# Sentiment Analysis of Movie Reviews

**Aman Singhal**
amansinghalml@gmail.com

## Abstract

Sentiment classification is used to categorize user opinion in a document - such as movie reviews as positive or negative. The opinion expressed towards movie reviews is also accompanied by a rating expressed by the same user which serves as a heuristic for the degree of user sentiment. The contextual meaning of a movie review is understood through dense word embeddings. The derived meaning is then understood by a LSTM classifier to mark each movie review with a hypothesized sentiment label. In this paper we discuss the choice of corpus and hyperparameters for training word embeddings for sentiment classification tasks along with an evaluation of design choices that boost the performance of the LSTM classifier.

## 1 Introduction

Sentiment analysis is a language processing task that uses a computational approach to identify opinionated content and categorize it as positive or negative. The unstructured textual data on the Web often carries the expression of opinions of users. Sentiment analysis tries to identify the expressions of opinion and mood of writers. A simple sentiment analysis algorithm attempts to classify a document as 'positive' or 'negative', based on the opinion expressed in it. The document-level sentiment analysis problem is defined as follows: Given a set of documents D, a sentiment analysis algorithm classifies each document d belonging to D into one of the two classes, positive and negative. Positive label denotes that the document d expresses a positive opinion and negative label means that d expresses a negative opinion of the user. The new user-centric Web hosts a large volume of data created by various users. Users are now co-creators of web content, rather than being passive consumers. The social media is now a major part of the Web. The statistics show that every four out of five users on the Internet use some form of social media. The user contributions to social media range from blog posts, tweets, reviews and, photo/ video uploads, etc. A large amount of the data on the Web is unstructured text. Opinions expressed in social media in form of reviews or posts constitute an important and interesting area worth exploration and exploitation. With an increase in accessibility of opinion resources such as movie reviews, product reviews, blog reviews, social network tweets, the new challenging task is to mine large volumes of texts and devise suitable algorithms to understand the opinion of others. This information is of immense potential to companies that try to know the feedback about their products or services. This feedback helps them in making informed decisions. In addition, to be useful for companies, the reviews and opinion mined from them is helpful for users as well. For example, reviews about hotels in a city may help a user visiting that city locating a good hotel. Similarly, movie reviews help other users in deciding whether the movie is worth the watch or not. In this paper, we want to investigate the effectiveness of long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) for sentiment classification of short texts with distributed representation in social media. First, a word embedding model based on Word2Vec (Mikolov et al., 2013) is used to represent words in short texts as vectors. Second, LSTM is used for learning long-distance dependency between word sequences in short texts. The final output from the last point of time is used as the prediction result. (Wang et al., 2018) The remainder of this paper is organized as follows: Section 2 lists the related works, and section 3 describes the proposed method. The experimental results are described in section 4. Finally, section 5 lists the conclusion and future work.

## 2 Related Work

This work is based on the model architecture introduced in (Wang et al., 2018). The paper follows the idea of dense word embeddings to initialize a deep learning-based classification model as in (Cano and Morisio, 2019) and (Yin et al., 2017). Non-neural classifiers for sentiment analysis leverage SVM and Naïve Bayes classifiers in (Devi et al., 2020) and (Singh et al., 2017). While the performance of such techniques is comparable to neural network-based techniques in the absence of large datasets, neural network-based techniques, especially LSTM based classification models are superior when sufficient data to train them is available (Wang et al., 2018). Word embedding is a text mining technique of establishing a relationship between words in textual data (Corpus). The concept of distributional hypothesis suggests that words occurring in a similar context are semantically similar (Sahlgren, 2008). Count-based embeddings or sparse word embeddings and prediction-based embeddings or dense word embeddings are the two broad approaches to generate word embedding (Pang et al., 2002). Just like the traditional bag-of-words model, the count-based embeddings performs poorly at preserving contextual information in textual data according to (Baroni et al., 2014). Dense word embeddings outperform the sparse word embeddings for representing the sentiment of a document as they are continuous in the vector representation space and thus have a lower dimensionality and higher contextual information as compared to sparse word embeddings (Baroni et al., 2014). The efficiency of word embeddings for representing the sentiment of a document is dependent on the underlying corpus used for training word embeddings (Wang et al., 2019). Word embeddings trained on in-domain corpus outperform word embeddings trained on out-of-domain dataset or word embeddings trained on general domain datasets. Instead of evaluating the word embeddings on the ultimate sentiment analysis task, which is both time-consuming and computationally expensive, (Cano and Morisio, 2019) suggests evaluation on downstream semantic word analogy tasks for methodological efficacy. In the subsequent sections, we introduce the dataset used for experimentation in section 3, followed by description of the individual components of the sentiment classification pipeline in section 3.1 through section 3.4. Finally, we discuss the experiments performed in section 4 and conclude the findings of this paper in section 5.

