Natural language processing project: Sentiment analysis

Simon Frank (6095707)

Submission on July 13, 2023
SS 23

Dr. Ali Bahrainian
Health NLP Lab
University Tübingen

Abstract

Sentiment analysis plays a crucial role in understanding public opinion and sentiment expressed in textual data. In this project, the goal was to classify movie reviews as positive or negative using different models and feature sets. The dataset used was the Rotten Tomatoes dataset, which contains labelled movie reviews. The Naive Bayes models performed well, with an accuracy of about 76%. SVM models performed slightly worse than Naive Bayes, while LSTM models, especially with word2vec embeddings, showed improved accuracy, reaching around 77%. The BERT model, fine-tuned to the dataset, emerged as the best performing model with an accuracy of 80.30%.

Contents

List of Figures

List of Tables

1 Introduction

Natural Language Processing (NLP) is a rapidly developing field of study that focuses on the interaction between computers and human language. It encompasses a wide range of techniques and algorithms designed to enable computers to understand, interpret and generate meaningful human language. At its core, NLP is about bridging the gap between human language and computational systems. It involves the development of algorithms and models that can process, analyse and derive meaningful insights from large amounts of textual data. A prominent area within NLP is sentiment analysis, which aims to understand and interpret the subjective information expressed in text, such as opinions, emotions and sentiments. Sentiment analysis is the process of computationally identifying and categorising the sentiment expressed in a piece of text. The primary goal of sentiment analysis is to classify text into different sentiment categories, such as positive, negative or neutral. In this research project, sentiment analysis is performed on the rotten tomatoes dataset. It contains film reviews that are marked as positive or negative. First, the dataset is preprocessed and analysed. Then two basic models, Naive Bayes and Support Vector Machine, are used for this task. Also, a long short-term memory network with two different feature embeddings was trained from scratch. Finally, a pre-trained language model (BERT) was fine-tuned. The performance of all the different models was compared using appropriate metrics.

2 Dataset

The dataset used for sentiment analysis throughout this project is the 'Rotten Tomatoes' dataset provided by hugging face [9]. It is a dataset of movie reviews in English with two possible classes for each review, positive and negative. In total it contains 5331 positive and 5331 negative reviews. These are divided into a training set with 8350 reviews, a validation set with 1066 reviews and a test set with 1066 reviews. All three data splits contain the same number of positive and negative labels, so they are balanced. Throughout the project, the training data split was used to train the model and the test data split was used to evaluate the performance of the trained model.

2.1 Preprocessing

The aim of the pre-processing is to simplify the data set and to remove any information that is not helpful and may even hinder the subsequent task, the sentinel analysis. Therefore, all stop words and punctuation are removed and all words are converted to lower case and lemmatised. This was done using NLTK: The Natural Language Toolkit [4].

2.2 Analysis

The token, unigram and bigram counts for the three different datasplits are shown in 1.

	Train	Validation	Test
Token count	90729	11358	11462
Unigram count	16413	4979	4989
Bigram count	72582	9932	10003

Table 1: Token and n-gram counts for the three different data splits

In the training dataset, the number of bigrams is 4 times higher than the number of unigrams. So there is much more information when bigrams are used as input to a model. It needs to be investigated whether this is useful or not.

Figure 1 shows the histogramm of the sequence length of the training dataset. The mean sequence length is 10.6. The longest review has 32 tokens. This finding is of particular interest when using the LSTM approach, as in these models all sequences are usually truncated to a fixed length in order to achieve better results.

