Beyond Binary Judgments: Using CNNs to Unravel Complex Sentiments in Movie Reviews

Ask Iver Sylte
Computer Science / 4th year
askis@stud.ntnu.no

Abstract

This project investigates the application of Convolutional Neural Networks (CNNs) to sentiment analysis of movie reviews, leveraging word embeddings from GloVe and RoBERTa to address both binary and numerical sentiment scoring. By transforming textual data into high-dimensional vectors, this study assesses the effectiveness of CNNs in decoding complex sentiment nuances embedded within movie reviews. The exploration reveals that while CNNs with RoBERTa embeddings outperform those with GloVe embeddings, they fall short of the benchmarks set by a pure RoBERTa transformer model. The project highlights the substantial impact of embedding quality on CNN performance, with contextual embeddings providing a distinct advantage over static ones. Furthermore, the adaptation of CNNs to handle evolving language over a decade and their ability to generalize across different sentiment labels are discussed.

1 Introduction

Sentiment analysis can be used to improve decision making, and is therefore of great interest to businesses. In the context of movie reviews, a model for sentiment analysis can help companies gather sentiment about movies outside of review sites, for example through social media. This project creates a model which can predict both binary sentiment and a numerical score for movie reviews. This can give companies a deeper understanding of public sentiment, enabling better market analysis. This is done by using the Large Movie Review Dataset from Stanford researchers containing 50 000 different movie reviews from the movie review site IMDB (Maas et al., 2011).

A CNN was chosen for this task, as this network architecture has previously shown promising ability in the field of natural language processing, due to its effectiveness at processing spatial hierarchies in data. No other study was found using a CNN to predict the score of reviews for the dataset, so this project is among the first to present this. The project checks the feasibility of predicting score of movie reviews using word embeddings and CNNs. Additionally, binary sentiment classification results are compared with other papers who have also used this dataset. This is explained in section 3. Two different word embedding models are used to study the difference between static embeddings and contextual embeddings. The word embedding models used are GloVe and RoBERTa. Going forward, the CNN using Glove vectors will be referred to as CNN-GloVe and the network using RoBERTa vectors will be called CNN-RoBERTa. Note that both models have almost identical structure, which is shown in section 4.

A Naive Bayes model and a RoBERTa model were used as baseline models for comparing the CNNs. Naive Bayes models are established methods for performing sentiment analysis, and RoBERTa is a state-of-the-art transformer model. The CNNs were evaluated against these baselines to determine their effectiveness in capturing nuanced sentiment from text data.

2 Background

This project contains tools, methods and terminology which will be unfamiliar for someone that does not have a good grasp of machine learning and natural language processing in general. This section will attempt to explain these concepts such that everyone can understand the contents of this project.

2.1 PyTorch

PyTorch is a framework for training deep learning models, and is recognized as one of the most popular machine learning libraries. It was chosen for this project due to its numerous features enabling easy customization for the models, as well as it's strong integration using GPU's, enabling faster training. (Paszke et al., 2019).

2.2 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a type of specialized neural network that typically learns using the spatial dependencies of data, for example in images. The network uses filters to convolve neighbouring values such that patterns can be found in the data. In text processing problems, this means that word-wise values are convolved together. The data is then passed through additional layers such as activation functions to introduce non-linearity, and pooling layers that reduce dimensionality and computational load (Lecun et al., 1998).

2.3 Word Embeddings

A CNN can not learn directly from textual data, and it is therefore necessary to encode the data into a numerical format. We use word embedding models for this, which transforms text into high-dimensionality vectors. Word embeddings specifically turn each individual word into its own vector, compared to other embedding types which turn sentences or entire paragraphs into a single vector[1]. The different dimensions are meant to represent different meanings of a word. Similar words will therefore be close to each other in the vector space (Mikolov et al., 2013). For creating word embeddings in this project, two different models were used: GloVe and RoBERTa.

2.3.1 GloVe

GloVe stand for Global Vectors for Word Representation, and is a method of word embedding used in Natural language processing. It builds a co-occurence matrix which finds how often words appear within the context of other words, and uses this information to create word vectors. GloVe is a static word embedding technique, meaning that each word in the vocabulary is assigned a single value irregardless of the surrounding context in the text. The word vectors created by the GloVe model used have 300 dimensions (Pennington et al., 2014).

