Sentiment Classification from Movie Reviews

S2316445 **S2448375** **S2313516**

Abstract

The work presented in this report focuses on the sentiment classification task. Using movie reviews on Rotten Tomatoes as data, we implement six data representation methods to transform the natural language sentences into vectors and classify them into different sentiment categories (positive and negative) with four machine learning models. The best outcome is achieved by Logistic Regression model with the TF-IDF transformation (AUC-ROC value of 0.772). We also apply the same methods to the multiclass classification tasks and discuss how the imbalanced data distribution affects the results.

1 Introduction

In recent years, the rise of movie platforms has provided great conveniences for users to share their feelings towards movies in the form of textual messages. These comments, which can also be called movie reviews, can help users to decide whether they would enjoy the movie or not. However, reading all of the available reviews could be somewhat time-wasting. In order to improve users' experience, some movie platforms provide users with a more visual overview system by transforming textual reviews into sentiment labels, like "positive" and "negative". This automated transformation process is also called sentiment classification in the natural language processing area.

Sentiment classification is one of the most researched topics of natural language processing. We explore several previous publications that study machine learning methods on identifying the sentiment of documents. Vivek et al. propose a enhanced Naive Bayes model in combination with methods like effective negation handling, word n-grams and feature selection by mutual information to improve the classification accuracy[6]. Huang et al. introduce words' syntactic properties into the basic words-bag features and get a more accurate solution for sentiment classification [12]. Khin and Thi propose an ontology based combination approach to enhance the existing approaches of the sentiment classification [8]. Apart from machine learning methods, deep learning methods, like Convolutional Neural Network, are also applied to classify sentiment of documents[2].

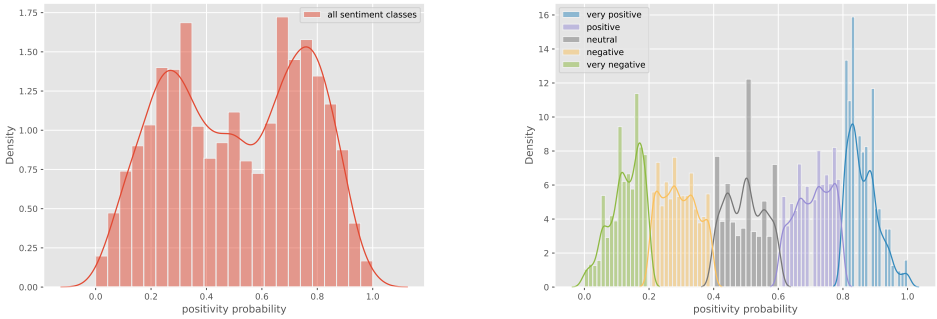
In this report, we will apply specific machine learning models, including K Nearest Neighbors, Naive Bayes, Decision Tree and Logistic Regression, to sentiment classification tasks to find the outstanding performance. We also study six different data vectorization methods (One-Hot, PCA, Word Count, Top-N, TF-IDF and Word2vec) and how these way we represent data would affect the classification results.

2 Exploratory Data Analysis

2.1 Data Introduction

We conduct our experiments on the Stanford Sentiment Treebank data set [9], which consists of a total of 215,154 unique phrases from movie reviews on Rotten Tomatoes. Each phrase is labeled with a positivity probability that shows the degree of sentiment that the phrase expresses. In that case, all the sentences can be divided into five classes by mapping the corresponding probability using the cut-offs ([0, 0.2], (0.2, 0.4], (0.4, 0.6], (0.6, 0.8], (0.8, 1.0]) for very negative, negative, neutral,

positive, very positive, respectively. We only use the instances that are complete sentences as our selected data set in order to simply the task. The selected data set includes 11,285 sentences and is split into training, validation, and test set. There are 8,116; 1,044; 2,125 instances in the training, validation, and test set, respectively.



(a) Kernel density for all training data

(b) Kernel density for training data in five sentiment classes

Figure 1: Kernel density for training data

2.2 Data Exploration

The pie chart in Appendix A shows the class distribution of the training data. We also use kernel density plots to represent the distribution of the training data. Figure 1 show the positivity probability distributions of all training data and data in each class, respectively. As shown in the Figure 1 1(a), the probability density curve has two peaks at about 0.2 and 0.8. These two values are exactly the partition boundaries of "negative and very negative" and "positive and very positive". We can also notice the similar trend in the Figure 1(b) that the peaks of the curves for very negative and negative data are close, which is same as the curves for positive and very positive data. It implies a possibility that there might be little differences between two sentences from two adjacent sentiment classes if their positivity probabilities are close to a partition boundary (i.e. 0.2 or 0.8).

