COMP 551 Project 2 - Group 68
IMDB Movie Review Sentiment Analysis

Yan Miao 260711311 Sitong Chen 260654457 Linge Xie 260660974

Abstract

In this project, we investigated the performance of several discriminative and generative models, as well as their ensembles, on predicting the sentiment of classic IMDB movie review dataset [4]. The best performance we discovered came from the ensemble method, especially when discriminative and generative models were combined. The other shocking findings is that rather than doing careful processing and feature selection, using all possible tri-gram features with barely any pre-processing boosted the model performance drastically.

Introduction

The goal of this project is to conduct binary sentiment classification of IMDB move reviews dataset with 50k full length review, of which each was labeled with binary values, where one indicates that a review is positive, vice versa. To tackle this problem with machine learning approaches, we imposed a bag-of-words representation over this dataset and investigated different modifications over this model, including TF-IDF, n-gram [8], and word embedding [5, 6]. In terms of the classifier choice, we examined three individual models (Naive Bayes, Logistic Regression and SVM) to find that their performance were all limited, regardless of whether they were generative or discriminative. In order to break this limit, we utilized the ensemble methods (NBSVM NBLR) that combined generative language model with discriminative classifier [3, 7]. Furthermore, we attempted to improve our ensemble model by combining it with other individual models into a voting classifier. However, all those attempts still could not beat the benchmark, so we made a final desperate attempt with binary count of all possible trigram word features and minimal processing (only remove non-letter character) in a NBSVM model to find around 0.92 5-fold cross-validated accuracy. Our final testing accuracy over 30% test data is 0.925. Our best model demonstrated that minimal processing to generate maximum number of features led to best classifier performance.

Related work

Binary classification of IMDB movie review is often considered as one of the simplest in NLP because basic machine learning techniques can yield strong baselines [3], which are able to beat much complicated models. The most common approach to represent those variable length document is to apply bag-of-words, which loses word order information. However, some word structure features can be included by incorporating n-gram features [8]. There are other representations which utilize recurrent neural networks or its variations [2], but it is not obvious whether these provide any significant gain over the simple bag-of-words and bag-of-ngrams representations [3, 7]. Word embedding taken from hidden layers of neural network has also been proposed to help capture semantic and syntactic rules of words in reviews [1, 5, 6]. Incorporation of this new representation as the paper claimed further improved accuracy over state-of-art models, but many suggested the results were difficult to reproduce. As for classifiers, ensemble models have been shown to perform better than any individual techniques [7], especially when the estimators are complementary, for instance, generative and discriminative. Most of the papers focus on designing new representation and models, but mention very little about processing, which makes it difficult to reproduce their reported accuracy sometimes.

Dataset

The dataset we are analysing is the IMDB movie review dataset with 50k full-length reviews [4]. The dataset were splitted into 25k training and 25k testing samples before passing to us. We used 5-fold cross-validation during training over the 25k training samples, and the same process was applied during hyper-parameter tuning. In general, we utilized the most common way of representing text data for machine learning – bag-of-words, in which the feature matrix is constructed by counting how often each word appears in each text in the corpus. Counting this word occurrence where text structures are ignored leads to picture of representing text as a bag [8].

In particular, to compute this bag-of-words representation, the following steps are applied on the IMDB dataset:

- Cleaning1: remove any non-letter characters and html breaks and decode those data into utf-8 format.
- Tokenization: lowercase and split each document by space into the words that appears in them (now each word is called a token).

- Cleaning2: lemmatize each token so that word derivatives will not be treated as different tokens.
- Vocabulary Building: collect the vocabulary of all tokens that appear more than 5 times and order them alphabetically
- Encoding: construct the feature matrix by counting, for each document, how often each of the words in the vocabulary appears

Note we did not remove any stopwords as we found that accuracy decreases in the presence of this process.

