Linköping Universitet
Department of Computer and Information Science

Model Comparison for Review Classification

732A92 Text Mining Project

Author
Erik Anders

LiU-ID
erian805

January 17, 2021

Contents

Chapter 1

Abstract

There are a wide variety of classification models available but the hardest part is trying to pick which one to use for the task at hand. Depending on the type of data, hardware resources, amount of data and other factors this can vary a lot. This project is trying to find answer to how available resources and data volume influence this decision between a number of Machine Learning models and a `RoBERTa` Transformer. While clearly outperforming the other models on little data the Transformer couldn't be trained on higher amounts of data. Because of this it was surpassed by not just the moderately time consuming Support Vector Classifier but also the very efficient Multinomial Naive Bayes.

Chapter 2

Introduction

The task of this project is to compare different approaches to classify textual reviews on a scale from one to five. The objective is to find out how different models perform this classification task, what the best configurations of the models are, how their training times compare and how they perform with different volumes of training data. The models compared are some of the time proven supervised machine learning models and a newer transformer model. The introduction of Transformers in "Attention is all you need" [1] in 2017 has led to much recent progress in NLP and in many cases is replacing older RNN models. It is highly suited for sequential data because it doesn't require the data to be processed in order, which makes it's training parallelizable. Transformers are a big topic nowadays and it is therefore of great interest to see how it compares to the other models. I chose this project because the topic of sentiment analysis is very interesting to me and data in the form of reviews offers a great base to train models on since it offers a value responding to the textual review.

Chapter 3

Theory

3.1 Preprocessing

3.1.1 Tokenization

In order to use textual data for predictive modeling, the text must be tokenized. This means to demarcate and possibly classify sections of a string of input characters.

3.1.2 N-grams

Instead of building a simple collection of unigrams (n=1), word n-grams can collect bigrams (n=2), where occurrences of pairs of consecutive words are counted. Because of this they capture phrases and multi-word expressions and account for word order dependencies. Character n-grams can be used to account for potential misspellings or word derivations.

3.1.3 Feature extraction/ Vectorization

Vectorization is the general process of turning a collection of text documents into numerical feature vectors which represents documents by counting how often they include which word. This way machines are able to process them.

3.1.4 Binary Values

Converts the counts of words in documents into binary values.

3.1.5 Term Frequency – Inverse Document Frequency (TF-IDF)

To shift the focus away from very common words and towards the more excluding words it gives those words more weight compared to more frequent ones.

3.2 Classification

3.2.1 Multinomial Naive Bayes

The multinomial Naive Bayes classifier is a simple but effective probabilistic text classifier that builds on Bayes' rule [1] where the adjective Naive says that features in the dataset are mutually independent. Therefore it assumes that the occurrence of one feature doesn't affect the probability of occurrence of the other features, Theoretically the multinomial distribution requires integer feature counts but in practice, fractional counts such as tf-idf work as well.

3.2.2 Bernoulli Naive Bayes

Different to the Multinomial Naive Bayes, the Bernoulli Naive Bayes does not just count the occurrences of a word in the document. It explicitly penalizes the non-occurrence of a word that is an indicator for a class, where the multinomial variant would simply ignore a non-occurring word. It is especially popular for classifying short texts [2]. To do this effectively the model needs a binary input or binarize the input itself. [3].

3.2.3 Support Vector Machines

The Support Vector Machine is a linear model for classification and regression problems. By creating the optimal hyperplane the algorithm separates the data into classes and is able to solve linear and non-linear problems. It uses the Kernel Trick and adds extra dimensions to the data so that it becomes linearly separable and then projects the decision boundary back to original dimensions using mathematical transformation. This hyperplane maximizes the margin, which is the distance of the closest observations to the frontier, called support vectors.

