CMPE462 - Project Report (Step 1)
Sentiment Analysis on IMDB User Reviews

Nursima Çelik, 2016400090
Meltem Arslan, 2016400117
Feyza Duman, *2016401123*

I. LITERATURE REVIEW

We used paperswithcode.com for literature review. In Papers with Code website, there are rankings for various data sets. The related one is Sentiment Analysis task on IMDB user reviews corpus. Among the best models, we immediately filtered out the ones that use neural networks, language models, or word embeddings.

Leader method suggests Naïve Bayes weighted bag of n-grams, and this gives 97.4% accuracy. They propose using cosine similarity instead of dot product while vectorizing documents. But we found out that this article also uses neural networks [1]. So, we could not use this approach, but we inspired that we could use Naïve Bayes.

Although there are many methods, the following articles draw us the ways we could use in the project:

We also checked Naive Bayes and Sentiment Classification chapter of Speech and Language Processing book. We concluded that we could try Binary Naive Bayes as well [2].

Also, another paper has used logistic regression and SVM on IMDB reviews to test their semantic vector representation of the words. They tried to make the words with similar expressions to have a similar vector representation [3]. So, we have learned that vector representation is important and SVM and Logistic Regression can be applied in this problem.

There is also a paper which tests hybrid feature extraction method with 4 different models on IMDB reviews. These models are SVM, KNN, Logistic Regression, and Maximum Entropy. Besides, it mentions about machine learning based feature extraction which is TF-IDF and suggests hybrid feature extraction models which combines machine learning based and Lexicon based feature extraction [4]. Thus, we wanted to used TF-IDF in our project.

As we found out that most of the novel approaches use advanced techniques, we thought that we should start experimenting with well know methods like Logistic Regression, Naive Bayes, and SVM.

II. DATA AND FEATURES

IN this project, we have previously collected user reviews on IMDB website, whose primary language is English. The reviews are collected on individual txt files with 3 labels that shows the user rate to that movie as positive, negative or neutral. Each file for a sample has a name of "xxx_y.txt", where "xxx" resembles the number of that sample and "y" resembles the label of that sample.

We have 2 data sets; one is for training the model and the other is for testing the model. Training data set includes 3000 samples, and the testing data set includes 750 samples.

To see the frequency of the most used words for each label, we used word cloud. The Word cloud of the 3 classes can be seen at Fig. 2.

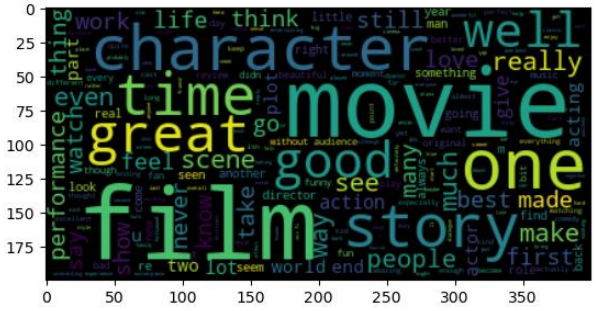


Fig. 2.1. Word Cloud of the positive class.

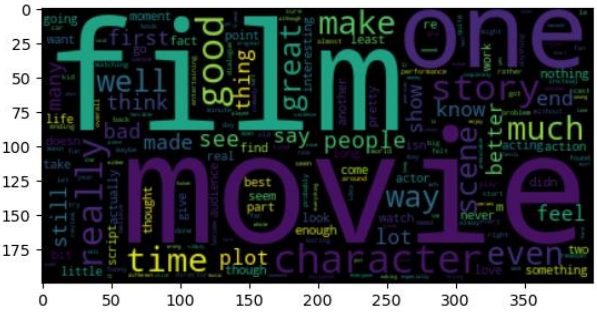


Fig. 2.2. Word Cloud of the neutral class.

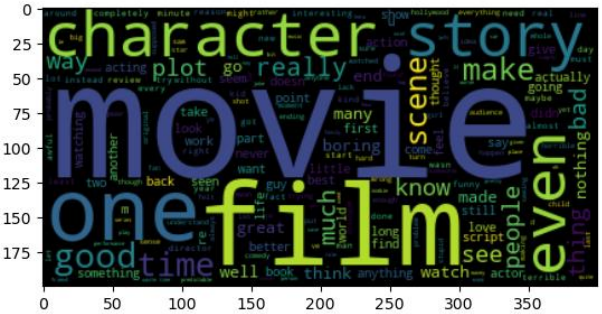


Fig. 2.3. Word Cloud of the negative class.

