COMP 551 - WINTER 2022

APPLIED MACHINE LEARNING

MiniProject 2 Report

Classification on News20groups and Sentiment140 Datasets with Guassian Naive Bayes and Logistic Regression

Group 70

George Qiao	260779462
Atia Islam	260737946
Vraj Patel	261022581

School of Computer Science, McGill University

March 7, 2022

Abstract

In this project we compared the performance of Naive Bayes and logistic regression on News20Group [1] and the Sentiment140 [2] datasets. We explored text preprocessing for natural language, feature extraction with CountVectorizer() and TfidfTransformer() from Sci-kit Learn, and the hyperparameters for logistic regression including solver and penalty types. Gaussian Naive Bayes model achieved 43.0% test accuracy on News20Group and 69.9% test accuracy on Sentiment140; for Sentiment140, the test accuracy values were 63.1% and 69.1% respectively. The investigation also confirmed our hypothethis that model performance increases as the fraction of training data used increases.

Introduction

The goal of this project is to implement and explore the performance of two classifiers, namely Naive Bayes and logistic regression, on two different datasets, namely the News20Group [1] and the Sentiment140 [2]. The task is to classify the category of a news piece for the first dataset, and to classify the sentiment (positive/negative) for the second. An important aspect of the project is preprocessing and feature extraction on the text data, where key decision are made in the process of string manipulations and techniques such as CountVectorizer and TfidfTransformer. We explored different hyperparameters (different solvers, max_iter, etc.) for logistic regression. Final results show moderately superior performances for both models on Sentiment140, with test accuracy around 69-70% compared to 43-63% on News20Group.

Datasets

For News20Group dataset, we removed 'headers', 'footers' and 'quotes' columns from both train and test sets [1]. We extracted the 'text' (x) and 'target' (y) columns. We performed basic analysis on the set to see what is the total number of words per text, total number of sentence and average length of words per text. We plotted histogram for the total number of words see the frequency of words per class. We also created bar plots to see class distribution. We remove the alphanumeric along with any punctuation from the texts. Proceeding to the next step, we transform text strings to counts using CountVectorizer. We use tfidfTranformer to transform counts to term-frequencies (tf) or term-frequency-inverse-document-frequency (tf-idf) which highlights the importance of each unique term.

In Sentiment140 dataset, we have tweets from Twitter API [2]. We plot a bar graph to see the distribution of the two classes. We confirm from this plot that the class is balanced and data is with little skewness. However, since there are expressions in regular speaking language that does not pertain any significance we needed to preprocess the text. We remove digits, stopwords, url, hyperlinks, mentions, punctuation and hashtags using regex. We proceed to apply CountVectorizer and tfidfTranformer the same way. We also use wordCloud to see if there are any differences in how these two different methods generates frequency for our dictionary of features. We use N-grams (eg, unigrams, bigrams and trigrams) to analyse the most frequent 25 words and plot the frequencies of this features.

The following figures outlines key statistics from the two datasets. Please also see the Appendix for figures on class distributions.

