the reviews, which is a time and energy consuming work. Therefore, we want to find a way to evaluate the sentiment of reviews without ratings to make the use of reviews easy and convenient.

Sentiment represents an overall opinion or personal feeling that is embed in a word or a piece of sentence and usually relates to personal preference. The aim of our project is to use machine learning methods to learn and predict whether a movie review is positive or negative. There are several reasons why we combine movie review sentiment analysis with machine learning together. At first, some members of this team are movie fans, and reading and extracting the sentiment from reviews are a part of their daily life. Secondly, recently proposed techniques used to represent words as vectors make it easier to use machine learning methods to process tasks with data composed of text, instead of numbers. Additionally, we all have a strong interest to NLP and sentimental analysis, so this project is a perfect starting point for us to research further on more complicated and valuable problems, such as predicting the possible rating of a rating-less review, in the future.

In this project, we would train some machine learning models using features extracted based on Word2Vec and make the predictions given this features. Also, a detailed analysis on Word2Vec would be provided. In other words, upon extracting features based on Word2Vec, we aim to utilize the relationships of the words in the review to predict the overall polarity of the review.

1.1 Related Work

Acosta u.a. ((2017)) state that Word2Vec are two novel model architectures for computing continuous vector representations of words from very large data sets. Chen u.a. ((2018)) announce that Word2vec is a group of two-layer neural networks that are used to produce word embedding. In our project, we are primarily focusing on understanding, implementing and using this architecture to solve an actual problem.

There are some publications with sentiment analysis that we have finished reading. For example, Nawangsari u.a. ((2019)) proposed a good method to use Word2Vec on sentiment analysis. Liu ((2017)) showed a hotel review analysis pipeline very similar to our project. Giatsoglou u.a. ((2017)) proposed an effective method for sentiment lexical dictionary enrichment based on Word2Vec for sentiment analysis. Our Word2Vec architecture implementation is based on the knowledge learnt from these materials.

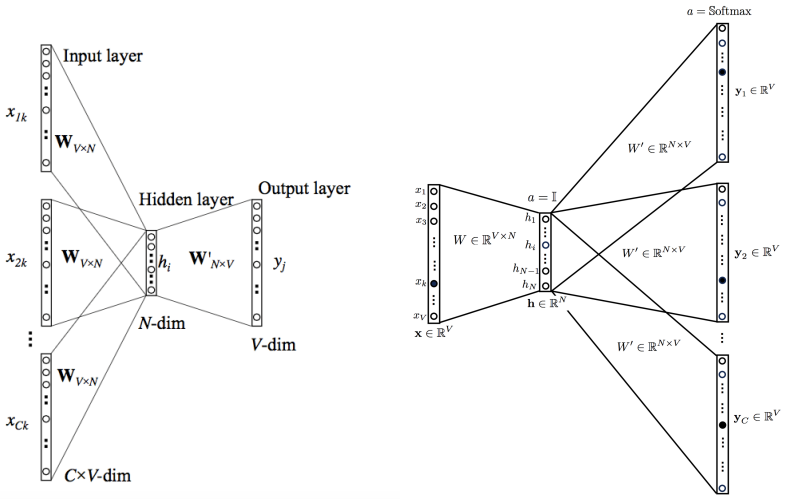
2 Methods

2.1 Word2Vec & Doc2Vec

Word embedding means vector representation of words such that similar words would be closer to each other (Adewumi u.a. ((2020))). It's able to capture context of a word in a text, syntactic and semantic similarity, related to other words, etc. Word2Vec exactly is aiming to capture the context of words while at the same time proposing a hight efficient way of preprocessing raw text. This model takes a large text of documents like tweets or news articles as input and generates a vector space of typically hundreds of dimensions. Rezaeinia u.a. ((2017)) state that each word in the text is assigned a exact vector in the vector space. Word2Vec consists of two different methods: Continuous Bag of Words (CBOW) and Skip-gram. Regarding to the CBOW method, the goal is to predict a word with respect to the surrounding words.

In detail, CBOW takes the context of each word as the input and tries to predict the word corresponding to the context. The hidden layer has the number of dimensions in which the current word needs to be represented at the output layer. Its neurons just copy the weighted sum of inputs to the output layer without activation. Then the output layer applies softmax calculations on the weighted sum of inputs from hidden layer to output the final vector with the elements being the softmax values. Figure 1 shows as example a CBOW model taking C words encoded as V length vectors and outputing a V dimentional vector.