Also, there are 27077 different words that are used in training data. So that, our feature vector has 27077 attributes. The ratio of features to sample number is high and this coincides with the usage of State Vector Machines (SVM) and logistic regression.

Before training the model, we should have cleaned the data. Because the upper-case letters do not have an importance, we change them into the lower-case letters. Besides, we replaced the punctuations with white space in the texts. But we had another issue with some special situations like line break element which is represented in the text as "<br\>". So, we removed some observed special situations.

In the model samples we examined, some words named as stop words were omitted because of their meanings, as it was thought that they did not contribute to the model, such as "I", "do", "movie", etc. To see the effects of removing stop words from attributes, we trained our model with stop words and without stop words.

The meanings of words are mostly come from the root of the word in English language. For example, "s" makes nouns plural, and this plurality does not have any contribution as new attributes. We tried two methods for the reduction the words to the roots: lemmatization and stemming. While the lemmatization operation gives us the roots of the words, stemming operation deletes the inflections of the words.

Also, this causes a reduction in the number of attributes of feature vector. By removing stop words, the number of different words reduces to 26950. When lemmatization or stemming is applied in addition to the removing of stop words, the word number decreases to 24050. So that, this represents the the number of attributes of feature vector reduces to 24050.

III. Classification Models

In this project we have tried 5 different classification methods. In the first two method, we used Naïve Bayes algorithm with the frequencies of the words in the reviews and with the binary existence representation. In the third method we used Support Vector Machines (SVM) instead of Naïve Bayes with word frequencies, while for the fourth model, we again used SVM but for this time we vectorized our words with TF-IDF instead of using just frequencies. As a last model, we used logistic regression with TF-IDF. We will compare the results of each model at Part 4. The methods explained briefly in this part of the report:

A. Naive Bayes (NB)

It is a classification technique that based on Bayes' rule in probability. Also, it assumes that each attribute is probabilistically independent from the others. This assumption makes this method Naïve. It calculates the conditional probabilities of the words in the sample by using the frequencies of the words in the labels.

B. Support Vector Machine (SVM)

SVM aims to divide the d dimensional space (where n represents the number of attributes), and the sub-hyperplanes represent each class. It tries to maximize the distances from the sample points to the separation lines (support vectors), so that the error risk of new data points near the boundaries is reduced.

C. Logistic Regression (LR)

Logistic Regression tries to maximize likelihood function. It wants to increase the probability of being in correct class, in other words it tries to make the samples further from the separation boundaries of the classes.

D. TF-IDF

While we can vectorize the data with respect to the frequency of the words (Count vectorization), we can also vectorize by using TF-IDF method. It statistically measures the relevance of a word in a file with respect to the aggregate of the files. TF-IDF equals to the multiplication of term frequency (TF) and inverse document frequency (IDF).

IV. Word Embeddings

Pretrained word embeddings are mapping from words to vectors obtained by training on large corpuses like Wikipedia, Twitter, Common Crawl, etc. The mapping from words to vectors preserves the semantics and usage of that word in the language.

Until now, we used Count vectorizer, TF-IDF for vectorization before giving it to the models. For Step3 of the project, we wanted to integrate pretrained word embeddings to our pipeline. We tried Glove, Word2vec and BERT for this purpose. We tried to pick the best kernel type and C values for each SVM combinations.

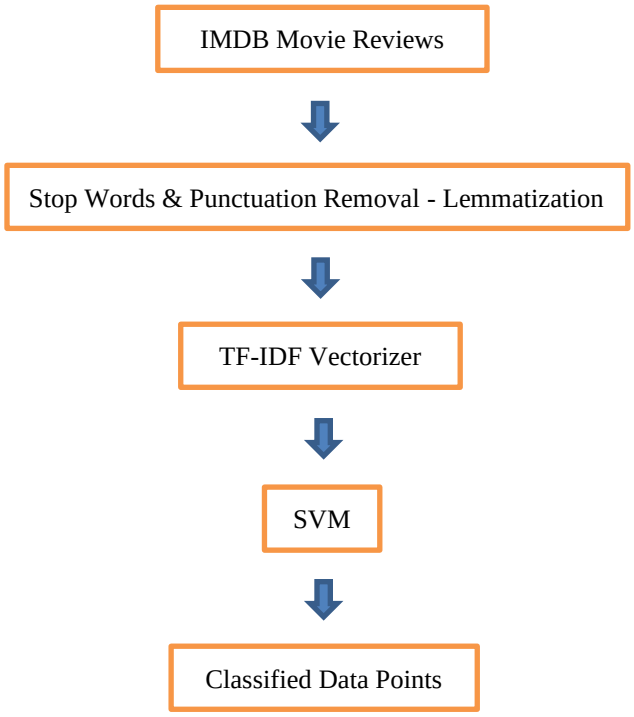
Bert + SVM: First, cleaned data is tokenized by using BERT tokenizer. Then we got the ids of the tokens and we vectorized the data with respect to the frequency of the ids of the tokens. Lastly, we applied SVM on the vectorized data.