## 3 Classifying Movie Sentiment

The IMDb dataset (Maas et al., 2011) consists of 50,000 reviews of movies such that the maximum number of reviews for a movie do not exceed 30 to avoid overfitting on the reviews of popular movies such as "The Godfather". Sentiment labels are dependent on the user rating (1 to 10), while Positive (user rating between 7 to 10) and Negative (user rating 1 to 4) reviews are preserved, Neutral (user rating between 5 to 6) reviews are discarded to demarcate a clear distinction between classification labels. Both these sentiment labels are present in an equal ratio of 25000:25000 in this dataset.

### 3.1 Data pre-processing

Text collected from user reviews is liable to the presence of unnecessary punctuations, misspellings, semantic and syntactic inconsistencies across multiple writing styles. Text pre-processing in the form of stop-word and punctuation removal, lowercasing, stemming and lemmatization is used to standardize text before model design. Extending (Camacho-Collados and Pilehvar, 2017), the effectiveness of different combinations of text pre-processing techniques on the sentiment analysis has been evaluated in this paper.

### 3.2 Word Embeddings

Following (Wang et al., 2018), this paper limits itself to word embeddings generated using word2Vec. Popular pre-trained word embeddings trained by Google on News dataset (100 billion tokens) and self-trained word embeddings trained with varying hyper-parameters such as window size, model architecture, minimum word frequency, and embedding dimension are evaluated in the context of their performance on sentiment classification. Given a set of (word, context) pairs extracted from the corpus, word2Vec (Mikolov et al., 2013) representations of words can be derived through various estimation methods, such as predicting words given their contexts (CBOW) or predicting the contexts from the words (Skip-Gram). This refers to the hyper-parameter of the model architecture. In all of these approaches, the choice of context is a crucial factor that directly affects the resulting vector representations. The most common method for defining this context is to rely on a window centered around the word to estimate

(often called the focus word). The context window thus determines which contextual neighbors are taken into account when estimating the vector representations. The most prominent hyper-parameter associated to the context window is the maximum window size (maximum distance between the focus word and its contextual neighbors). Finally, embedding size refers to the 1-D matrix which carriers the contextual representation of a word.

### 3.3 LSTM Network

The Long Short Term Memory (Hochreiter and Schmidhuber, 1997) Network used in this paper is initialized with pre-trained word embeddings with the dimensions given by vocabulary size used considered for prediction and size of the word embeddings as defined in Section 4.2. The vocabulary size is the number of unique words in the corpus having frequency greater than the minimum frequency defined in Section 4.2. The remainder of the words are replaced by an "¡OOV¿" or Out of Vocabulary token and assigned a 1-D zero vector word representation. To avoid a very long training time, we adopt a single hidden layer neural network. The number of neurons in this hidden layer is the same as the dimension of word2Vec model, and the number of neurons in output layer is the number of classes, which is 2 in this case. The output of the hidden layer is fed to a dropout layer and finally a dense layer outputs the predicted sentiment label. By gradient-based backpropagation through time, we can adjust the weight of edges in the hidden layer at each point of time. After several epochs of training, we can obtain the sentiment classification model. The network hyperparameters such as dropout probability is fixed as 0.3 and batch size as 50. The learning loss is binary cross-entropy and the Adam optimizer function (Kingma and Ba, 2015) has been used to maximize model performance