2.3.2 RoBERTa

RoBERTa (Robustly Optimized BERT Approach) is an extension of the BERT model introduced by Google in 2018. RoBERTa uses the same model architecture as BERT, but significantly outperforms BERT on several NLP benchmarks, such as in sentiment analysis. RoBERTa can be used to generate contextual word embeddings. Unlike GloVe, word embeddings by RoBERTa are influenced by the context, meaning the representation of a word can change depending on the circumstance. A word vector created by RoBERTa have 768 dimensions (Liu et al., 2019).

2.4 spaCy

spaCy is one of the most popular libraries for natural language processing. In this project, spaCy was used to tokenize movie reviews for the Naive Bayes model and create word vectors using it's integration with the GloVe model. It's ease of use simplified the preprocessing step measurably, enabling a higher focus towards the rest of the project. (Honnibal and Montani, 2017).

2.5 Naive Bayes

The Naive Bayes model is a probabilistic classification technique based on Bayes' theorem, and assumes that there is independence among predictors, hence the "naive" in Naive Bayes. In this project specifically Multinominal Naive Bayes was used, since this model is tailored towards classification tasks that follows a multinominal distribution. Naive Bayes works by learning the frequency of words for different

[1]Transformer models typically do not always create tokens out of words, and sometimes use subword embeddings instead. This is also the case for the RoBERTa model

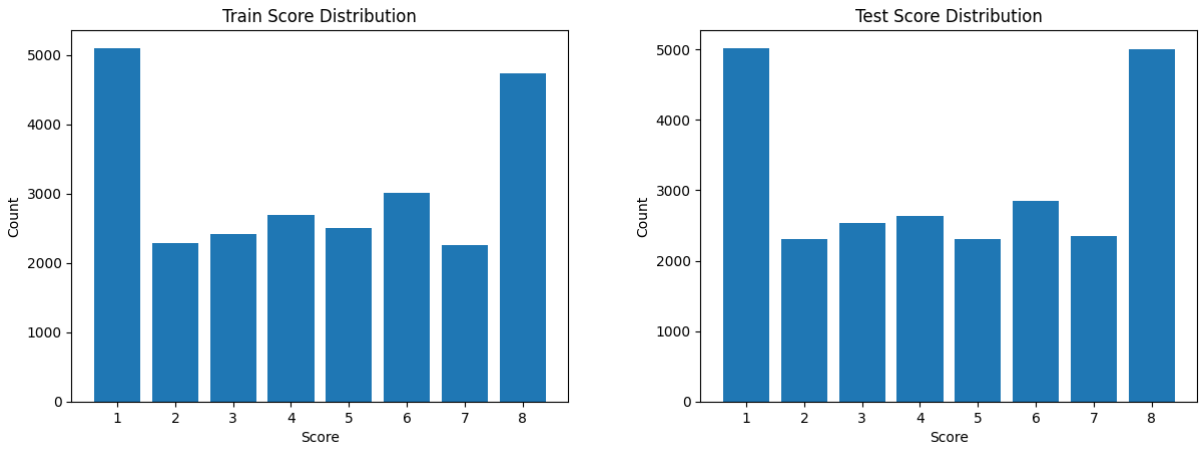
sentiments and calculating the most likely sentiment for each document. The sklearn library was used for creating a Naive Bayes model[2]. In this project, TF-IDF values was chosen to create the numerical features for the Naive Bayes model (John and Langley, 2013).

2.5.1 TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) is a statistic that represents how important a word is in a specific in a collection of documents. It consists of two parts, Term Frequency and Inverse Document Frequency. See 9.1 in appendix for how these values are calculated. TF-IDF produces a sparse vector with length equal to the number of unique terms in the corpus, and values for each term relative to their importance to the document. The sklearn library was used to generate these values[3] (Manning et al., 2008).

2.6 The Dataset

The dataset used for the sentiment analysis task has been created by researchers at Stanford university (Maas et al., 2011). The dataset contains 50 000 samples, with an even split between training and test samples. Additionally, each set has an equal amount of positive and negative reviews. Each review contains the score and the body of the review as a text file, which enables both binary classification and score prediction. Do note that the dataset only includes higher and lower scores while neutral reviews have been intentionally left out. Thus all reviews should clearly be either negative or positive. Given that there are no reviews with a score of 5 or 6 in the dataset, the scores of all positive reviews have been adjusted such that the scores are continuously distributed, enhancing model performance. This was done by simply subtracting 2 from all scores with a value of 7 or higher. It is also worth to note that the endpoints of the scoring range is overrepresented in the dataset, possibly due to self-selection bias. People who have strong opinions on a movie are more likely to leave a review, making the more extreme views more prominent. See figure 1 for distribution of scores in train and test set after this transformation.