Proposed Methods

We surveyed a list of methods that introduced various modifications to the bag-of-words model [5, 6, 7, 8], leading to promising increase in accuracy. We will start by formulating our baseline model before establishing how each of the modifications improves our baseline:

Raw Count (or Binary Count) Representation

To compute the raw count, we followed exactly the same steps listed in Dataset section, from which the binary count representation can also be derived to be used in testing the Bernoulli Naive Bayes model. However, this simple method lacks normalization and present words that are not expected to have good indication of the review sentiment, for example, the token "movie".

TF-IDF

Tf-Idf stands for term frequency, inverse document frequency. Rather than the raw count (term frequency) of each token, this technique also normalize those counts by the overall frequency of the corresponding token. [8] The formula is given by:

$$tfidf_{i,j} = tf_{i,j} \times \log(N/df_i)^1$$

In this way, each token can be ranked by its importance as in that it is frequent in a document, but not among all the documents. For example, the token "movie" will receive a much lower TF-IDF score, indicating a lower importance compared to its raw count.

[1] $tf_{i,j}$ = total number of occurrence of token i in document j; df_i = number of documents that contain the token i; N = total number of documents.

N-gram

As mentioned previously, the bags of word model discards the document structure and great chances are those structural information can be crucial in interpretation the review sentiment. For instance, in a simple unigram model, a bigram token "not good" is considered separately as "not" and "good", potentially leading to the opposite interpretation in our model. Thus we strongly felt the need to bring back some context information in the data by adding bigram and trigram token features [8]. We did not go any further because those additional considerations add to the vocabulary size exponentially, which can lead to memory error and overfitting during training.

Word Embedding

Word embeddings in the hidden layer of the neural network have been shown to capture some the semantic and syntactic rules of words in the documents. Words are semantically related if they frequently appear in the same context, while they are said to be syntactically related if they have the same syntax [1]. This indicates that word embeddings in the hidden layer of a neural network can be used to cluster similar words together which can be used to train a text classifier. We applied the word2vec function from Gensim to train a distributed representation of the words in our tokenized training documents [5], using the following hyperparameters: (dbow_words=0, size=100, window=10, negative=5, sample=1e-4, iter=20, min_count=1). Those hyperparameters available in Gensim were selected to be as close to what information Mikolov provided in his paper[6]. In this representation, words with similar meaning are clustered together, while those clusters are also spaced out with 100 vectors so that word relationship can be captured using vector math, i.e. "not"+"bad" = "fine". We did not go beyond 100 vector because of our memory limit and training time, despite the possibility of increasing accuracy.

With the same processing as mentioned in Dataset, we used the average word vector to represent the corresponding review in the feature space, which ensures that feature extracted from each review is of the same length, even though the reviews are of various lengths. This averaging method is suggested and used in many Kaggle kernels.

Classifiers

We surveyed three most commonly used text classifiers as suggested by the Scikit-Learn Library, including a generative Naive Bayes (Multinomial) and discriminative Logistic Regression and Support Vector Machines (Linear SVC). In particular, to compare how different modifications change our model performance, we only used the Logistic Regression classifier. Exhaustive grid searches with 5-fold cross validation were performed to ensure best hyper-parameters were used in each of the modifications. For all the three classifiers, L2 regularized penalty is applied since we would like to be computationally efficient and have stable and non-sparse output (no further feature selection is needed). Note that we excluded decision tree and random forest as they consumed a large amount of time in training and cross-validation.

Ensemble Methods

Model combination has been shown to perform better than any individual technique. The ensemble best benefits from integrating models that are complementary [7]. Indeed, this is aligned with our observations that those individual techniques have limit in their performance after exhausting all optimizations. Most of individual models proposed on Kaggle and in literature are discriminative in nature since they perform better generally than generative models. However, we found an ensemble method that combines both types of models to yield better performance than either of two [7]. In particular, for the generative component, we followed the procedure in [3, 7] to train two Naive Bayes models, one on the positive reviews and the other on the negative ones. The likelihood ratio of these two models evaluated on the data is used as an additional feature to feed into a discriminative model, where the underlying assumption is that a positive review will have a higher likelihood generated by a model trained on only the positive reviews. We chose Logistic Regression (NBLR) and Linear SVC (NBSVM) to be our discriminative model based on their strong performance as individual models.