Skip-gram is somehow the way around: we need to predict a window of words given a single word (Figure 2). Both methods apply artificial neural networks as their classification algorithm. At the



(a) CBOW model (b) SG model

Figure 1: Word2Vec models architecture

beginning, each word in the vocabulary is just a random N-dimensional vector. In training, the algorithm learns the optimal vector for each word given the CBOW or Skip-gram method.

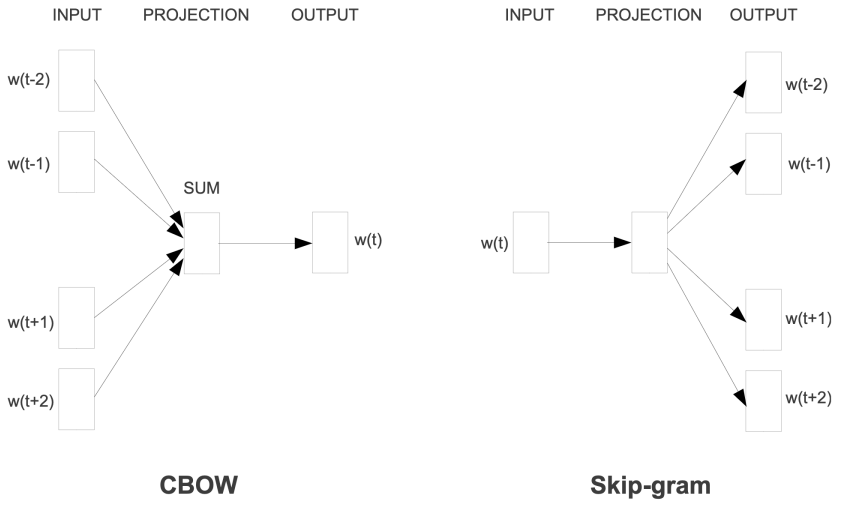
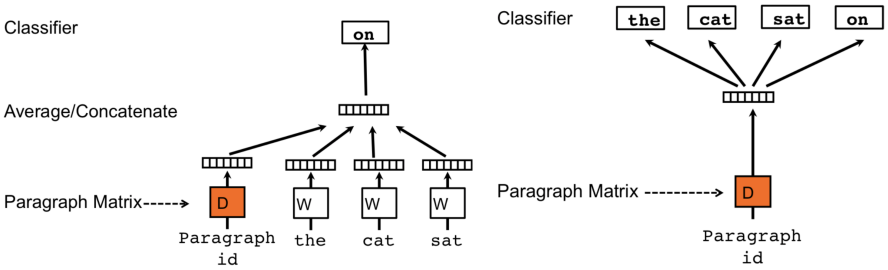


Figure 2: CBOW and Skip-gram method. $w(t)$ is the current word, while $w(t-2)$, $w(t-1)$, etc. are the surrounding words.

These word vectors are capable of capturing the context of surrounding words. This can even be treated by using algebra to find word relations (i.e. "$king$" − "man" + "$woman$" = "$queen$"). Word vectors are fed into the classification algorithm, unlike bag-of-words, to predict sentiment. The contribution is that we now have some word context, and our feature space is much lower (typically 300). Adewumi u.a. ((2020)) show that all we need to do is very little manual feature composition because the neural network is able to extract the features we want for us. Because text have varying length, we might need to take the average of all word vectors as input to a classification method (like machine learning methods) to classify the entire text.

Nevertheless, with the above-mentioned method, averaging word vectors, word order is still ignored to perform the text sentiment analysis. There is a method to summarize bodies of text of varying length: Mikolov *et al.* proposed Doc2Vec method based on, and very like Word2Vec. This method

is identical to Word2Vec, except that we have to generalize the method by adding a text vector. Considering Word2Vec, there are two methods: Distributed Memory (DM) and Distributed Bag of Words (DBOW). DM trys to predict a word with respect to the previous words and a text vector. Even if the context window moves across the text, the text vector still does not allows for some word-order to be captured. Here, DBOW predicts a random group of words in a text with respect to only the text vector (Figure 2).



(a) DM (b) DBOW

Figure 3: Architecture for Doc2Vec

Theoretically, it is better to use Doc2Vec to create our input features to perform a text sentiment analysis, because we risk throwing away rich features when we ignore word order and context if Word2Vec is chosen. In the experiment and result section, we would report the statistics in testing using these two methods respectively.