3.2.4 RoBERTa

`RoBERTa` stands for Robustly Optimized BERT Pre-training Approach and was introduced in [4]. `RoBERTa` has a very similar architecture to BERT, which stands for Bidirectional Encoder Representations from Transformers and is a `NLP` model proposed by Google Research in [5]. BERT can be described as an Encoder stack of transformer architecture. A transformer architecture is an encoder-decoder network that uses self-attention on the encoder side and attention on the decoder side. This means that it uses the attention mechanism to assign weight to the parts of the input sequence according to their importance. By doing this the network only focuses on specific points which is especially helpful when trying to learn about the context of words. It has been pretrained on Wikipedia, BooksCorpus and others in a self-supervised fashion but requires task-specific fine-tuning.

3.3 Classification Metrics

3.3.1 Precision

Precision represents the proportion of correctly classified over all the observations labeled in a class. The classes with larger number of observations typically have the worst precision

as the model labels too many observations as those classes.

3.3.2 Recall

Recall represents the proportion of correctly classified documents among all documents from that class. The "large" classes typically have a good recall due to the fact that the model labels many observations in those classes.

3.3.3 F1-Score

The F1-Score is a balanced score between precision and recall. It is good for samples where classes have very different sizes. It combines both precision and recall.

Chapter 4

Data

The selected dataset for the project consists of reviews of fine foods from amazon. The reviews were written between October 1999 and Oct 2012. There are 568,454 reviews in total by 256,059 users on 74,258 products. The data was found and acquired through `Kaggle` [6] but was first collected in [7] and then made available on [8].

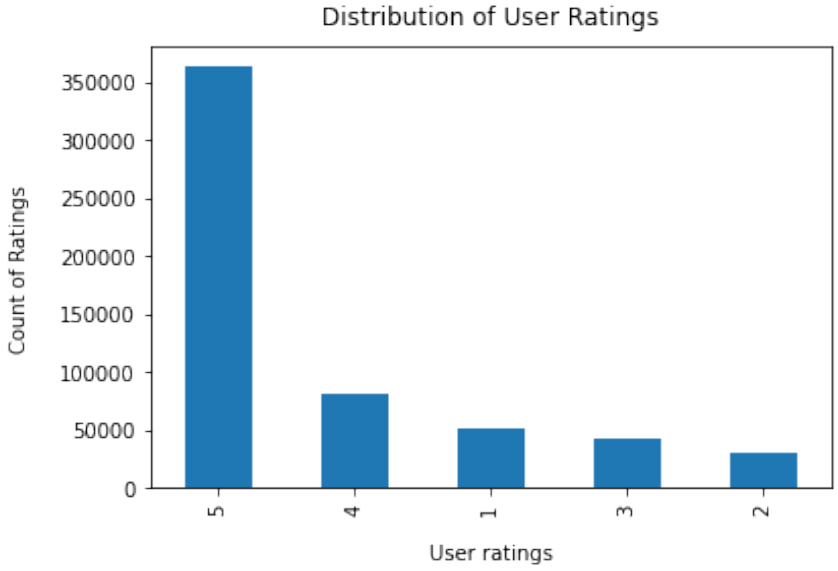

Figure 4.1: Distribution of user ratings

Chapter 5

Method

The study was carried out by first taking a deeper look at the dataset available and the removal of observations, which have Na values.
Beforehand it was decided, that my main objective was to predict user ratings based on their review Text. Taking a look at the distribution 4.1 of the review scores it was apparent, that the five-star review is by far the most common and the other scores are still not uniformly distributed but their occurrences are a lot more similar.

Because of this very uneven distribution a comparisson between the results of a simple `MultinomialNB` with balanced and unbalanced data was made. This means that instead of just training on randomly selected observations they were sampled by their Score. Every Score only got sampled as many times as the rarest Score available to have a uniform distribution between the ratings. To guarantee comparison the preprocessing with $CountVectorizer(binary = True, ngram_range = (2, 2))$and the overall amount of training data were the same in both cases.

Because of overall far better accuracy on unbalanced data all training on the following models was done on the original, unbalanced data.