BERT + TF-IDF + Logistic Regression: Firstly, we got tokens of the comments by using BERT Tokenizer. Then we supposed that these tokens are units, and we constructed the data by using this units as words. After that, we vectorized the data by TF-IDF, and we classified the vectorized data by Logistic Regression.

Glove + SVM: By using gensim library, we downloaded Glove embeddings which is pretrained on Wikipedia 2014. We compared two versions of Glove consisting of 50d and 300d vectors. For each review, we constructed a new vector by averaging the vectors of all words in that review so that we obtained 50d(or 300d) vector for each review. In result, a matrix with dimension (50(or 300) x #ofreviews) is fed into SVM.

Word2vec + SVM: First, we installed 'en_core_web_sm' model by using spacy library. This model was trained on OntoNotes 5. The library's function "nlp" directly gave us a vector for each review which is the average vector of the words on that review. Then we applied SVM on this vectorized data.

V. System Diagram



VI. Results

As a result, we trained the models on training set and get performance metrics on both training and validation data sets. We start with eliminating stop words and reducing the words to their root by Lemmatization. Because our aim is the best accuracy on validation set, we give accuracy results of the models on the Table 1.

(w/o stop words)	Train accuracy	Validation accuracy
SVM + Count vector.	91.4%	62.5%
Naïve Bayes + Count vector.	92.9%	64.5%
Naïve Bayes + Binary Vector	93.8%	66.0%
Logistic regression + TF-IDF vector.	99.6%	68.0%
SVM + TF-IDF vector.	99.6%	69.9%
SVM(C=10) +Glove (50d)	62.1%	52.4%
SVM(C=10) +Glove (300d)	58.4%	53.9%
SVM(linear kernel, C=4) + BERT	100%	58.0%
Logistic Regression + BERT + TF-IDF	94.6%	68.7%
SVM (rbf kernel, C=15) + Word2vec	64.5%	48.1%

Table 1. Accuracy of the models on the training and on the validation sets. *All models were trained with lemmatization except LR+BERT+TF-IDF

From table 1, we can say that SVM with TF-IDF vectorization works best in these models. Both the train accuracy and the validation accuracy have the highest values.

The Count vectorization outputs are not so distinguishable on space, so that SVM need very long time for running. On the other hand, TF-IDF gives more grouped vectors. Thus, SVM with TF-IDF vectorization is faster and it gives better results.

Binary Naïve Bayes gives better results than Naïve Bayes. This can be because of the probability calculation is not divided for many states, so making the probability higher gives better results.

When the training set is large as in our situation, SVM and Logistic Regression gives better results than Naïve Bayes. Also, SVM and Logistic Regression gives very close results for training set, however SVM works better for validation. This can be because of both sets very accurate results in training but SVM tried to find best margin, so this reduces the risk of error. Besides, the performance metric results can be seen at Table 2.

(SVM + TF-IDF vector.)	Training set	Validation set
Accuracy	99.6%	69.9%
Precision (positive class)	0.999	0.798
Precision (negative class)	0.992	0.726
Precision (neutral class)	0.996	0.585
Recall (positive class)	0.996	0.804
Recall (negative class)	0.998	0.648
Recall (neutral class)	0.994	0.644
Macro Average Accuracy	0.997	0.799
Macro Average Precision	0.996	0.703
Macro Average Recall	0.996	0.699

Table 2. Accuracy of the models with Lemmatization on the training and on the validation sets.

Because all performance metrics are almost higher with respect to the other models, we do not give all the result to avoid a long report.