We also use the wordcloud figures to visualize the word frequency of the training data. A wordcloud is a cluster of words depicted in different sizes. The bigger the word is in a wordcloud, the more frequently it appears within a text. Considering that there should be many neutral nouns like movie(s) and film(s) in the data, we remove those word firstly before generating the wordcloud figures. Figure 2 shows the wordcloud of all sentences and sentences in each class. Despite of some neutral words like "time", "story", "character", "work", "make" etc., there are still some words of high frequency that can embody the feature of the corresponding class. For example, we can see several positive and commendatory words like "best", "funny", "well" and "good" in the wordcloud of (very) positive data. On the contrary, several negative and critical words like "little", "bad", "n't" and "dull" are more frequent in the (very) negative sentences.

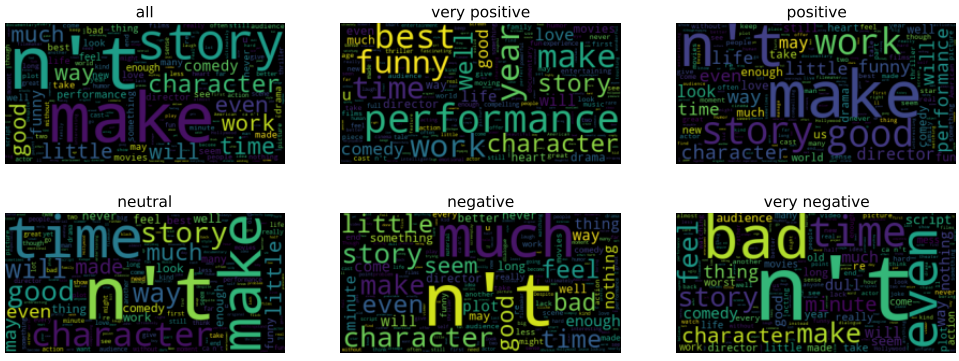


Figure 2: Wordcloud of all training data and of data in five sentiment classes

3 Preprocessing & Vectorization

The raw data are movie reviews collected from the Rotten Tomatoes platform, which means these sentences are all in natural language and cannot be used as the inputs of a machine learning model directly. Moreover, we should also filter some less relevant information and highlight certain features in the data, which are necessary and useful for the downstream tasks. In our experiments, we first implement data cleaning, stop words filtering and lemmatization in the data preprocessing stage. And then we use six different methods to vectorize the natural language sentences.

3.1 Data Preprocessing

Data Cleaning The first step is to clean the raw sentence data. Apart from lowercasing characters and removing all the characters that are not English alphabet and space, we also replace all the "ca" with the word "can" and replace all the "n't" with the word "not". This is because we find the words "can't" are all divided into "ca" and "n't" in the data set and these two words can, to some extent, carry the sentiment of a sentence.

Stop Words Filtering The next step is to remove the stop words in the sentences. Stop words are those really common and frequent words in a document. We can always remove these words because they convey little insights into the specific topic of a document. Our stop word list is based on the one in Natural Language Toolkit Library (NLTK). On top of that, we remove several words ("no", "not", "very", "few", "too", "most" and "more") that carry sentiment characteristics from the list.

Lemmatization The last step is to lemmatize words. A sentence is likely to consist of different forms of a word owing to grammatical reasons. Additionally, there are families of derivationally related words with similar meanings, such as democracy, democratic, and democratization. [3] In that case, it would be useful to condense a sentence if we reduce inflectional and derivationally related forms of a word to a common base one. It will not only help with concentrating the feature words in a sentence but can also reduce the dimentionality of the sentence.

3.2 Vectorization Methods

One-Hot One-hot encoding is a simple and intuitive method on converting discrete feature variables into numerical vectors which encompasses with 1 and 0 values. However, since each word is controlled by one binary value independently, its vector would comprise a huge amount of numeric, which could be very lengthy.

Principal Component Analysis (PCA) PCA is implemented to reduce the dimensionality of our data set that are vectorized by One-Hot encoding. By projecting each data point onto only the first few principal components, PCA method can transform the high-dimensioanl data into a lower one while preserving as much of the data's information and features as possible. [11] As we can see the Figure in Appendix B, when we choose the first 3,000 principal components, we can still preserve over 90% of information of the original data.