Voting Classifier

In addition, we extended our exploration in combining multiple models in the hope of increasing our accuracy further, with the help of the Scikit-Learn function Voting Classifier, we integrated NBLR with two more individual models, Multinomial Naive Bayes and Logistic Regression. The soft voting classifier predicts based on the weighted average of probability output from the three participating classifier, where the weights can be customized. We performed explicit grid search with 5-fold cross validation to determine the best weight assignment as well as hyper-parameters in order to optimize our model performance. Notice that we did not simply apply hyper-parameters tuned previously in individual models. In additional, we did not consider integrating NBSVM or Linear SVC as as neither of these estimator could output probability which was required by our soft voting classifier.

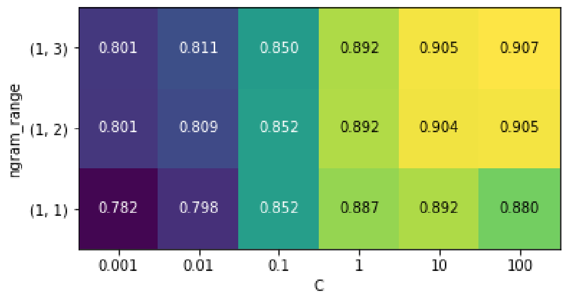
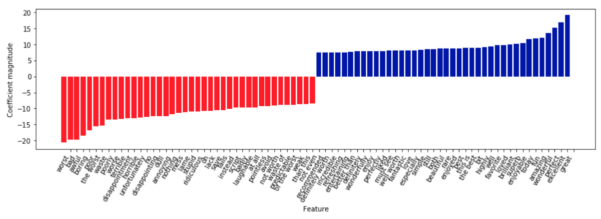
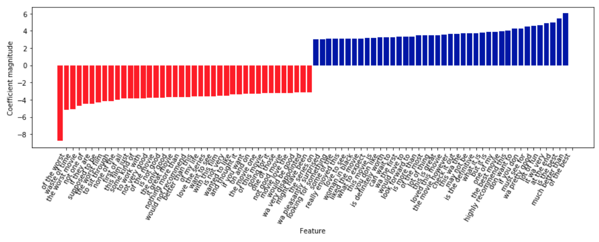


Figure 1: 5-fold cross validation result with respect to n-gram range and C in Logistic Regression. We tuned n-gram range along with C because we tested those model modifications on Logistic Regression only. Note that C was set between (0.001, 100) to prevent under- or overfitting.



((a)) Top 40 Tokens;Y-limit is (-20,20)



((b)) Top 40 Tri-gram Tokens;Y-limit is (-8,6)

Figure 2: Coefficient magnitude of top 40 tokens or top 40 tri-gram tokens taken from a Logistic Regression classifier trained on TFIDF bag-of-trigrams.

Results

TF-IDF Vs. Raw Count

As expected, the TF-IDF normalized version of bag-of-words model was able to achieve higher accuracy than the unnormalized raw count model. The mean of 5-fold cross-validated accuracy saw an 0.009 increase from 0.883 to 0.892 in the Logistic Regression [Table.1]. Therefore, for all subsequent models, we consistently applied the TF-IDF score to construct feature matrix, rather than the raw count.

Bag-of-ngrams Boosts Model Performance

Before implementing any n-gram features, our best model accuracy was around 0.892

[Fig.1][Table.1] and we were struggling to obtain anything over 0.9. Nevertheless, we were able to surpass this threshold by integrating bi- and tri-gram features. In particular, the largest improvement came from the addition of bi-gram features, where the mean of 5-fold cross-validated accuracy increased to 0.905 [Fig.1]. However, the incorporation of tri-gram features did not lead to significantly larger improvement than the bi-gram. To investigate this observation, we explored the top 40 features and the top 40 tri-gram features in our logistic model and found that the top 40 features contain no tri-grams tokens, but a number of bi-gram ones [Fig.2]. Furthermore, the top tri-gram words are also associated with much lower coefficients in the model, which indicated their relatively lower importance in determining sentiment [Fig.2].