2.2 Dataset

The dataset we use is derived from the Internet Movie DataBase (IMDb), which is an excellent source for data analytics and research. The table below shows the data format of this dataset used for binary sentiment classification. The labelled dataset consists of 50,000 IMDB movie reviews. It provides a set of 25,000 negative movie reviews and 25,000 positive reviews. In addition, there are other 50,000 IMDB reviews provided without any rating labels to use. Table 1 shows some instances.

Id	Review Text	Sentiment
0	"Foolish hikers go camping in the Utah mountains only to ..."	0
1	"So after years and years I finally track this film down! I ... "	1
2	"Mark Walhberg in a great role, idolises a rock star to the ..."	1
3	"Terrible direction from an awful script. Even the DVD ..."	0
4	"I am a chess player and I wanted to like this. Trouble ... "	0
.	...	...

Table 1: Large Movie Review Dataset

The dataset has been pre-cleaned such that the sentiments of reviews are classified into binary values 0 and 1, by assigning 0 to the records whose IMDB rating is less than 5, and assigning 1 to those with the IMDB ratings larger or equal to 7. For any given movie, there will be at most 30 reviews about it, which makes the dataset more balanced that avoid the influence of correlated ratings in the labelled sets. Thus reviews with more neutral ratings (rating between 5 and 7) are not included in the labeled set. In the unlabeled set, reviews of any rating scores are included and split 50-50 between ratings less or equal to 5 and ratings higher than 5. However, the final format is subjected to change due to possible variation of task. The dataset is publicly accessible on here. It can be downloaded directly without using an API.

2.3 Preprocessing

Review text could be messy sometimes as people would like to use punctuation or misspelling to exaggerate opinions, which could be un-readable for machine learning models. Besides, text cannot

be fed to models directly as input. So, pre-processing is necessary to map the words into numerical format. We use Word2Vec and Doc2Vec here to generate word embeddings for one-hot encoded vectors, such that each unit can be mapped to a vector.

Following are the steps of data pre-processing we perform:

1. Data cleaning
   HTML tags, puctuation, numbers and short words are meaningless to sentiment analysis. We remove them to improve the performance of trained Word2Vec and Doc2Vec model. Meanwhile, Alshari u.a. ((2020)) show that the existence of upper case words expands the size of the vocabulary, and so adds up the computational complexity. Converting them to lower case words helps decrease training cost and improve performance.

2. Removal of stop words
   Stop words are words which are filtered out before processing of natural language data. They usually refer to the most common words in a language, such as "a", "this" and "is" in English. These words normally don't contain sentimental orientation. Therefore, we removed stop words to enhance the classification accuracy and decrease the computational cost.

3. Stemming
   Stemming process reduces the words to their root word. Unlike Lemmatization which uses grammar rules and dictionary for mapping words to root form, stemming simply removes suffixes/prefixes.

4. Sentence splitting
   We split a text by sentences, return a list of sentences, for each is a list of words.

As a result of the pre-processing of data, a review becomes a list composed of lists of cleaned words. After preprocessing, we could feet the data to train the word2vec and Doc2Vec model, then unify the feature set, and finally feed them to classifiers to perform the classification.

3 Experiment

3.1 Parameters

Before discussing the experiments and results, we introduce some vital parameters about Word2Vec and Doc2Vec as specified in Table 2.

Parameters	Meaning	Value chosen
num_features	Word vector dimension	300
min_word_count	Lower than which the model ignores the words with frequency	40
context	The length of word span (window size) a word to capture	10

Table 2: Parameters

The purpose of Word2vec is to numerically group similar words together. Generally, syntactically similar words have dimensionally closer vectors. For example, *"bad"* is pretty similar to *"good"*, because they are both adjectives and either of them always appears in the same position of a sentence, even though they are semantically opposite.

Figure 4 shows that by applying a lower window size, it is easy to observe that syntactically interchange words, such as *"brilliant"* and *"outstand"*, are spatially closer. But when window size is larger, words with similar content will be grouped together as well, such as *"direct"* and *"actor"*, but syntactically similar words are still remained, like *"brilliant"* and *"outstand"*.