### 3.4 K-fold Cross-Validation

Cross-Validation (RefaeilzadehLei and Liu, 2009) is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to validate the model. In typical cross-validation, the training and validation sets must cross-over in successive rounds such that each data point has a chance of being validated against. The basic form of cross-validation is k-fold cross-validation. In k-fold cross-validation, the data is first partitioned into k equally (or nearly equally) sized segments or folds. Subsequently k iterations of training and validation are performed such that within each iteration a different fold of the data is held-out for validation while the remaining k  1 folds are used for learning. K was set to 10 for the scope of all LSTM based experiments discussed in the paper.

| Corpus Domain | Word Embeddings | F1 Score |
|---|---|---|
| none | none | 23.32 |
| general | GoogleNews | 56.42 |
| movie reviews | self-trained | **62.59** |

Table 1: In-domain v/s out-of-domain corpus training of word2vec

## 4 Experiment Design

The experiments listed below are performed on a reduced dataset instead of the full dataset (30 percent of the dataset post train-test split) to ensure a shorter execution time as well as lower resource utilization. Nevertheless, the reduced dataset size is sufficient to justify extrapolation of results obtained on the reduced dataset to the full dataset. F1 score is used as the evaluation metric. In the experiment design section, in 4.1 we compare the performance of popular pre-trained word embedding models such as GoogleNews and self-trained word embedding models, in 4.2 we explore parameter tuning for the self-trained Word2Vec model by evaluating an intrinsic downstream task of semantic word analogy, in 4.3 we explore the significance of incorporating stop-words and k-fold cross-validation while training the classification model.

### 4.1 Comparing pre-trained and self-trained word embeddings

The experiment aims at studying the impact of the domain of training corpus on the F1 score obtained in downstream sentiment analysis task. For this, we compare word embeddings trained on the movie reviews corpus itself (self-trained) and word embeddings trained on a generic corpus. Self-trained word embeddings are initialized with the same parameters as the Google News Word2Vec. Hence, the embedding size is 300, misspelled words are not removed and stop-words are selectively removed. The F1 score obtained for sentiment classification on the reduced IMDb dataset is given in Table 4.1.

The difference in F1 scores despite similar embedding parameters can be attributed to the relevance of domain for self-trained word embeddings. This verifies the experiments performed in (Cano and Morisio, 2019).

### 4.2 Evaluating word analogy task for Word2Vec parameter tuning

(Cano and Morisio, 2019) suggests a strong correlation between performance of word2Vec for sentiment classification and on downstream semantic word analogy task. We use the semantic queries present in the Bigger Analogy Test Set (BATS) (Gladkova et al.) to evaluate various self-trained Word2Vec models - trained with varying choice of window size, minimum count and model type (Skip-Gram and CBOW) parameters. We varied the window size between 5 to 25, minimum count between 0 to 10 and embedding size between 50 to 400 in increments of 50. Table 4.2 shows the top results based on number of correctly answered queries. Coverage refers to the number of analogy questions that were posed to the Word2Vec model. It is dependent on the vocabulary overlap between the training corpus and the BATS dataset queries and also varies with the variation in minimum count. Percentage of correct answers as the ratio coverage varies between 7.5 percent and 8 percent. The word embeddings with window size as five, minimum count as two, embedding size as three hundred and skip-gram model type are used for subsequent experiments.

### 4.3 Modeling choices for the classification model

While stop-words are included selectively for training the Word2Vec model for generating embeddings with better contextual information, we experiment with their inclusion in the vocabulary of the classification model. Additionally, we also evaluate the impact of using 5-fold cross-validation during model training. The results obtained are presented in Table 3.