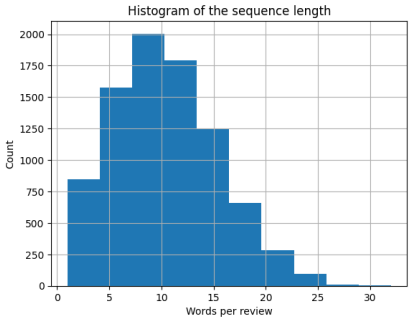


Figure 1: Histogramm of the sequence length of the training reviews

To analyse the different topics covered in the dataset, Latent Dirichlet Allocation (LDA) was performed on the training dataset with a topic number of ten. The top four keywords with their corresponding probabilities are shown in table 4. For 9 topics 'movie' or 'film' is the word with the highest or second highest probability. For 4 topics both words are the first words. This could be expected for a dataset of film reviews, but it also shows that there is not much variance between the topics. It can also be concluded that there are three topics (topic 2,3,4) with high probability words that can be interpreted as negative film attributes, e.g. 'boring' and 'bad'.

Topic	Top Keywords (Probability)			
	Word 1	Word 2	Word 3	Word 4
1	like (0.022)	movie (0.022)	film (0.015)	comedy (0.010)
2	movie (0.011)	like (0.011)	boring (0.011)	character (0.011)
3	start (0.012)	episode (0.010)	dumb (0.008)	movie (0.008)
4	movie (0.028)	bad (0.018)	one (0.013)	like (0.012)
5	film (0.012)	movie (0.010)	obvious (0.007)	drag (0.006)
6	movie (0.015)	film (0.012)	like (0.009)	exercise (0.008)
7	movie (0.026)	film (0.017)	one (0.011)	isnt (0.009)
8	movie (0.017)	dull (0.013)	film (0.011)	one (0.011)
9	film (0.022)	two (0.008)	hard (0.008)	may (0.008)
10	film (0.033)	movie (0.011)	much (0.007)	life (0.007)

Table 2: Top keywords and probabilities for each topic using LDA

To calculate the coherence score for the top 20 words for each topic with different numbers of topics, the Normalised Point-wise Mutual Information was used and is shown in figure 2. Since the best scores are obtained with

only 1 or 2 topics, this underlines the above findings that there are not many different topics in this dataset.

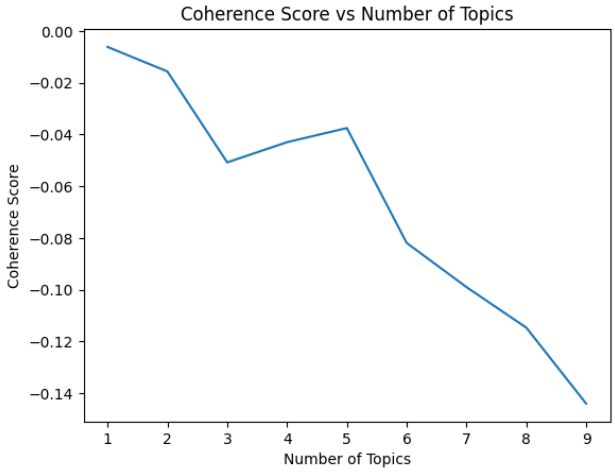


Figure 2: Coherence score for different number of topics

2.3 Metrics

To evaluate the performance of the different models, accuracy, precision, recall and F1 score are used. Accuracy calculates the number of correctly predicted instances out of the total number of instances. Although this is a commonly used metric, it may not give a complete picture, so other metrics were also used. Precision is the proportion of true positive predictions out of all positive predictions made by the model. Recall measures the proportion of true positive predictions out of all actual positive instances. The F1 score is the harmonic mean of precision and recall and provides a balanced score. All of these metrics were used to provide a comprehensive assessment of the model's effectiveness.

3 Models

In the following five different model and feature seetings were trained and tested on the rotten tomatoes dataset.