Word-Count Word-count encoding is an application of the Bag Of Word model. Unlike the One-Hot encoding, Bag Of Word model will take the frequency of words into account. Although the Word-Count encoding could extract more information of a sentence than One-Hot encoding, it still disregards grammar and even word order and does not add the context information into the sentence representation.

Top-N Top-n encoding is a data representation method proposed by ourselves, which could be regarded as a simplified version of the Word-Count encoding. It only uses the most n frequent words as the features of training data. This method is proposed on the intention to reduce the dimensionality of the vectorized sentence while keeping the main word features.

TF-IDF TF-IDF encoding tends to choose meaningful words and filters some words with high frequency but of less importance, like "a" and "the". The formula could calculate the words' weight:

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

where $n_{i,j}$ is the frequency of one specific word i in a movie review j, and the denominator represents the total number of words in the movie review j.

$$idf_i = log\frac{N}{df(i)} \tag{2}$$

where N is the total number of movie reviews and $df(i)$ is the number of movie reviews including the word i. Then the words' weight could be calculated by using the formula: **TF-IDF** $= tf_{i,j} \times idf_i$

Word2Vec Word2Vec is one popular way of Word Embedding. It depends on neural networks to assign words with similar meanings into vector space, making them point in a similar direction[4]. In our experiments, we use Gensim Python library to train our own Word2vec model on the training data. The dimentionality of word vectors is set as 200 and we use the mean vector of all the word vectors in a sentence to represent this whole sentence. We note that we train the Word2vec model on the training data without stop words filtering because the training process of Word2vec model depends on the context information.

4 Sentiment Classification

In this section, we focus on the sentiment classification task. The goal is to predict the sentiment label of a given sentence from a movie review. We build our classifiers based on four different supervised machine learning algorithms separately, including K Nearest Neighbors, Naive Bayes, Decision Tree and Logistic Regression. It should be noted that although the original data are initially labeled with five sentiment classes as we mention in section 2.1, we mainly focus on classifying the data into two sentiments, negative and positive, to simplify the task. Multiclass classification tasks are only discussed in section 4.5.

4.1 Classification Algorithms

K Nearest Neighbors will assign a instance to the class that is the most common one in its nearest K neighbors. The value of K is the only hyperparameter in this algorithm. If K is too small, the result would be very sensitive to its neighbors and be easily effected by the noise data. If K is too large, the model would become too simple and thus underfitting. In our experiment, we compute the Euclidean distance between two instances and set the value of K as 100 according to the different results on the validation data.

Naive Bayes strongly depends on the Bayes' theorem and the independence assumption. The classifier assume that every pair of features of the given data set are conditionally independent and seeks to choose the most probable class given the observation.

Decision tree could be used for nonlinear classification. The classifier recursively partitions the input feature space by computing and reaching the maximum purity [5] and then defines local models in each of the resulting regions. In the experiment setting, we choose to compute the entropy to measure the quality of a split and set the maximum depth of the tree as 200.

Logistic regression is a generalized linear model and that estimates the probability of an event to take place. The model can be regarded as a linear combination of independent variables of the data and it uses a logistic function to model the probabilities of the output.

4.2 Metrics

F1 score and Area Under the Receiver Operating Characteristic Curve (AUC-ROC) are two widely used metrics when it comes to classification tasks. We consider that AUC-ROC [1] aims at reducing the proportion of non-true samples that are positive, namely false positives, while F1 score hopes to increase the proportion of samples that test positive are actually true. We conclude that AUC-ROC tries to train a model that does not misreport as much as possible and F1 score plans to train a model that does not miss any possible, which is more useful in the case of detecting disease. Moreover, AUC-ROC does not depends on the choice of threshold. Hence, we use AUC-ROC value as our data is balanced in the binary classification task and we are more worried about the accuracy of prediction.

4.3 Binary Classification

This section covers the implementation details of our experiments. Considering that the positivity probabilities of many training data are close to the values of 0.2 or 0.8, it may be comparatively difficult for our classifiers to distinguish the differences between negative sentences and very negative sentences or the differences between positive sentences and very positive sentences. Therefore, we just simplify the classification task to a binary one. In that case, a sentence with the positivity probability that is larger than 0.5 would be labeled as positive, if not, it would be a negative one.