The models chosen for comparison are Multinomial Naive Bayes, Bernoulli Naive Bayes, Support Vector Classification and the pretrained Transformer `RoBERTa`. For the first three models the implementations by `scikit-learn` [9], namely `MultinomialNB`, `BernoulliNB` and `svm.LinearSVC` were used. An initial attempt to use `svm.SVC` failed because of limited ressources and a switch to the more efficient `svm.LinearSVC` had to be made. For `RoBERTa` the implementation by `SimpleTransformers` [10] was used. Unlike the other models it was run on a GPU.

While trying to find the best preprocessing and model parameters for the `scikit-learn` models the limits of my environment were quickly reached. Therefore instead of running the hyperparameter search on all data (511584 observations) it was run on just 25000 observations. The transformer was run with default settings, since even small training and prediction runs were pushing the capability of my environment, a hyperparameter search didn't seem feasable. `GridSearchCV` returned following best parameter set:

Preprocessor	Tf-idf Transformer	Model
CountVectorizer(binary=True,ngram_range=(2,2))		MultinomialNB(alpha=1)
CountVectorizer(ngram_range= (1, 2))	TfidfTransformer(use_idf=False)	MultinomialNB(alpha=0.1)
CountVectorizer(ngram_range=(1, 2), binary=True)		BernoulliNB()
CountVectorizer(ngram_range= (1, 2), binary=True)	TfidfTransformer(use_idf=True)	svm.LinearSVC()

Table 5.1: Best parameters according to `GridSearchCV`

The hyperparameters of the first model in the table actually weren't found by `GridSearchCV` but instead randomly found in the environment from earlier experiments. Because the its classification results were far better than then ones of `MultinomialNB` with the 'optimized' hyperparameter they were included in the project.

For the comparison between the models, including their individual preprocessing they were trained on different amounts of data, while always being tested on 10% of the total volume. To cover stronger suspected changes with the smaller training volumes the volume was increased exponential, starting at 0.001% of the total volume, multiplying that percentage by five for every step possible. After reaching the last step of 62.5% a final step at 90% was done to reach the models highest potential.

Percentage of data	0.001	0.005	0.025	0.125	0.625	0.9
Number of Observations	568	2842	14210	71053	355266	511584

Table 5.2: Steps of Training Data Volume used for Comparison

Another aspect of the comparison between models was measuring how long it took them to train on the above mentioned data volumes. To do so a Timer class as done here [11] was created. This class then was started before each training and stopped right afterwards. Stopping the class resulted in an output of the passed time in seconds. Since `RoBERTa` doesn't require any prior preprocessing the preprocessing time of the other models was also included in their training times. This means that entire time it took to fit the corresponding pipe was measured.

For the models supported by `scikit-learn` the Text and Score columns were used as training and testing inputs. For the `RoBERTa` the data had to be adjusted by renaming the columns into text and labels and subtracting 1 off of every Score so the scoring starts at zero.

All the project code is visible in this GitHub repository [12]

Chapter 6

Results

6.1 Balanced vs. Unbalanced Data

	precision	recall	f1-score	support
1	0.25	0.72	0.37	5360
2	0.32	0.48	0.38	2996
3	0.38	0.51	0.44	4194
4	0.34	0.44	0.39	7894
5	0.90	0.52	0.66	36399
accuracy			0.53	56843
macro avg	0.44	0.53	0.45	56843
weighted avg	0.69	0.53	0.56	56843

Table 6.1: Balanced data with `MultinomialNB`

	precision	recall	f1-score	support
1	0.71	0.40	0.51	5360
2	0.67	0.14	0.23	2996
3	0.57	0.28	0.38	4194
4	0.52	0.21	0.30	7894
5	0.74	0.98	0.84	36399
accuracy			0.72	56843
macro avg	0.64	0.40	0.45	56843
weighted avg	0.69	0.72	0.67	56843

Table 6.2: Unbalanced data with `MultinomialNB`

Comparing the outputs of scikit-learn's classification report it is apparent that the model trained on a balanced dataset is better at predicting class 2-4 but get outperformed by the model trained on an unbalanced dataset when predicting class 1 and 5.