This observation above indicates that potentially we can exclude tri-gram features but still achieve similar accuracy at much lower cost of memory. However, we kept the tri-gram features anyway since we did not run into issues with memory while more features seemed to always increase our model accuracy slightly throughout our experiments.

Word Embedding not Improving Baseline Model

Unfortunately, in our implementation of word embedding trained with Logistic Regression, we observed a drop in the cross-validated accuracy to 0.82, not anywhere close to the improvement shown in the paper [6]. We came to agreement that a huge flaw in our implementation was that the averaging processes ignored the fact that words are not all of the same importance. For instance, in the case of movie reviews, an adjective will be of more importance than a proposition unigram-wise, yet the averaging procedure uniformized them. In other words, those insignificant words are not filtered out but act like noise in the sentiment classification. Thus, we integrated the TF-IDF scoring scheme to construct a TF-IDF weighted averaging of the words, but the accuracy only increase for about 0.02, which is still a lot lower than the baseline Logistic Regression [Table.1]. Most online resources conclude the difficulty to reproduce the improvement presented in the paper [6], so at that point, we did not explore any further down this direction as we had done more promising work in the ensemble method.

Individual Model Performance is Limited

Table.1 reports how the three different models performed individually under a TF-IDF bag-of-trigrams feature matrix; in general, none of those individual models was able to achieve accuracy higher than 0.91 [Table.1], given that we had performed exhaustive grid search to provide the best possible parameters. Out of the models tested, Logistic Regression demonstrated the best performance based on the cross-validated accuracy at 0.907, so it became our primary choice when it came to combining models later in a voting classifier. Note that the Bernoulli Naive Bayes we implemented on unigram binary features achieved 0.843 accuracy, it is not reported in Table.1 as we did not use it to do any comparison or further test.

Ensemble Method Leading to Most Improvement

Following the ensemble method proposed by [3], our implementation of NBLR was able to reach a cross-validated accuracy of 0.912, which outperformed all individual models. Note that this ensemble method is different from a simple combination of two models, but a discriminative model with feature input from generative language models. To demonstrate this difference, we also trained a voting classifier with Multinomial Naive Bayes and Logistic Regression, which as expected were not able to match performance of our NBLR. However, since these two models are complementary, the voting classifier out-performs the individual models, which is consistent with the fact that model combination leads to better performance than any individual techniques [7]. Similar improvement was observed for the NBSVM models too, reaching a cross-validated accuracy of 0.912.

In the hope of further improving our model performance, we attempted to construct voting classifiers that integrated NBLR with other individual models that demonstrated promising performance alone. The two candidates were narrowed down to Logistic Regression and Multinomial Naive Bayes. As Table.1 reported, the final voting classifier model that incorporated NBLR and the two candidate models reached a cross-validated score of 0.916. To see if this model could generalize, we submitted the prediction on Kaggle and obtained a testing accuracy of 0.916 with 30% of the test data.

In addition, as mentioned previously, the voting classifier computes a weighted average of the probabilities, where the weights can be customized. We performed grid search with different combinations of weights and assigned the best set based on cross-validated accuracy. Those weight assignment are displayed in square brackets following their corresponding models in a voting classifier [Table.1]. Retrospectively, those weights indicate the relative importance of each of the participating classifiers. Our best performing NBLR is consistently the most important estimator within the voting classifier.

Final Desperate Attempt!!

Up to this point, we have exhausted our optimization on the model but could not achieve the accuracy claimed in the papers, thus we suspected that our data processing and feature matrix construction was not ideal. Those paper did not explicitly state their way of processing so we referred to several versions of implementation of NBSVM [3] on Github for help. To our surprise, none of those implementation did any feature normalization or selection, but used bag-of-trigrams in simple binary count format with no minimum occurrence requirement. Therefore, as our final desperate attempt, we constructed such a feature matrix which came down to around 5 million features and fed it to our implementation of NBSVM. To our great surprise, it actually worked and gave the expected accuracy of 0.921 [Table.1]. The testing accuracy on 30% testing data of this model is 0.925.