We can observe from Table 2 that the highness of the metric values of training set means that the system almost separates the hyperplane to sub-planes with very small error. However, when we test the system on validation set, the performance metric values drop because the system coincides with new inputs.

Besides, the precision values are higher for positive and negative classes than neutral classes. It is because there is no direct boundary between neutral and positive class, and between neutral and negative class. A review in the neutral class can likes the movie but because of some scenes or some mistakes the rate of the review can be lower. It can be also true for vice versa. So that, a neutral review can be close to positive or negative class and it is very normal that the precision for that

class is lower than the other two classes. The same logic also works for recall. Also, precision and recall for positive class is much higher than macro average precision and macro average recall. This means that the model detects positive reviews better than the overall detection.

On the other hand, we continue with the current best model (SVM with TF-IDF vectorization) to observe the effects of stop words and Lemmatization. We gave the results on Table 3.

(SVM + TF-IDF vector.)	Train accuracy	Validation accuracy
Stemming + w/o stop words	99.5%	68.8%
Lemmatization + w/o stop words	99.6%	69.9%
Lemmatization + w/ stop words	99.0%	66.3%

Table 3. Comparison of accuracies of the SVM model with TF-IDF vectorization.

We can observe that after reducing stop words, the method of Lemmatization gives validation accuracy of 69.1%, while the method of Stemming gives validation accuracy of 68.8%. So that Lemmatization works better, and this shows that real roots of the words are more important in the reviews. Besides, deleting stop words make this model better as we can see at Table 3. If we do not delete the stop words, we get an accuracy rate of 66.3%, which is obviously smaller than accuracy rate of 69.1% which we get by deleting stop words.

For the improvement of the model, we wanted to use pretrained word embeddings as a part of the vectorization of the data. We applied BERT, Word2vec and Glove on the data in that purpose. We were expecting to increase the accuracy by using these pretrained word embeddings as they are more sophisticated than TF-IDF however, we got 58.0% as the best validation accuracy which is lower than our previous best model result. After this result, we also tried to combine BERT with TF-IDF, but it also gave a lower accuracy which is 68.7%. Since we could not improve our model by using pretrained word embeddings we increased model accuracy to 69.9% by changing stop words set and we updated the best accuracy in the table accordingly.

VII. Conclusion

We began with using Naive Bayes since it is the one that we are most familiar with. Since we need to compare the results, we had to try different methods. Our second method was Logistic Regression. We have met logistic regression in binary classification before, but we could use it also for the multinomial classification. However, the results were not good enough for us, so we continued with another method, Support Vector Machines, that we met while reviewing literature. Since it reduced the error risk of new data points near the decision boundary, it gave us the best accuracy on both the training and the validation data sets.

The vectorization part was also one of the most important point of the classification. Instead of Count vector which vectorize as word frequency, we tried another vectorization method: TF-IDF vector. With TF-IDF vector we were not just using the frequencies of the words(features), we were also considering the inverse document frequency of a word across a set of samples. It worked better than Count vector, gave us the maximum accuracy we had on validation set: 69.1%.

We also compared the results with removing stop words and without removing stop words. So that we trained the model by deleting stop words from the feature list since the accuracy was lower when we trained with them. We also trained our model "SVM with TF-IDF vectorization" with the features obtained from stemming operation but the accuracy was lower than lemmatization, so we decided on lemmatization. We did all the tasks together from Zoom by sharing our screens sequentially.

References

[1] T. Thongtan, T. Phienthrakul, "Sentiment Classification Using Document Embeddings Trained with Cosine Similarity," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, 1st ed., F. Alva-Manchego, Ed. Florence, Italy: ACL, 2019, pp. 407–414.

[2] D. Jurafsky, J. H. Martin, *Speech and Language Processing*, 3rd ed. draft. 2020, pp. Ch. 4, 1–23.

[3] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, C. Potts. "Learning Word Vectors for Sentiment Analysis" in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, Portland, OR: ACL, 2011, pp. 142–150.

[4] H. M. Keerthi Kumar, B. S. Harish, H. K. Darshan, "Sentiment Analysis on IMDB Movie Reviews Using Hybrid Feature Extraction Method" in *Int. J. Interact. Multim. Artif. Intell.* Vol. 5, No. 5, 2018, pp. 109-114

Model Link:

https://drive.google.com/file/d/1UzphcjGzuVXPEL9fEazIoGHZt3um__wZ/view?usp=sharing