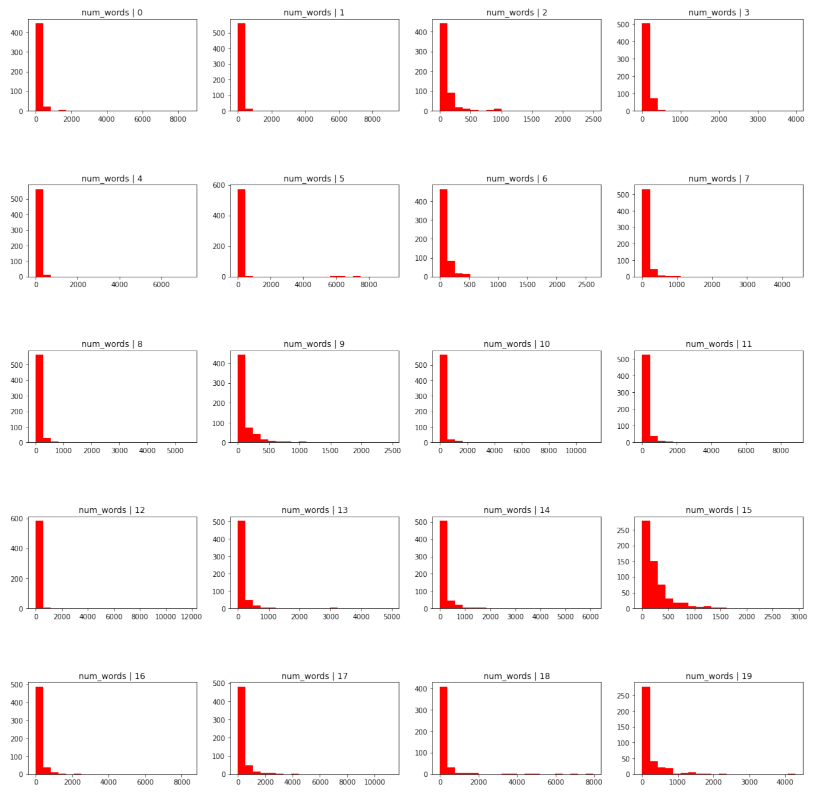


Figure 1: Word Frequency for Each Class for News20Group

Word cloud is another nice way to represent term frequencies. We have computed one for Sentiment140 dataset as shown here (code in Data_Analysis.ipynb):

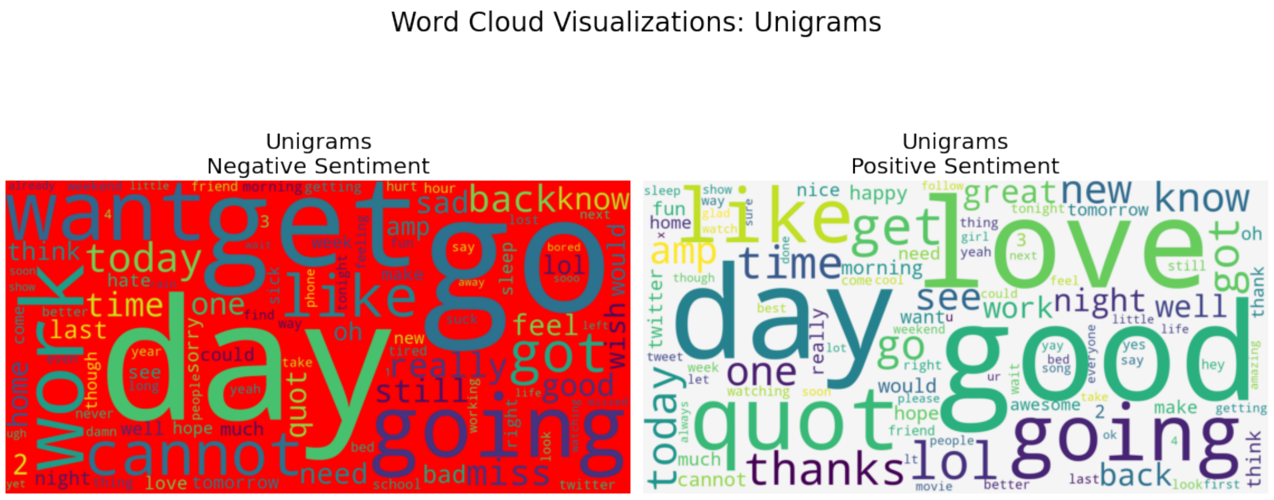


Figure 2: Word Cloud for Sentiment140 (Unigrams)

Please also see the Appendix for more figures.

Ethical Implications

In the research process we must upload human rights including the right to not have arbitrary invasion of privacy. Unfortunately, covert research and underground sources have been prevailing in the research market where consent-seeking is a far cry for help. For example, the Sentiment140 dataset, when used by large corporations for potential commercial gain, could potentially negatively impact certain groups of people. Hence, it is important we protect the privacy of the users where these Twitter data came from (i.e. only maintaining information required for research objectives) and strictly follow relevant guidelines for dissemination of data.

Results

We implemented train-validation split through the cross_validation_split function, returning indices. Then we run experiments in a for-loop to achieve cross-validation (we did not specifically name it kFoldCV, but our for-loop essentially serves the same purpose). Naive Bayes model does not have any hyperparameters; we implemented Gaussian Naives Bayes due to using tf (otherwise need to use multinomial for counts). This could potential be a ”hyperparameter”, as discussed in Future Work section later. For logistic regresssion, the hyperparameters we considered are solver types (newton-cg vs. lbfgs vs. sag vs. saga) and penalty types (’l1’ vs. ’l2’ vs. ’none’).