Discussion and Conclusion

Overall, the simple bag-of-words representation is still very effective in producing strong baseline accuracy. Further augmentations over this representation, like TF-IDF and n-gram, are demonstrated to improve the model performance. With respect to classifier, with no doubt that model combination, especially with complementary estimators, is the key in improving the accuracy. However, we are unsure about the necessity of data processing and feature selection, as our best model was trained on feature matrix with no feature selection and minimal processing (only removal of non-letter character). Intuitively, these two processes are supposed to help filter out words that are redundant or of low importance with the hope of increasing accuracy. However this may be the opposite case, as in that what we consider as unnecessary is significant for a machine learning model to produce the subtle distinctions.

In hindsight, although we could not deny the importance of techniques for data processing as well as feature construction and selection, the accuracy of our model was more contingent on the number of features, at least in the case of NLP machine learning problems. Potentially, if there is enough computational power, using lots of features (with regularization of course) and combining multiple models in soft voting scheme can achieve really high accuracy. This does not sound like old-fashioned science but rather brute force, yet many of the paper we saw during our search was able to achieve over 95% accuracy this way without using deep learning models.

Statement of Contributions

Linge was in charge of developing the Bernoulli Naive Bayes model, while Yan and Sitong collaborated on constructing and experimenting different data processing methods, feature constructions and classifiers. The report was splitted equally among the three members.

Table 1: Performance of different combinations of models and features matrices. The "best score" represents the mean of the 5-fold cross-validation score with best hyperparameters obtained in the same grid search. Models connected with a "+" sign are estimators in the same voting classifier. Note that NBLR is not constructed with a voting classifier and is treated as a single model that can be combined with other estimators in the voting classifier. In particular, the weight assigned to each estimator in a voting classifier is specified in the square bracket following the corresponding estimator.

	Best score
Raw Counts Uni-gram(min_df=5)	
Logistic Regression	0.883
TF-IDF Uni-gram(min_df=5)	
Logistic Regression	0.892
Logistic Regression (with Word-Embedding)	0.844
TF-IDF Tri-gram(min_df=5)	
MultinomialNB	0.894
LinearSVC (SGDClassifier with 'hinge')	0.902
Logistic Regression	0.907
MultinomialNB (0.5) +Logistic Regression (0.5)	0.909
NBSVM	0.912
NBLR	0.912
NBLR (0.6) +Logistic Regression (0.4)	0.914
NBLR (0.5) +Logistic Regression (0.2) +MultinomialNB (0.3)	0.916
Binary Counts Tri-gram(no min_df)	
NBSVM	0.921

References

[1] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th international conference on Machine learning, pages 160–167. ACM, 2008.

[2] T. Mikolov, M. Karafiat, L. Burget, J. Cernock ´ y, and S. Khu- danpur. Recurrent neural network based language model. In INTERSPEECH, 2010.

[3] Wang, Sida and Manning, Chris D. Baselines and bigrams: Simple, good sentiment and text classification. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, 2012.

[4] Maas, Andrew L, Daly, Raymond E, Pham, Peter T, Huang, Dan, Ng, Andrew Y, and Potts, Christopher. Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, pp. 142–150. Association for Computational Linguistics, 2011.

[5] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean.Distributed representations of words and phrases and their compositionality, 2013.

[6] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents, 2014.

[7] Mesnil, Gregoire Mesnil, et al. “Ensemble of Generative and Discriminative Techniques for Sentiment Analysis of Movie Reviews.” ICLR, 2015.

[8] Muller Andreas C, and Sarah Guido. Introduction to Machine Learning with Python: a Guide for Data Scientists. OReilly., 2017.