(a) Transformed train set

(b) Transformed test set

Figure 1: Distribution of transformed train and test set

2.7 Evaluation Measures

Several evaluation measures were used to measure performance both for binary classification and score prediction.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

[3] TfidfVectorizer was the specific funtion used for this task. Link: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

2.7.1 Accuracy

The accuracy evaluation considers the amount of correct predictions from all predictions, and is given as a percentage of how many predictions were correct.

2.7.2 Precision

Precision measures the ability of the model to detect correct cases without falsely classifying them as correct. See 9.2 in appendix for how it is calculated. Both positive and negative precision is used to evaluate the models in this report.

2.7.3 Recall

Recall measures the ability of the model to identify all correct cases in a dataset. See 9.3 in appendix for how it's calculated. Both positive and negative recall is used to evaluate the models in this report.

2.7.4 MAE

Mean average error (MAE) calculates the error from every prediction and finds the mean error by dividing over the test set size. This is used to evaluate the models when predicting review scores instead of binary sentiment.

3 Related Work

The Large Movie Review Dataset is an established dataset for sentiment analysis on movie reviews. It has been used quite extensively to test different model architectures in the past, and is featured on a dedicated benchmarking site, PapersWithCode[4]. This site showcases different research papers and their performance on the dataset. This allows a direct comparison between the results of this project and earlier papers. However, few of the models featured on the site have a CNN architecture. The best CNN on the ranking list achieved an accuracy of 92.33% in the paper by Rie Johnson and Tong Zhang (Johnson and Zhang, 2015). This study used a bag-of-words approach instead of word vectors for their input data, so the there are some differences between the experiments. This was still chosen as a target value to beat in this project for binary classification, as it is the highest performing CNN model on the dataset.

A paper titled *Contextual Embeddings: When Are They Worth It?* from 2020 explored the performance differences between the static embedding model GloVe and contextual embedding model BERT. The study showed that the contextual embeddings generally outperforms the static ones, but with the GloVe embeddings getting an absolute accuracy within 10% of the BERT embeddings most of the time. The paper also highlights the increased complexity of contextual models, which leads to increased costs for inference and training (Arora et al., 2020). This project examines how accurate this metrics are for the movie review dataset.

4 Model

The model used in this project is a convolutional neural network trained using the PyTorch framework. The structure of the model has been tuned using previous experience and extensive testing during the project. The entire model structure is displayed in table 1.

4.1 Shape of Input Data

In this model the dimensions of the word vectors are used as input channels into the network, while the embedding length represent the length of each input. A CNN architecture require that all input data have the same dimensions, so a common embedding length for each review is necessary. The input is structured in a one-dimensional way with the channels representing the different word meanings from the embedding models. This enables the filters of the network to compare the values of neighbouring tokens in one dimension at a time, instead of capturing several dimensions of words during a convolution operation.

[4]https://paperswithcode.com/sota/sentiment-analysis-on-imdb

4.2 Flexible Model

The network is designed to accommodate different word embedding models without structural modifications. The two embedding models used in this project generate vectors of different dimensions, and therefore needs different amount of neurons for input into the feed-forward part of the network when flattened.

The final activation function is different when predicting binary classification versus score predictions. When predicting binary sentiment, the binary cross entropy loss function is used with a combined sigmoid layer, which transforms the output of the model into a value between 0 and 1. This is adequate when we are predicting binary classification. When we are predicting movie scores however, the ReLU activation function is used. This is due to the MAE loss function not having an implicit loss function included. The ReLU activation function was chosen for the final layer instead of GELU to disallow negative values for model output, since movie scores are not negative.

4.3 Kernel and Padding

The kernel used in the convolution layers ensure that the next layer only decreases in size due to max pooling between layers. Using a padding of 2 and a kernel size of 5 the output from the layer will be the same size as the input. The type of padding is zero padding, meaning two zeros are added at the front and end of the input into the layer before the convolution operation.