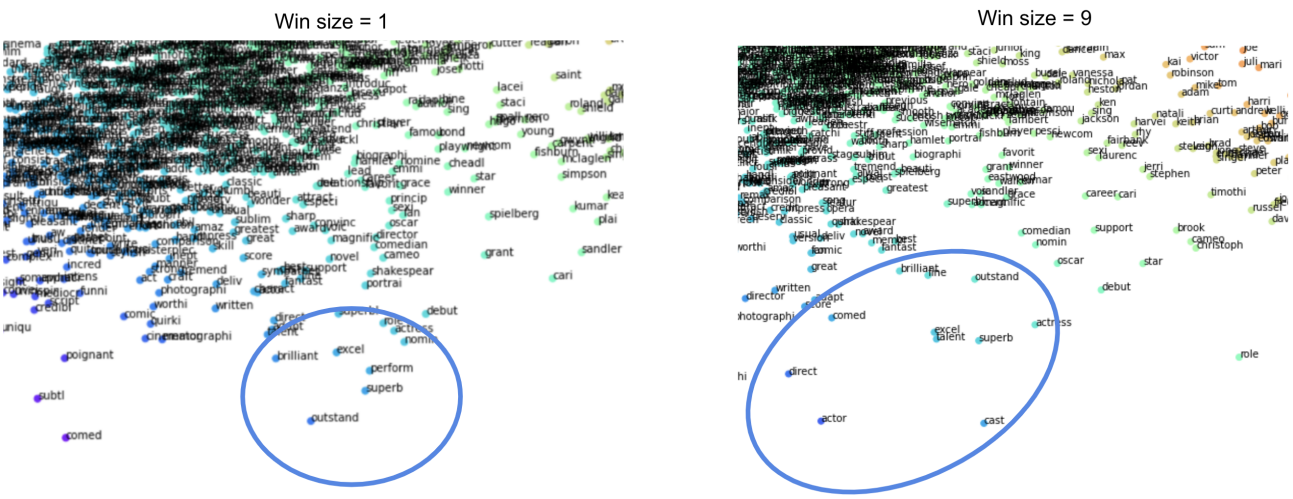


Figure 4: Word embedding mapping to 2D using PCA

Leveraging the length of word span a word to capture, we force the model to better group together semantically similar words. Figure 5 shows that a smaller window size gives syntactically interchangeable words higher similarity scores (cosine similarity), thus resulting in lower test accuracy. A larger window size enables the word vector to capture more context meaning locally, thus resulting in higher test accuracy.

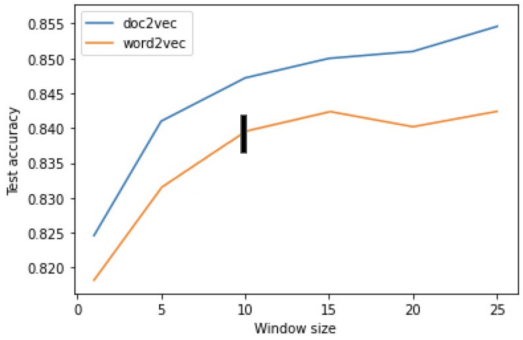


Figure 5: Parameter "**content**" tuning vs. test accuracy by random forest classifier

The calculated average length of our cleaned review sentence in dataset is 10. As shown in Figure 5, the performance starts to converge at around window size = 10. This shows that average length of sentence might be a good indicator for window size setting.

3.2 Normalization

Since different review texts have different length in words, we convert the individual word vectors of each review into a uniform feature set for later classification task. An intuitive method of averaging vectors is used in our experiment. Figure 6 shows a text in vector representation D normalized from averaging a set of words in vector representation $\{W_1, W_2, ..., W_n\}$. After the normalization, the prepared feature set can be fed into classifiers for training and predicting.

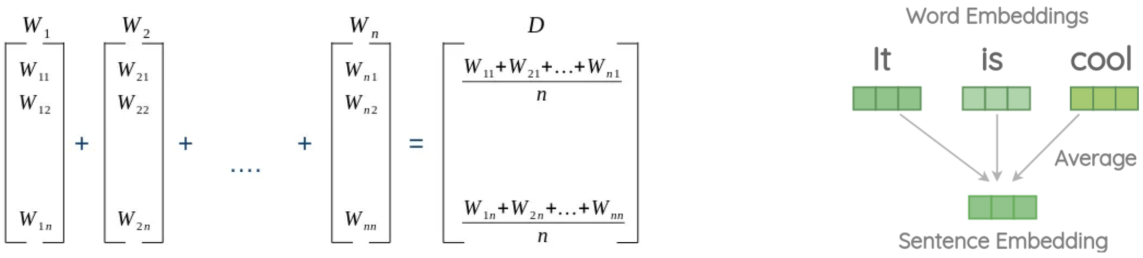


Figure 6: Normalization of text in vector representation

3.3 Training

Both underfitting and overfitting are not very distinct upon cross validation using 80% dataset, so we commit to this ratio to perform the model training and testing. We use multiple machine learning models, including Gaussian Bayes, Logistics, random forest, linear and Gaussian SVMs, as classifier to perform the text sentiment analysis respectively.