All the data are preprocessed before vectorization, but the vectorization transformations are only fitted on the training data. In the implementation, we use six methods mentioned in section 3.2 to represent the data separately. After vectorization, we build four classifiers (K Nearest Neighbors, Naive Bayes, Decision Tree and Logistic Regression) and fit them with training data. We also use the results of the vaidation data to choose the hyperparameters of those models. At last, we take the optimised models and evaluate their performance on the test data.

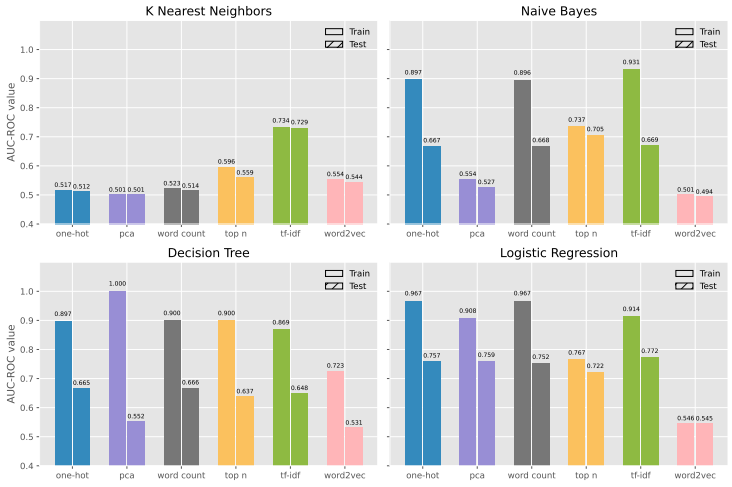


Figure 3: Model performance on the binary classification task

4.4 Results and Discussion

Figure 3 shows the binary classification results produced by different combination of six vectorization methods and four machine learning classifiers. The best performance on test data was achieved by Logistic Regression model with TF-IDF transformation with AUC-ROC value of 0.772. From the perspective of classifiers, the Logistic Regression model outperforms the other four models and the K Nearest Neighbors model is the worst performing one in general. It should be noted that, although the Decision Tree model performs quite well on the training data and even the AUC-ROC value reaches 1.0 when using the PCA method, its results on the test data are not satisfactory. It may just suggest that the model is overfitted.

When we compare the results of different vectoration methods, we can notice that TF-IDF transformation performs well on all models in terms of AUC-ROC values. A possible reason is that TF-IDF takes both term frequency and document frequency into account, which means that TF-IDF also assigns a higher weight to words that occur only in a few documents. To be more specific, audiences may use some unique words to describe their feelings about the movie and TF-IDF is able to capture these less common but useful features. Contrary to the TF-IDF, Word2vec seems not to be suitable for this task for the corresponding AUC-ROC values are all close to 0.5 no matter in combination of what classifiers. We argue that this is probably because our dataset is relatively small and does not provides enough examples for the Word2vec model to learn the features of words. In that case, the model could only produce word vectorization results of poor quality.

As we have expected, the results of applying PCA with One-Hot encoding are very close to those of One-Hot encoding when we use the Logistic Regression model. The results indirectly demonstrate

that PCA could reduce the dimensionality of a dataset while preserving as much information as possible. Similarly, Top-N only chooses most n frequent words as the features of training data and is another way to reduce dimensionality. Therefore, its results are also close to those of Word-Count.

4.5 Multiclass Classification

Now, let us move forward to a more challenging task, multiclass classification. We conduct our experiments on the same data set for multiclass classification with the same preprocessing steps. The vectorization methods and classification algorithms we use are all same as those in the binary classification task. The only difference is how to map the positivity probalities for different sentiment classes. As for the five-class situation, we just follow the original cut-offs for the mapping. For the three-class situation, we map the corresponding probability using the cut-offs ([0, 0.4), [0.4, 0.6], (0.6, 1.0]) for negative, neutral and positive, respectively. Moreover, as we can see in the pie chart in Appendix A, if we divide data into more than two classes, the class distribution will become imbalanced. Therefore, we would like to use accuracy instead of AUC-ROC value as the evaluation metrics for the multiclass classification tasks.

As we have expected, the model performance is much worse than that on the binary classification task. The classification accuracy on the test data is shown in the Appendix C. Even though the Logistic Regression model with TF-IDF vectorization still has the best performance, the accuracy only reaches 66.1% for the three-class classification task, which is nearly 10% less than that for the binary classification task. The accuracy for the five-class classification task is even lower, which only reaches 41.5%. We suppose that the unsatisfactory result is partially because of the imbalanced class distribution in the training data. Additionally, it should be more difficult for the model to distinguish the little differences between two sentences from two adjacent sentiment classes if their positivity probabilities are close to a partition boundary.