3.1 Naive Bayes

Naive Bayes methods are a set of supervised learning algorithms; as they are a really fast method and can still achieve good results in text classification tasks, they mark a good start for this project.
Naive Bayes methods apply Bayes' theorem with the "naive" assumption of conditional independence between each pair of features, given the value of the class variable. Bayes' theorem states the following relationship, given the class variable y and the dependent feature vector x_1 through x_n:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Using the naive conditional independence assumption this relationship is then simplified to:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

The following classification rule can then be used:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

The $P(y)$ is the relative frequency of class y in the training set. The different naive Bayes classifiers differ in their assumptions about $P(x_i \mid y)$ [10].
There are two different naive Bayes algorithms used in text classification such as sentiment analysis. In Multinomial Naive Bayes (MNB), the distribution is parameterised by vectors $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ for each class y, where n is the number of features in text classification, the number of unigrams or bigrams. θ_{yi} is the probability $P(x_i \mid y)$ that feature i appears in a sample of class y. The parameters are then estimated:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

N_y is the total number of features for class y and N_{yi} is the number of times feature i appears in a sample of class y. α is a smoothing prior and prevents

zero probabilities when features are not present in the learning sample [5]. The second algorithm is the Complement Naive Bayes (CNB) algorithm, which is an adaptation of the standard multinomial naive Bayes algorithm. It uses statistics from the complement of each class to compute the model weights, often outperforming MNB. The weights are computed as follows

$$\hat{\theta}_{ci} = \frac{\alpha_i + \sum_{j:y_j \neq c} d_{ij}}{\alpha + \sum_{j:y_j \neq c} \sum_k d_{kj}}$$

$$w_{ci} = \log \hat{\theta}_{ci}$$

$$[?w_{ci} = \frac{w_{ci}}{\sum_j |w_{cj}|}$$

The sums are over all documents j not in class c and d_{ij} is the number of terms i in document j. α_i is a smoothing hyperparameter and the second normalisation prevents longer documents from dominating the parameter estimation. The classification rule assigns the document to the class that is the worst complement match:

$$\hat{c} = \arg\min_c \sum_i t_i w_{ci}$$

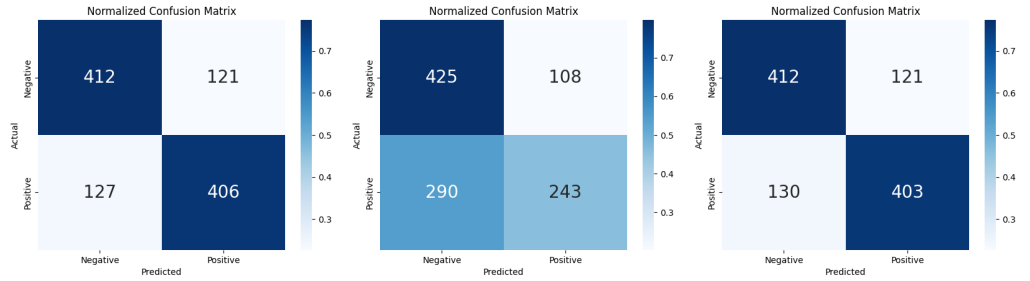
[6] [8]. For both classifiers, three different feature vectors are used, unigrams, bigrams and both, to investigate the influence of different input feature sets on the performance of a simple model, since it was found in chapter 2.2 that the number of bigrams is much higher than the number of unigrams.

Input features	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Unigram	76.74	77.04	76.17	76.60
Bigram	62.66	69.23	45.59	54.98
Unigram & bigram	76.45	76.91	75.61	76.25

Table 3: Performance metrics of the multinomial naive Bayes model for different input features

Table 3 shows the metrics of the trained NBM of the test set. The first and third input feature sets both achieve relatively high accuracy and F1 scores, so they are good classifiers. When only the bigrams are used as input features, the model has a low recall, indicating a lower ability to capture all positive instances. So the bigrams are not a good set of features.

Figure 3 illustrates the metrics shown in table 3. With the bigram as an input feature, many positive reviews are predicted to be negative. Between



(a) Unigram (b) Bigram (c) Unigram & bigram

Figure 3: Confusion matrices of the multinomial naive Bayes model for different input features

the other two input features, the false positive and negative rates are quite similar and really good for such a simple model. When the CNB model is used instead of the MNB, there is no change in the performance metrics, which is due to the fact that the training dataset is balanced. Thus, depending on the data, CNB does not always outperform MNB.
It could be said that with the right input features, even a very simple model can achieve good results on this dataset.