4.4 Roles of Layers

The different layers are important for defining the structure of the network The convolution layers perform convolution operations on input data, resulting in an transformed version of the input sequence. Batch normalization was used due to speeding up training and its stabilizing effects (Ioffe and Szegedy, 2015). Max-pooling was used to reduce the dimensionality of the data, and identify the most important tokens in a window of words. Activation functions are used to introduce non-linearity into the model. In this model GELU was chosen as it is a state-of-the-art activation function (Hendrycks and Gimpel, 2023).

5 Experiments and Results

This section goes into the details regarding the experiments of the project, and displays the results from the CNN models and the baseline models.

5.1 Experimental Setup

This section contains the hyperparameters for training the CNN models and what steps was necessary for preprocessing the data. The process for training the baseline models is also covered in this section.

5.1.1 Data Generation

Before any training can occur, the data must first be converted into a numerical format. The text from the reviews have not been altered much before being turned into tokens. The only manual preprocessing step was performing text cleaning, as the movie reviews contained html breakline tags. This was done due to the belief that breaklines would not add much information regarding sentiment. The reviews were then broken down into tokens by Spacy or the RoBERTa tokenizer. These tokens were then used to generate word embeddings from GloVe and RoBERTa respectively. The word vectors was extracted from the models and the length of each sequence of word vectors was normalized such that each movie review was represented by 200 tokens. The sequences were then saved to file with their corresponding review scores. See figure 2 for visual flow chart.

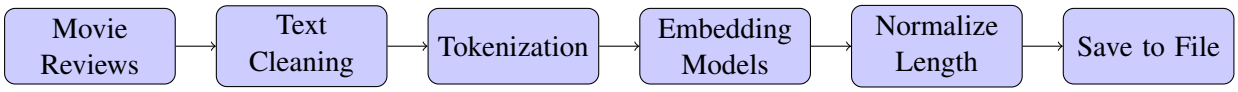


Figure 2: Process flow from movie review to saving embeddings to file

Table 1: Detailed Architecture of the CNN Models

Operation	Input → Output Shape	Details	Activation
Conv1d	(300 or 768)→ 64 channels	Kernel=5, Stride=1, Padding=2	-
BatchNorm1d	64 channels	-	-
GELU	-	-	GELU
MaxPool1d	64 channels	Kernel=2, Stride=2, Padding=0	-
Conv1d	64 → 128 channels	Kernel=5, Stride=1, Padding=2	-
BatchNorm1d	128 channels	-	-
GELU	-	-	GELU
MaxPool1d	128 channels	Kernel=2, Stride=2, Padding=0	-
Conv1d	128 → 256 channels	Kernel=5, Stride=1, Padding=2	-
BatchNorm1d	256 channels	-	-
GELU	-	-	GELU
MaxPool1d	256 channels	Kernel=2, Stride=2, Padding=0	-
Flatten	-	-	-
Linear	Variable → 400	76800 (GloVe) or 196608 (RoBERTa)	-
BatchNorm1d	400 units	-	-
GELU	-	-	GELU
Linear	400 → 200	-	-
BatchNorm1d	200 units	-	-
GELU	-	-	GELU
Linear	200 → 1	-	Identity or ReLU

5.1.2 CNN Hyperparameters

Configuring a model's hyperparameters correctly is essential for a model learn effectively. These values were found over time during testing of the network. See table 2 for a full overview. These are the most important ones:

- Added noise std is a value which decides how randomized the training samples are during training. Each batch samples values from a normal distribution with a mean of 0 and a standard devation of the noise parameter, and add this value to each individual sample for that training epoch. This helps the model generalize and makes it less likely to overfit.
- Weight decay is also meant to generalize the model. The weight decay is a form of regularization which punishes the growth of weights in the network, forcing them to be lower. This makes the network less likely to overfit on the training data.
- The Loss function is used to generate a value for the difference between the predicted value and the target value. The size of this value decides to what degree the weights in the network will change after being multiplied by the learning rate. Cross Entropy was used for binary classification, and MAE loss was used for score prediction.