Furthermore, although the accuracy of multilabel classification tasks decreases, the AUC-ROC values are surprisingly high. It motivates us to study the underlying logistics and the difference between these two metrics. We learn that the ROC curve aims to record the tradeoff which commits to various decision thresholds between false positives and false negatives [10], which means each point in our ROC curve represents a particular threshold. Therefore, the AUC value comes from a series of thresholds. However, the accuracy is calculated under an unchanged threshold, a point from the ROC curve. As a result, a model with a high ROC-AUC value and low accuracy is possible.

5 Conclusions

This paper focuses on the sentiment classification task along with the related preprocessing work. How we vectorize data and choose classifiers directly affects the final classification result. In the experiments, we apply six different methods to convert natural language movie reviews into numeric vectors and combine each of them with four supervised machine learning methods to make the binary classification (positive and negative). We achieve the best performance with Logistic Regression model in combination with TF-IDF vectorization method and the corresponding AUC-ROC value reaches 0.772. We also apply the same methods on the multiclass classification tasks. Although the results are less satisfactory, they still prove the effectiveness of our methods and we argue that the imbalanced class distribution in the training data is one of the potential reasons behind the results.

For the future work, we would like to make use of the information in parse trees related to the data to improve the performance. For example, we could divide the whole sentence into several phrases according to its parse tree and combine the sentiments of each phrase to get the final sentiment classification result of a sentence. Moreover, we notice that the "thwarted expectations" phenomenon [7] where the author sets up a deliberate contrast to earlier discussion in a sentence would make the classification task more difficult for classifiers. Therefore, we plan to explore more state-of-the-art methods, especially some deep learning methods, to see whether they can handle this more challenging classification task.

References

[1] Muschelli J. Roc and auc with a binary predictor: a potentially misleading metric. *J Classif*, 37(3), 2020.

[2] Hannah Kim and Young-Seob Jeong. Sentiment classification using convolutional neural networks. *Applied Sciences*, 9(11):2347, 2019.

[3] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, USA, 2008.

[4] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013.

[5] Anthony J. Myles, Robert N. Feudale, Yang Liu, Nathaniel A. Woody, and Steven D. Brown. An introduction to decision tree modeling. *Journal of Chemometrics*, 18(6):275–285, 2004.

[6] Vivek Narayanan, Ishan Arora, and Arjun Bhatia. Fast and accurate sentiment classification using an enhanced naive bayes model. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 194–201. Springer, 2013.

[7] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 79–86. Association for Computational Linguistics, July 2002.

[8] Khin Phyu Phyu Shein and Thi Thi Soe Nyunt. Sentiment classification based on ontology and svm classifier. In *2010 Second International Conference on Communication Software and Networks*, pages 169–172, 2010.

[9] Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[10] Wanhua Su, Yan Yuan, and Mu Zhu. A relationship between the average precision and the area under the roc curve. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, ICTIR '15, page 349–352, New York, NY, USA, 2015. Association for Computing Machinery.

[11] Wikipedia contributors. Principal component analysis, 11 2022.

[12] Huang Zou, Xinhua Tang, Bin Xie, and Bing Liu. Sentiment classification using machine learning techniques with syntax features. In *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 175–179, 2015.