3.2 Support vector machine

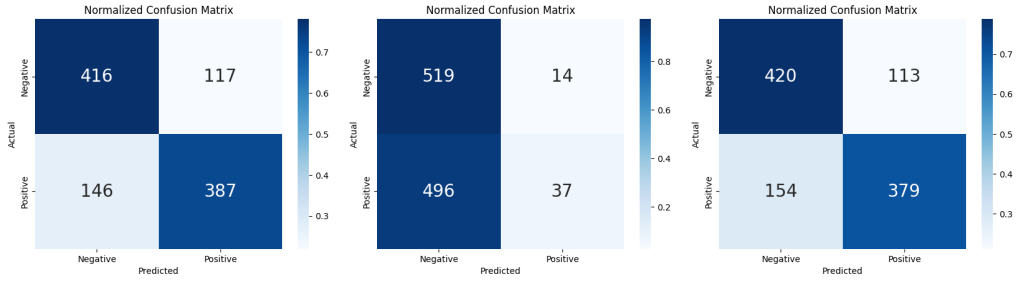
A support vector machine (SVM) constructs a hyperplane or set of hyperplanes in a high-dimensional space. It can be used for classification tasks such as sentiment analysis. Good separation can be achieved if the hyperplane has a large distance to the nearest training data points of each class. Since there are two possible output classes for sentiment analysis, the C-Support Vector Machine (SVC) is used. The goal of the SVC is to find $w \in \mathbb{R}^p$ and $b \in \mathbb{R}$ such that the prediction given by $sign(w^T\phi(x) + b)$ is correct for most samples given training vectors $x_i \in \mathbb{R}^p$, $i = 1, ..., n$, in two classes, and a vector $y \in \{1, -1\}^n$ [7].
The same input features were used for the SVM as for the naive Bayes model, for the same reasons.

As for the Bayesian model, the unigrams slightly outperform the combination of unigrams and bigrams as input features. Using only the bigrams as input features the SVM has an accuracy just slightly above chance and a really low recall (table 4).

The confusion matrices for the different settings are shown in figure 4. Compared to the naive Bayes model, the support vector machine more often predicts the review as negative when it was positive. For the Bigram model

Input features	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Unigram	75.33	76.79	72.61	74.64
Bigram	52.16	72.55	6.94	12.67
Unigram & bigram	74.95	77.03	71.11	73.95

Table 4: Performance metrics of the support vector machine for different input features



(a) Unigram (b) Bigram (c) Unigram & bigram

Figure 4: Confusion matrices of the support vector machine for different input features

this tendency is extremely strong. It almost always predicts the review to be negative, so the model is not able to construct a good hyperplane in the bigram feature space. As similar problems occurred for the naive Bayes model, it can be said that the bigrams are not good input features. This could be due to the fact that the bigrams are much harder to categorise as good or bad than the unigrams. However, with the correct unigrams as input features, both simple models are able to achieve good performance.

3.3 LSTM

The first neural network-based model in this project is based on a Long Short-Term Memory (LSTM) network. This is a type of recurrent neural network that is particularly effective for processing and analysing sequential data such as text. The LSTM overcomes the vanishing gradient problem that occurs when training traditional RNNs through memory cells and gating mechanisms. The LSTM cell has five core components: Cell State (holds the information over time), Input Gate (determines the amount of new information stored), Forget Gate (controls the extent to which previous memory is forgotten or retained), Output Gate (controls the amount of memory to be

output) and the Hidden States (output that carries information to the next time step) [3]