4 Results

4.1 Evaluating metrics

As mentioned in the training section, the dataset is divided into a training set and testing set, where 80% data is used for training and validation, and 20% data is used for testing. Since the result of the test set is objective and accurate, we use the proportion of the number of correctly-predicted data in the test set to represent the performance of a model. The higher the correct rate is, the more accurate our model is. For example, if there are n reviews in the test set and in our result m reviews have the same labels as those of test set, then the correct rate could be easily computed by $C = m/n$. An evaluation table based on the correct rate C is given in Table 3.

Correct Rate(C)	Performance(P)
$C \geq 0.8$	Good
$0.6 \leq C < 0.8$	Fine
$C < 0.8$	Bad

Table 3: Performance Metrics

4.2 Final performance

Each model is upon fine-tuning to optimality (with respect to the regularization parameter and so on). Other parameters are tuned as specified in Table 2. Our final performance is summarized in Table 4.

Method	Correct Rate(C)	Performance(P)
Random Forest (given Word2Vec features)	84.42%	Good
Random Forest (given Doc2Vec features)	84.46%	Good
Gaussian Bayes (given Word2Vec features)	75.98%	Fine
Gaussian Bayes (given Doc2Vec features)	82.22%	Good
Logistics (given Word2Vec features)	87.90%	Good
Logistics (given Doc2Vec features)	**88.04%**	Good
Linear SVM (given Word2Vec features)	87.28%	Good
Linear SVM (given Doc2Vec features)	88%	Good
Gaussian SVM (given Word2Vec features)	87.68%	Good
Gaussian SVM (given Doc2Vec features)	87.9%	Good

Table 4: Final performance

Table 4 shows that all classifiers we implemented demonstrate the capabilities of carrying a sentiment analysis tasks. It is obvious that Doc2Vec model works better than Word2Vec in any case. This is reasonable because Doc2Vec is an improved model of Word2Vec, which considers rich features including word order and context. The best classifier in our experiment is logistic regression, which has a score as high as 88.04% on Doc2Vec model.

5 Conclusion

In this project, we worked on a text sentiment analysis task on a large IMDB review dataset, using multiple machine learning models based on Word2Vec and Doc2Vec word embedding features. In training and testing, text data in normalized vector representation are fed into a variety of classifiers to perform the sentiment analysis task. Larger word span a word could capture provides better semantic similarity among words representation in higher-dimension. Our methods achieved good performance with the highest accuracy of 88%, which shows that text sentiment analysis is effective using both Word2Vec and Doc2Vec, and obviously, Doc2Vec performs better than Word2Vec on text sentiment analysis. Normalization method of averaging words vectors to text vector causes word embedding to lose some important semantic similarity and spatial closeness. We would try to propose a novel method to optimize the vector representation of text in the future.

References

Rezaeinia, Seyed Mahdi / Ghodsi, Ali / Rahmani, Rouhollah(2017): *Improving the accuracy of pre-trained word embeddings for sentiment analysis*.

Giatsoglou, Maria / Vozalis, Manolis G / Diamantaras, Konstantinos / Vakali, Athena / Sarigiannidis, George / Chatzisavvas, Konstantinos Ch(2017): *Sentiment analysis leveraging emotions and word embeddings*214–224.

Liu, Haixia(2017): *Sentiment analysis of citations using word2vec*.

Acosta, Joshua / Lamaute, Norissa / Luo, Mingxiao / Finkelstein, Ezra / Andreea, C(2017): *Sentiment analysis of twitter messages using word2vec*1–7.

Chen, Qufei / Sokolova, Marina(2018): *Word2Vec and Doc2Vec in unsupervised sentiment analysis of clinical discharge summaries*.

Nawangsari, Rizka Putri / Kusumaningrum, Retno / Wibowo, Adi(2019): *Word2vec for Indonesian sentiment analysis towards hotel reviews: An evaluation study*360–366.

Alshari, Eissa M / Azman, Azreen / Doraisamy, Shyamala / Mustapha, Norwati / Alksher, Mostafa(2020): *SENTI2VEC: AN EFFECTIVE FEATURE EXTRACTION TECHNIQUE FOR SENTIMENT ANALYSIS BASED ON WORD2VEC*, 3: 240–251.

Adewumi, Tosin P / Liwicki, Foteini / Liwicki, Marcus(2020): *Word2vec: Optimal hyper-parameters and their impact on nlp downstream tasks*.