6.2 Accuracy Depending on Training Volume

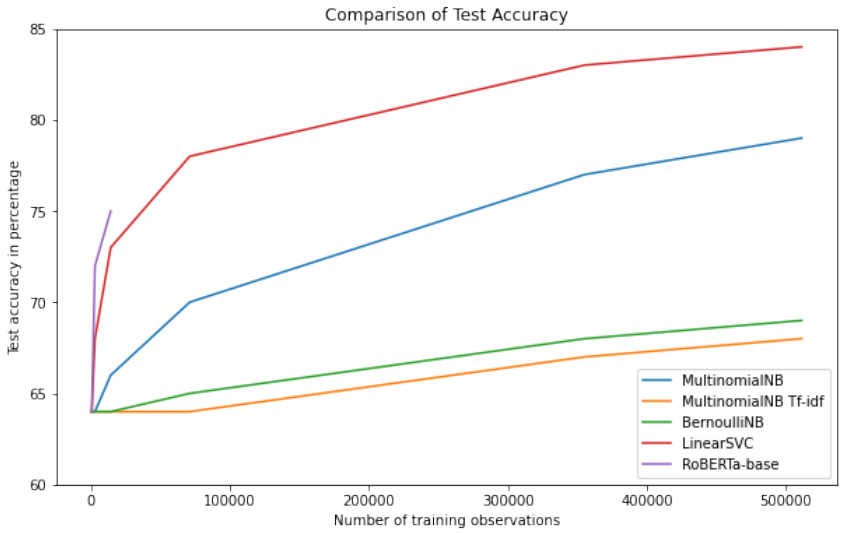

Figure 6.1: Test accuracies for different volumes of training data

`RoBERTa` had a very good accuracy from the beginning, with `LinearSVC` not far behind and eventually overtaking. `MultinomialNB without Tf-idf` could not keep up with that but still outperformed `MultinomialNB with Tf-idf` and `BernoulliNB`.

6.3 Training Duration depending on Training Volume

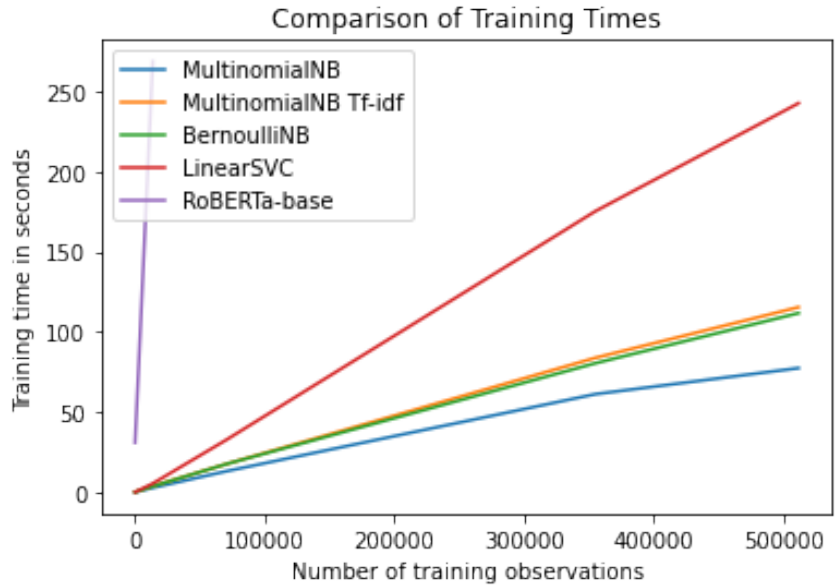

Figure 6.2: Training duration for different volumes of training data

`MultinomialNB without Tf-idf` was the quickest model to train across the board, with `MultinomialNB with Tf-idf` and `BernoulliNB` closely behind. `LinearSVC` took about twice as long and `RoBERTa` had extremely high training times compared to the others.