3.3.1 Feature embedding

The LSTM approach used two different feature embeddings as input to the network. Since a neural network cannot work with a word as input, each word must be transformed into a unique identifier. The first embedding is a simple mapping from a word to a unique integer via a constructed look-up table of all the words in the training data set. The second embedding uses word2vec. Word2Vec is an algorithm for generating word embeddings that capture semantic and syntactic relationships between words.This setup can be used to test whether an LSTM network benefits from already meaningful input features, or whether the model is able to learn with just a little bit of data provided. In a first test, all sentences in the training dataset were used to learn these embeddings. Using these embeddings, it was not possible to achieve good results in sentiment analysis using an LSTM. This must be due to the small corpus size of the data, which leads to no meaningful embeddings. Looking at the most similar words to the word 'film' in the embedding space, the results are as follows 'film' 99.9%,'one' 99.9%, 'character' 99.9%,'performance' 99.9% and illustrate the proposed hypothesis. For this reason, an existing word2vec model/embedding was used for the rest of the project.
To facilitate training, all sequences are truncated to a fixed size. If the sequence is shorter, the missing tokens are filled with zeros. The length of the sequence is a compromise. It should not be too short in order to remove information from a lot of reviews, but if it is too long, a lot of sequences will have to be filled with zeros. Looking at the histogram of the sequence length 20 marks a good compromise.

3.3.2 Network architecture

The network architecture for the LSTM approach is quite simple in order to keep training feasible without any dedicated computing power. If the feature embeddings are only integer, the first component is a learnable feature embedding layer, otherwise the input is given directly to the LSTM layer. The LSTM layer consists of two layers with a hidden dimension of 256, followed by a linear layer mapping from the 256 dimension to an output dimension. Two different approaches have been tested during the project. The first uses only the last output of the LSTM layer, the second uses the average of all outputs. The sigmoid function is used as the output activation function to

obtain the class probability. To prevent overfitting, a drop-out is applied after the LSTM layer.

3.3.3 Training and results

The Adam optimiser with a learning rate of $1x10^{-3}$ and a batch size of 128 was used for training. The network was trained on the test set. Training was stopped when the training loss decreased but the validation loss did not decrease. For the interger embedding this happened after 3 epochs, for the word2vec embedding after 7 epochs.

Used output	Accuracy	Precision	Recall	F1-score
Last	69.42	72.55	62.48	67.14
Average	70.73	71.54	68.86	70.17

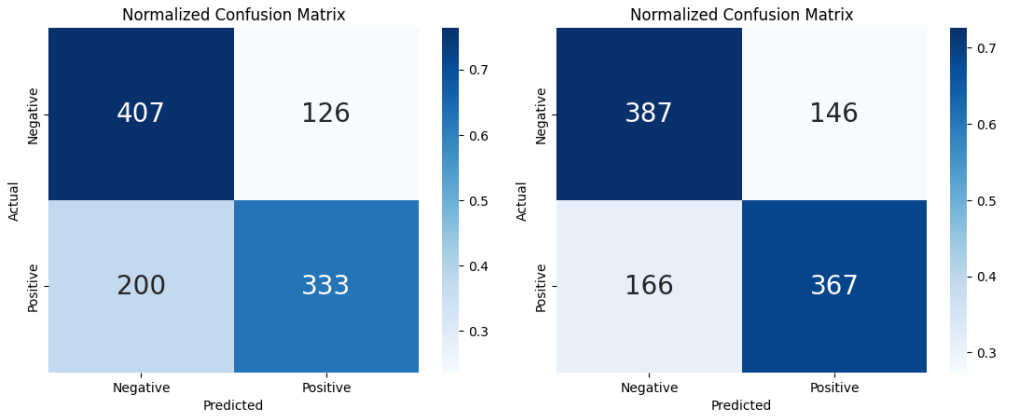
Table 5: Performance metrics of the LSTM network with the interger feature embedding using the last or all outputs of the LSTM cell for the class predicitons

For both models the performance is good (table 5). When only the last output of the LSTM cell is used for classification, the recall is slightly lower than when all outputs are used. This is also shown in the confusion matrix (fig 5). There are 34 more false negatives when only the last output is used. This could be due to the fact that the last words of the sequence would implement a negative review, although the overall review is positive. The model averaging over all outputs has a better F1 score and accuracy and is therefore the preferred model.