An important hyperparameter in CountVectorizer was the minimum cut-off frequency of words to be included in the dictionary (its purpose is to exclude rare words that don’t have enough data to train and thus to avoid over-fitting). We considered values between 0.001 and 0.010. This is a delicate balance since larger values give us too few features to achieve acceptable accuracy and smaller values give us too many features to run the experiments in a reasonable amount of time with limited computing resources.

Summary of results

- For News20Group, final parameter for CountVectorizer() is min_df=0.001 which gives 8068 features.
- For Sentiment140, final parameter for CountVectorizer() is min_df=0.004 which gives 226 features.
- News20Group with Gaussian Naive Bayes: 84.4% training accuracy, 43.0% test accuracy.
- News20Group with logistic regression ('newton-cg', 'l2'): 91.5% training accuracy, 63.1% test accuracy.
- Sentiment140 with Gaussian Naive Bayes: 67.4% training accuracy, 69.9% test accuracy.
- Sentiment140 with logistic regression ('lbfgs', 'l2'): 68.4% training accuracy, 69.1% test accuracy.

Comparing performance between two models, we see that logistic regression is more accurate on News20Group but Naive Bayes slightly edges out its opponent on Sentiment140. We expected logistic regression to win out overall due to its weaker assumptions (Naive Bayes assumes the distribution of data).

Comparing performance between the two datasets, we find that higher test accuracy is achieved on Sentiment140, probably due to its abundance of data (1.6 million entries). On News20Group, we seem to have some over-fitting issues (high train but low test accuracy). We did test several min_df values between 0.001 and 0.005 but could have tuned it finer so that we have slightly fewer features. In addition, Sentiment140 has much fewer classes than News20Group (2 compared to 20) which might help it achieve higher accuracy.

Unfortunately we did not manage to investigate the hyperparameters for logistic regression more thoroughly: for the first dataset only 'newton-cg' converges while for the second only 'lbfgs' converges (in 500 max iterations). In addition, both solvers only support 'l2' penalty.

Effect of using varying fractions of training data

We investigated the effect of varying the amount of training data used on testing performance. Unfortunately due to excessive runtime of self-implemented Naive Bayes, the following results are from logistic regression for first dataset only.

For News20Group, as we increase the fraction of training data used (20%, 40%, 60%, 80% to 100%), the training accuracy decreases slightly from 95.6%, 94.1%, 93.3%, 92.3% to 91.5% but the test accuracy increases from 55.3%, 59.3%, 61.1%, 62.4% to 63.1%. This result is expected since only using fractions of data leads to over-fitting on that fraction and not generalizing as well.

For Sentiment140, as we increase the fraction of training data used (20%, 40%, 60%, 80% to 100%), the training accuracy decreases slightly from 68.5%, 68.4%, 68.3%, 68.4% to 68.4% but the test accuracy increases from 69.3%, 69.0%, 69.0%, 69.0% to 69.1%. We do not see large difference here due to the sheer size of the dataset; each 20% random fraction trains the model well enough already.

Discussion and Conclusion

Key takeaways from the project:

- Feature selection is extremely important and min_df parameter in CountVectorizer() has a significant influence on final performance.
- Logistic regression slightly out-performs Naive Bayes due to its weaker assumptions.
- Sentiment140 trains better overall than News20Group due to its larger size and fewer classes.

Possible directions for future investigation:

- Investigate the difference between using counts, tf and tf-idf.
- Fine-tune number of features from CountVectorizer() further to avoid over-fitting.
- Implement logistic regression by hand to have more control over its hyperparameters.
- Compare these two classifiers to other classifiers (neutron networks, etc.).