6.4 Model evaluation with for different volumes of training data

F1-Scores	568	2842	14210	71053	355266	511584
1	0.06	0.05	0.17	0.43	0.69	0.73
2	0.00	0.02	0.05	0.15	0.37	0.42
3	0.01	0.02	0.08	0.20	0.42	0.48
4	0.03	0.03	0.08	0.22	0.45	0.50
5	0.78	0.78	0.79	0.81	0.86	0.88

Table 6.3: F1-Scores of `MultinomialNB` for different volumes of training data

F1-Scores	568	2842	14210	71053	355266	511584
1	0.00	0.00	0.00	0.05	0.33	0.42
2	0.00	0.00	0.00	0.01	0.03	0.06
3	0.00	0.00	0.00	0.00	0.07	0.11
4	0.00	0.00	0.00	0.01	0.12	0.17
5	0.78	0.78	0.78	0.78	0.80	0.80

Table 6.4: F1-Scores of `MultinomialNB with Tf-idf` for different volumes of training data

F1-Scores	568	2842	14210	71053	355266	511584
1	0.00	0.01	0.05	0.15	0.41	0.46
2	0.00	0.00	0.01	0.03	0.14	0.18
3	0.00	0.00	0.01	0.06	0.22	0.27
4	0.00	0.02	0.08	0.19	0.33	0.36
5	0.78	0.78	0.78	0.79	0.80	0.81

Table 6.5: F1-Scores of `BernoulliNB` for different volumes of training data

F1-Scores	568	2842	14210	71053	355266	511584
1	0.07	0.39	0.61	0.72	0.79	0.81
2	0.00	0.05	0.18	0.38	0.56	0.60
3	0.01	0.11	0.30	0.45	0.61	0.63
4	0.02	0.16	0.31	0.45	0.61	0.64
5	0.78	0.82	0.86	0.89	0.92	0.93

Table 6.6: F1-Scores of `LinearSVC` for different volumes of training data

F1-Scores	568	2842	14210	71053	355266	511584
1	0.00	0.62	0.69			
2	0.00	0.00	0.22			
3	0.00	0.29	0.45			
4	0.00	0.06	0.32			
5	0.78	0.87	0.89			

Table 6.7: F1-Scores of `RoBERTa` for different volumes of training data

Looking at the tables it becomes apparent that at lower observation numbers, almost all correctly classified reviews belong to class 5, which is also the most common one. With higher training observations this the models still classify class 5 the best, but also manage to improve classification of the other classes. Especially `RoBERTa` and `LinearSVC` manage to do this already at a considerably low training volume. All the models improve at every step of more training data.

At full training data `MultinomialNB with Tf-idf` and `BernoulliNB` have the worst results, still relying heavily on class 5 and 1 and doing a bad job at predicting the other classes.

`MultinomialNB without Tf-idf` gave significantly better results with decent F1-scores over class 2-4 and good ones for class 5 and 1.

`LinearSVC` got really good at differentiating the classes with good scores across all classes, especially class 5 and 1.

`RoBERTa` showed great F1-scores at lower training volumes, outperforming all the other models but couldn't be computed for higher training volumes. Its best result showed a good classification of class 5 and 1 and decent scores on the other classes. Out of those other classes `RoBERTa` was the only model especially good at predicting class 3.

Chapter 7

Discussion

Looking at the results it is fair to assume that when the necessary resources are available `RoBERTa` gives the best classification results. Especially on lower training volume it can come in handy as its long training time is not that limiting yet and it is able to provide more insights than any of the other models. This early success can be contributed to the fact that it already is pretrained.
When resources are sparse however a `MultinomialNB without Tf-idf` with the right hyperparameters and preprocessing is the best choice. It is impressive how quick it computes, outperforming all other models in that aspect while still delivering respectable results. When it comes to speed or capacity it is definitely the best of the chosen models. For someone looking for the best of both worlds `LinearSVC` is the clear winner out of this comparison.For `LinearSVC` to outperform `MultinomialNB` in similar tasks is completely normal and can also be seen in[13]. But for some tasks like in [14] `MultinomialNB` can actually reach higher performance. While taking more than twice as long as the `Naive Bayes` based models the `LinearSVC` can still be considered as fast.

As mentioned in the results `MultinomialNB with Tf-idf` and `BernoulliNB` could not compare with the other models when it came to classification results but didn't need a lot of time to train. Both of them however got outperformed by `MultinomialNB without Tf-idf` in every aspect.