Used output	Accuracy	Precision	Recall	F1-score
Last	77.02	75.35	80.30	77.75
Average	77.02	80.25	71.67	75.72

Table 6: Performance metrics of the LSTM network with the word2vec feature embedding using the last or all outputs of the LSTM cell for the class predicitons

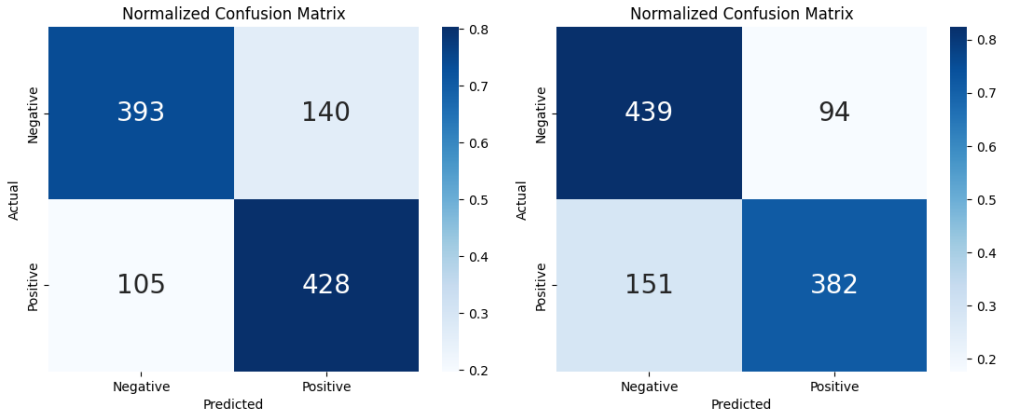
Comparing the performance of the word2vec feature embedding (Table 6) with the simple integer feature embedding, it can be concluded that both word2vec models perform better. Since the input features using word2vec



(a) Last (b) All

Figure 5: Confusion matricies of the LSTM network with the interger feature embedding using the last or all outputs of the LSTM cell for the class predicitons

already have semantic meaning, the downstream task of sentiment classification becomes easier for the network to learn. Depending on whether only the last or all hidden states were used, the precision or recall is better or worse, as shown in figure 6.



(a) Last (b) All

Figure 6: Confusion matricies of the LSTM network with the word2vec feature embedding using the last or all outputs of the LSTM cell for the class predicitons

It can be concluded that the LSTM approach is also able to achieve good performance on this dataset. However, it needs already meaningful input

features, e.g. produced by word2vec, to slightly outperform the naive Bayes model.

3.4 BERT

The transformer-based model used is BERT, which stands for Bidirectional Encoder Representations from Transformers. It is a language representation model. BERT is designed to learn bidirectional representations from a text by jointly conditioning on the left and right contexts in all layers. This approach allows it to be easily fine-tuned with just one additional output layer using the learned representation of the input text, without major task-specific architectural changes. These characteristics make BERT a perfect fit for the use of a complex model in the project, given the limited computational power provided [1].
The project used the pre-trained BERT base model with 110 million parameters provided by Hugging Face. It was pre-trained with two objective functions. Masked Language Modelling (MLM) where 15% of the words in the input were masked and it has to predict the masked words. Next sentence predition (NSP) were two sentences are concatenated and the model was to predict whether the two sentences follow each other or not [2].
Since BERT also only works with tokenised words, the whole dataset has to be tokenised first. Therefore the provided BERT tokeniser was applied to the whole dataset. The model was then fine-tuned on the training data set for 4 epochs.

Model	Accuracy	Precision	Recall	F1-score
BERT	80.30	78.89	82.74	80.77

Table 7: Performance metrics for the pretrained BERT model fine-tuned on the training dataset

BERT performs very well with an accuracy and F1 score of over 80% (table 7). This good performance is also shown in the confusion matrix (figure 7) with only 92 false negatives.

The BERT model achieves the best results in all metrics of all the models implemented and tested in the project, but it is also the most complex model, which has the most parameters. It was also already pre-trained on a lot of data.