A Class Distribution

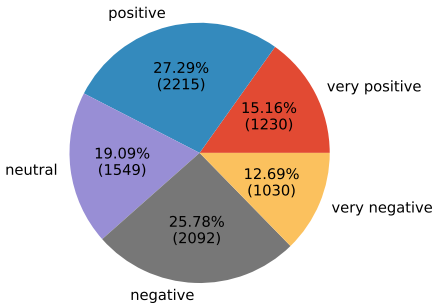


Figure 4: class distribution of training data

B PCA

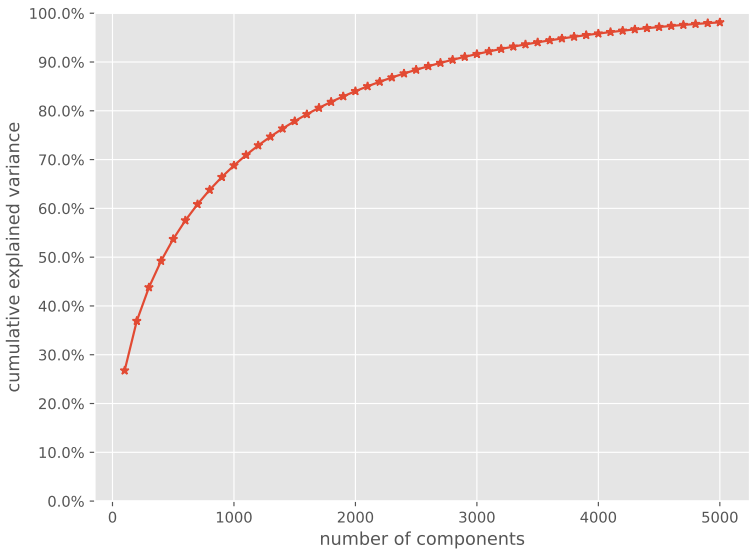


Figure 5: Cumulative Explained Variance

C Multiclass Classification

Table 1: Test Accuracy of the three-label Classification task

Model	One-Hot	PCA	Word Count	Top N	TF-IDF	Word2Vec
KNN	0.336	0.400	0.352	0.490	0.624	0.410
NB	0.440	0.281	0.441	0.480	0.439	0.373
DT	0.544	0.414	0.546	0.480	0.439	0.373
LR	0.644	0.637	0.640	0.604	0.661	0.412

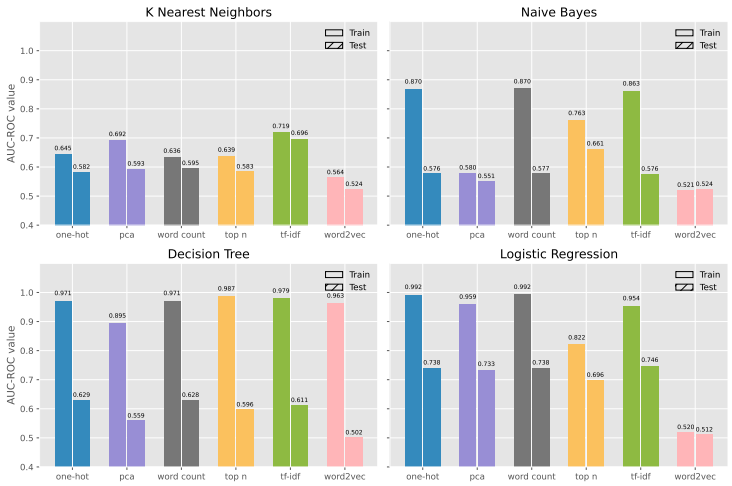


Figure 6: The Auc-Roc Value of the three-label Classification task

Table 2: Test Accuracy of the five-label Classification task

Model	One-Hot	PCA	Word Count	Top N	TF-IDF	Word2Vec
KNN	0.185	0.179	0.192	0.201	0.378	0.257
NB	0.277	0.232	0.278	0.249	0.279	0.177
DT	0.308	0.244	0.306	0.296	0.306	0.216
LR	0.415	0.395	0.409	0.385	0.415	0.233

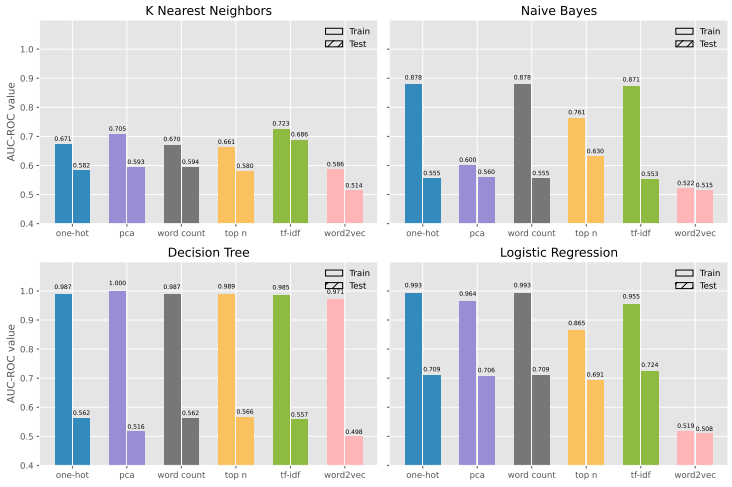


Figure 7: The Auc-Roc Value of the five-label Classification task