As seen in the fact that `MultinomialNB without Tf-idf` outperformed `MultinomialNB with Tf-idf` even though this setting was superior according to earlier hyperparameter optimization it is obvious that this might not be the best results possible with these models. Because all computations were run on the free version of Google Colab[15] `GridSearchCV` were executed with a very limited amount of data for it to actually deliver results. Another decision which might've had a strong negative impact on the results was that these optimizations were carried out on balanced data.

One thing that has to be mentioned is that the training times were computed including preprocessing, which is also the reason for the gap in training time between `MultinomialNB without Tf-idf` and `MultinomialNB without Tf-idf`.

Unfortunately `RoBERTa` did not manage to train from the fourth training step onward. It's training time increased at such a rate that with training on five times the amount of data of the previous step the environment could not handle it.

Chapter 8

Conclusion

As seen in this project, even if there is a big hype around a new technology it does not mean that it is the right choice for everybody. `RoBERTa` gave better results than any other model for the steps it was able to compute but even the Google Colab GPU could not process higher training volumes. This led to it being outperformed by even the `MultinomialNB` which only takes a fraction of `RoBERTa`s time to train.
`RoBERTa` performed very good and promises a lot more given more resources which would be a good starting point for future work. The same goes for doing more extensive searches for hyperparameters on bigger datasets, that give a better representation of the full data. `LinearSVC` was the clear overall winner when it comes to delivering good results in reasonable time domains. Achieving 84% accuracy on a classification problem with five classes is no easy feat and it is questionable how much room for improvement there is. Reviews are written by humans and for us often misleading, with many of them seeming to have unreasonable scores for the feedback provided.

Bibliography

[1] *732A92 Slides Lecture 2* (page 4).

[2] Andrew McCallum and Kamal Nigam. "A Comparison of Event Models for Naive Bayes Text Classification". In: *Learning for Text Categorization: Papers from the 1998 AAAI Workshop*. 1998, pp. 41–48. URL: http://www.kamalnigam.com/papers/multinomial-aaaiws98.pdf (page 4).

[3] *Scikit-learn BernoulliNB*. URL: https://scikit-learn.org/stable/modules/naive_bayes.html#bernoulli-naive-bayes (page 4).

[4] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL] (page 4).

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL] (page 4).

[6] *Amazon Fine Food Reviews*. 2019. URL: https://www.kaggle.com/snap/amazon-fine-food-reviews (page 6).

[7] J. McAuley and J. Leskovec. *From amateurs to connoisseurs: modeling the evolution of user= expertise through online reviews*. 2013. URL: http://i.stanford.edu/~julian/pdfs/www13.pdf (page 6).

[8] *Stanford Large Network Dataset Collection*. URL: http://snap.stanford.edu/data/ (page 6).

[9] *Scikit-learn*. URL: https://scikit-learn.org/stable/index.html (page 7).

[10] *SimpleTransformers*. URL: https://simpletransformers.ai/ (page 7).

[11] *Python Timer Functions: Three Ways to Monitor Your Code*. URL: https://realpython.com/python-timer/ (page 8).

[12] *GitHub Code Repository*. URL: https://github.com/andd3r5/Text_Mining_Project (page 8).

[13] R Kusumawati, A D'arofah, and P A Pramana. "Comparison Performance of Naive Bayes Classifier and Support Vector Machine Algorithm for Twitter's Classification of Tokopedia Services". In: *Journal of Physics: Conference Series* 1320 (Oct. 2019), p. 012016. DOI: 10.1088/1742-6596/1320/1/012016. URL: https://doi.org/10.1088/1742-6596/1320/1/012016 (page 13).

[14] Stan Matwin and Vera Sazonova. "Direct comparison between support vector machine and multinomial naive Bayes algorithms for medical abstract classification". In: *Journal of the American Medical Informatics Association* 19.5 (June 2012), pp. 917–917 (page 13).

[15] *Google Colab*. URL: https://colab.research.google.com/ (page 13).