Statement of Contribution

Atia Islam: data cleaning, data analysis, text preprocessing, write-up

George Qiao: Naives Bayes implementation, write up, formatting

Vraj Patel: logistic regression, write up

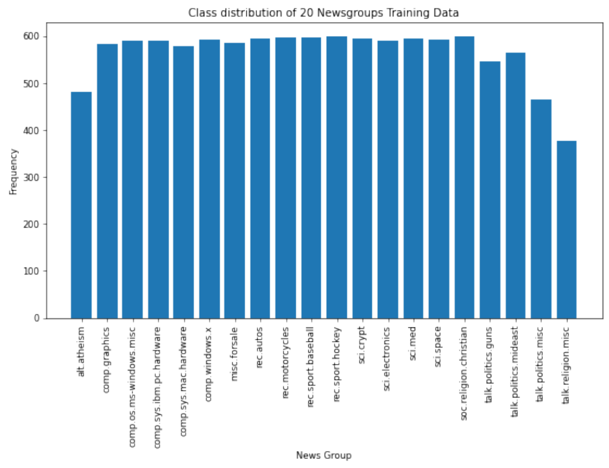
References

[1] K. Annervaz, S. B. R. Chowdhury, and A. Dukkipati, “Learning beyond datasets: Knowledge graph augmented neural networks for natural language processing,” *arXiv preprint arXiv:1802.05930*, 2018.

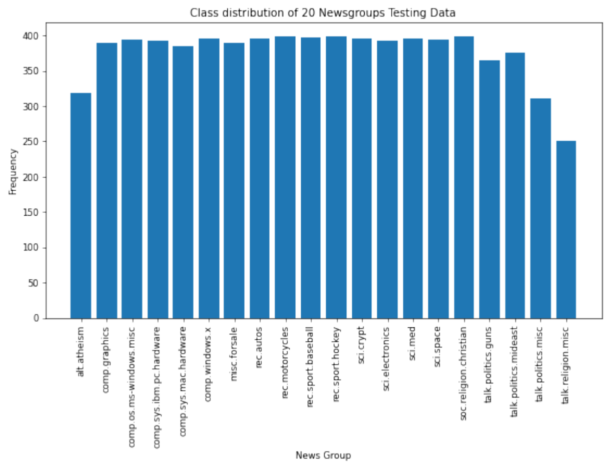
[2] A. Go, R. Bhayani, and L. Huang, “Twitter sentiment classification using distant supervision,” *CS224N project report, Stanford*, vol. 1, no. 12, p. 2009, 2009.

Appendices

Additional Figures

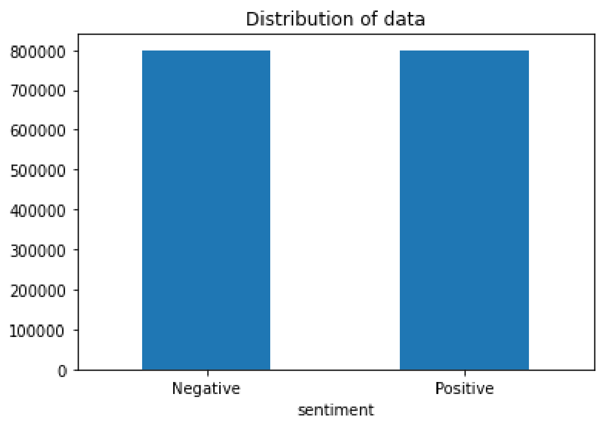


(a) Train

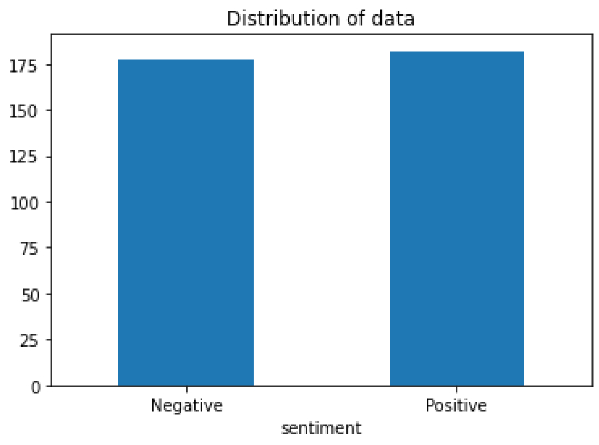


(b) Test

Figure 3: Class Distribution: News20Group



(a) Train



(b) Test

Figure 4: Class Distribution: Sentiment140