5.1.3 Naive Bayes Training

To train the Naive bayes model, the reviews were first tokenized using the spaCy framework. The data was then shuffled, before it was turned into TF-IDF vectors using a n-gram range of 1. Finally, the

[5]Note that while a random seed has been used during the training of the model, there could be slight variations in results if the experiment is repeated. This is because the models were trained using GPUs, which inherently include non-deterministic operations due to parallel execution. This approach was chosen to significantly reduce training time, accepting a minor trade-off in reproducibility.

Table 2: Model Training Parameters

Parameter	Value
Random seed	0 [5]
Embedding length	200
Batch Size	32
Number of Epochs	10
Added noise std	0.25
Loss Function	Cross Entropy or MAE
Optimizer Parameters	
Optimizer Type	Adam
Learning Rate	1×10^{-4}
Weight decay	1×10^{-1}

model was trained on the TF-IDF vectors. The test data was transformed using the same trained TF-IDF vectorizer, before results were predicted using the Naive Bayes model.

5.1.4 RoBERTa Training

To train the RoBERTa model, a RoBERTa tokenizer was used for tokenizing the data. This uses sub-word tokenization, unlike spaCy's word tokenization. The context length for this tokenizer was set to 256, which is discussed in section 7.1. The tokenizer truncated longer reviews and padded shorter reviews such that the tokenized length was normalized. The number of training epochs was set to 10, same as for the CNNs. See 11.1 in appendix for overview of chosen hyperparameters during training.

5.2 Experimental Results

This section contains the results of the CNN models as well as the baseline models. This includes the results of binary sentiment classification and review score predictions. Note that the scores displayed are the best scores obtained from a model during training. This is discussed in section 7.2.

5.2.1 Binary Sentiment Results

The results of binary sentiment classification on the dataset can be seen in table 3. Seeing as there is no difference between predicting correct for a positive or negative sentiment for a movie review, accuracy is the most important measure for evaluating model performance. There is a clear performance divide between the models, where the RoBERTa model and the CNN-RoBERTa performed quite similarly, and the Naive Bayes model performed similar to the CNN-GloVe model. It is also interesting to note the low positive recall and the high negative recall in the Naive Bayes model compared to its overall accuracy. This indicates that it more often misses a positive review compared to negative reviews. This might be due to reviews with negative sentiment using specific words more often.

Table 3: Results for binary sentiment classification sorted by accuracy

Model	Accuracy	Pos. Precision	Pos. Recall	Neg. Precision	Neg. Recall
RoBERTa	0.9356	0.9298	0.9422	0.9415	0.9289
CNN (RoBERTa vector)	0.9267	0.9155	0.9402	0.9385	0.9132
Naive Bayes	0.8316	0.8644	0.7866	0.8042	0.8766
CNN (GloVe vector)	0.8308	0.8320	0.8290	0.8296	0.8326

5.2.2 Score Prediction Results

The results of review score prediction on the dataset can be seen in table 4. MAE have been chosen as the most important metric for evaluating these models, as a model can have a high accuracy score but

miss by large margins in other predictions, which is not desirable. The results of the Naive Bayes model displays this, with a better accuracy score than both CNN models, but much worse MAE. This can be explained by the fact that the dataset is unbalanced, favoring the extremes to a higher degree. This is explored in section 6. For calculating accuracy, the model outputs was rounded and converted to nearest possible scores. The accuracy is quite a bit worse when predicting scores, which makes intuitive sense, as there are now 8 possible classes to predict instead of 2. It's noteworthy how much higher the accuracy of the RoBERTa model is compared to all other models, given that the MAE is only about 0.2 lower than the CNN-RoBERTa model.

Table 4: Evaluation Metrics for Regression Models sorted by MAE

Model	Accuracy	MAE
RoBERTa	0.4671	0.8156
CNN (RoBERTa vector)	0.3304	1.0244
CNN (GloVe vector)	0.2836	1.3663
Naive Bayes	0.3433	1.9714

6 Evaluation

This section evaluates the results from binary sentiment classification and score prediction on the dataset.

6.1 Binary Sentiment

The results obtained are better than previous studies and displays the importance of contextual word embeddings. The CNN-RoBERTa achieved an accuracy score of 92.67%, surpassing the target of 92.33% set in section 3. While this is only slightly better than the target value, this would be the best CNN ranking on the benchmarking site if the test results are valid. This is discussed in section 7.2.