## 5 Conclusion and Future Work

We report the F1 scores on a reduced IMDb dataset in Table 1 to compare the performance of pre-trained word embeddings and self-trained word embeddings with the same configuration trained on in-domain data. The better performance of self-trained word embeddings leads us to further explore the space of self-trained word embeddings and the impact embedding training parameters have on their performance on semantic word analogy tasks, the results of which are discussed in Table 2. The best five configurations all have a window size of five and a skip-gram model type. Based on the strong correlations between embedding performance on word analogy tasks and sentiment classification tasks, it can be concluded that the word embeddings which answer the maximum correct answers to the analogy task are indeed the best word embeddings for our use case. We also experiment with validation and stop-words as presented in Table 3 and conclude the best performance on the model which does not include stop-words in model vocabulary and uses 5-fold cross-validation. Other research with similar model architecture (Camacho-Collados and Pilehvar, 2017) and (Wang et al., 2018) reports a F1 score of 88.9 percent and 85.9 percent respectively on the entire dataset, while with our configuration a F1 score of 92.44 percent is achieved on the entire dataset. This jump is F1 score points to the significant importance of configuration choices on model performance. The current approach is limited to only the LSTM architecture and Word2Vec word embeddings. We can extend the approach to other popular word embedding models such as Byte-Pair Encoding and GloVe to identify the importance of model configuration with varying embedding models.

| Window | Min. count | Embeddings | Model | Coverage | No. Correct | %Correct |
|---|---|---|---|---|---|---|
| 5 | 2 | 300 | skip-gram | 29419 | **2273** | 7.73 |
| 5 | 2 | 200 | skip-gram | 29419 | 2250 | 7.65 |
| 5 | 2 | 400 | skip-gram | 29419 | 2235 | 7.60 |
| 5 | 3 | 300 | skip-gram | 28337 | 2226 | 7.86 |
| 5 | 2 | 300 | skip-gram | 29419 | 2211 | 7.52 |

Table 2: Word analogy evaluation results

| Validation type | Stop words | F1 Score |
|---|---|---|
| none | included | 38.70 |
| none | removed | 58.44 |
| 5-fold | included | 43.82 |
| 5-fold | removed | **62.59** |

Table 3: Experiments with k-fold validation and stop-words in the classification model

## References


Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Jose Camacho-Collados and Mohammad Taher Pilehvar. 2017. On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis.

Erion Cano and Maurizio Morisio. 2019. Word embeddings for sentiment analysis: A comprehensive empirical survey.

B. Lakshmi Devi, V. Varaswathi Bai, Somula Ramasubbareddy, and K. Govinda. 2020. Sentiment analysis on movie reviews. In *Advances in Intelligent Systems and Computing*, pages 3743–3751, Singapore. Springer.

Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. Analogy-based detection of morphological and semantic relations with word embeddings: What works and what doesn't. In *Proceedings of the NAACL-HLT SRW, address = San Diego, California, June 12-17, 2016, publisher = ACL, year = 2016, pages = 47-54 doi = 10.18653/v1/N16-2002, url = https://www.aclweb.org/anthology/N/N16/N16-2002.pdf*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8).

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *A3rd International Conference for Learning Representations*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality.

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

Payam RefaeilzadehLei and TangHuan Liu. 2009. Sentiment analysis on movie reviews. In *In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems*, pages 3743–3751, Boston, MA. Springer.

Magnus Sahlgren. 2008. The distributional hypothesis. In *Rivista di Linguistica*.

Jaspreet Singh, Gurvinder Singh, and Rajinder Singh. 2017. Optimization of sentiment analysis using machine learning classifiers. *Human-centric Computing and Information Sciences*, 7(32).

Bin Wang, Angela Wang, Fenxiao Chen, Yuncheng Wang, and C.C. Jay Kuo. 2019. Evaluating word embedding models: Methods and experimental results. In *APSIPA Transactions on Signal and Information Processing 8 (2019) e19*.

Jenq-Haur Wang, Ting-Wei Liu, Shashi Pal Singh, Xiong Luo, and Long Wang. 2018. An lstm approach to short text sentiment classification with word embeddings. In *The Association for Computational Linguistics and Chinese Language Processing*.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schutze. 2017. Comparative study of cnn and rnn for natural language processing.