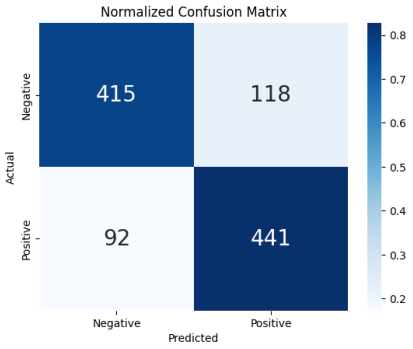


Figure 7: Confusion matrix of the pretrained BERT model fine-tuned on the training dataset

3.5 Comparison

The best models in each model category are compared below (table 8).

Model	Accuracy	Precision	Recall	F1-score
Naive Bayes	76.74	77.04	76.17	76.60
SVM	75.33	76.79	72.61	74.64
LSTM	70.73	71.54	68.86	70.17
LSTM word2vec	77.02	75.35	80.30	77.75
BERT	80.30	78.89	82.74	80.77

Table 8: Comparison of the performance metrics between the different models

As noted above, BERT is the best model across all metrics. All three neural network approaches require computing power for training and also have a longer training and inference time compared to the SVM and the naive Bayes model. As the training and inference times for all three neural network approaches are about the same, the BERT model should be preferred. In general, if performance is the most important aspect of the model, the BERT model should be used, even if it has a longer training and perturbation time. The Naive Bayes model is the best non-neural network model. It has almost no training and perturbation time. Since the performance is only slightly worse than the naive Bayes model, it is a really good alternative. This shows that sometimes even the basic models and algorithms can perform well compared to the big neural network architectures, especially when the dataset used and the task is quite simple like in this project.

4 Conclusion and future work

In this project, various models were applied to the Rotten Tomatoes dataset to classify film reviews as positive or negative. The models included Naive Bayes, Support Vector Machine (SVM), Long Short-Term Memory (LSTM) networks and BERT. The first step was to preprocess the dataset. The Naive Bayes models performed well, achieving an accuracy of around 76% when using either unigrams or a combination of unigram and bigram features. The SVM models performed slightly worse than Naive Bayes, with an accuracy of around 75% using the same input features. The LSTM models, using both integer and word2vec embeddings, achieved accuracies of around 70% and 77% respectively, with the word2vec embeddings outperforming the simple integer embedding. The BERT model, a transformer-based language representation model, gave the best results of all models with an accuracy of 80.30%. BERT emerged as the best performing model, outperforming all other models in terms of accuracy, precision, recall and F1 score. However, BERT required more computational resources and longer training and inference times compared to the other models; if simplicity and faster inference times are more important, the Naive Bayes model may be a viable alternative. For future work, different transformer-based models can be tested. In addition, more extensive hyperparameter search and the use of techniques such as ensembling or model stacking could lead to even better performance. Overall, sentiment analysis remains a dynamic field with significant room for further exploration and improvement, offering valuable insights into understanding and interpreting subjective information expressed in textual data.

5 References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Hugging face. Bert base model. `https://huggingface.co/bert-base-uncased`, 2023. Accessed on July 5, 2023.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[4] Edward Loper and Steven Bird. Nltk: The natural language toolkit, 2002.

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

[6] J Rennie, L Shih, J Teevan, and K Tackling. The poor assumptions of naive bayes classifiers. In *Proceedings of the twentieth international conference on machine learning (ICML), Washington, DC*, 2003.

[7] scikit-learn developers. 1.4. support vector machines. `https://scikit-learn.org/stable/modules/svm.html`, 2023. Accessed on July 3, 2023.

[8] scikit-learn developers. 1.9. naive bayes. `https://scikit-learn.org/stable/modules/naive_bayes.html`, 2023. Accessed on July 3, 2023.

[9] Rotten Tomatoes. Rotten tomatoes dataset. `https://huggingface.co/datasets/rotten_tomatoes`, 2023. Accessed on July 2, 2023.

[10] Harry Zhang. The optimality of naive bayes. *Aa*, 1(2):3, 2004.