However, it was the Transformer model RoBERTa which performed the best. This is not surprising when you consider the rankings on the aforementioned benchmarking website for the dataset. Currently both of the two highest rankings use a version of RoBERTa, achieving accuracies of 96.68% and 96.54%. While this is considerably better than 93.56%, one would still expect the RoBERTa model to perform the best out of all models in this project.

It is important to highlight the difference in performance between the two different CNN models, as they are separated by almost 10% in accuracy, CNN-GloVe having over double the error rate compared to the CNN-RoBERTa model. This aligns with what was previously established in section 3. It is therefore quite clear that the quality, and perhaps also the dimensionality of the word vectors fed into a CNN model have an important impact on model performance.

The results from the Naive Bayes model were also quite impressive. It was assumed that the Naive Bayes model would perform much worse than all other models, due to using simpler methodology compared to the other models. This was clearly not the case, matching the accuracy of the CNN-GloVe model.

6.2 Score Prediction

The difference between the models become more prominent when evaluating the their score predictions. When evaluating by MAE, each model has a distinct ranking, with the RoBERTa model being the clear best.

The Naive Bayes model showed decent accuracy, but quite bad MAE compared to the other models. This is probably caused by the uneven dataset, as Naive Bayes models tend to overpredict the majority class when the training data is skewed, due to the prior probability of these classes being increased. It will therefore often just guess scores of 1 or 8, and miss quite heavily when wrong.

The large difference in accuracy between the RoBERTa model and the CNN-RoBERTa model can perhaps, in part be attributed to usage of different loss functions. While the CNN models have used

a MAE loss during training, the RoBERTa model uses a mean squared error (MSE) loss function for regression tasks. The MSE loss function punish large differences more harshly than the MAE loss function, forcing the model to predict more confidently. A MAE loss function for RoBERTa might therefore have improved the MAE score, while decreasing the accuracy.

Achieving an MAE around 1 for the best models show that predicting scores for movie reviews, while being more difficult than predicting binary sentiment, can still be a viable form of sentiment analysis moving forward.

7 Discussion

This section discusses some of the important decisions made during the project and what could have been done differently.

7.1 Embedding Length

Since all data must share the same dimensions before training a CNN, a common length must be chosen for the word vectors representing a review. Unfortunately this choice was somewhat forced due to hardware limitations. The reviews embedded by the RoBERTa model consisted of 30 GB overall, while the machine used for training had 32 GB of ram. Increasing the embedding length over 200, while still loading in all training data was therefore not seen as possible. This could lead to a loss of information in the word vectors, as parts of longer reviews are not included in the data fed to the CNN model. A higher performance could therefore be achieved for the CNN models with more RAM capacity.

The Naive Bayes model did not have this problem, as TF-IDF scores were used to train the model. These have much lower memory demands, and it was therefore possible to use the entire length of all reviews when training the Naive Bayes model. Imposing the same limitations on length as for the other models could have impacted the performance negatively.

The RoBERTa model also suffered from hardware limitations during training. The RoBERTa tokenizer has a max context length of 512 tokens, meaning it can process 512 different tokens for maximizing it's understanding of each individual token. However, due to VRAM limitations of the GPU used, the context length used during training was 256, half of the maximum possible length. This limitation might have hindered performance. Note that this limitation only affected the training of the RoBERTa model, and did not limit retrieving word embeddings from the model. In that case the full context length of 512 could be used by the RoBERT. This could explain why the RoBERTa model trained here did not achieve the same results as shown on the benchmarking site for the dataset.

For this project, only the start of reviews were used when considering these length limitations. Attempting to instead use the last portion of a review, or the middle part of a review could be more beneficial for understanding the sentiment in longer reviews. However it is also necessary to consider the fact that about 75% of the reviews consist of less than 200 tokens. Most of the data will therefore include all of the context contained in the reviews with this embedding length.

7.2 Test Set Evaluation

The results from the models shown in section 5.2 include the best model performance found during training. This was found by evaluating the model on the test set after each epoch, and selecting the model with the best result. Example of this can be seen in figure 3. This incentivizes models that overfit on the test set, which is not desirable, as we want a generalizable model. To strengthen the results of the models a validation set should have been included to refine the network parameters, before doing a final evaluation on the test set. Because of this, it is possible that the RoBERTa and CNN models would perform slightly worse. The Naive Bayes model is unaffected by this, as the test set has not been used to tune the performance of the Naive Bayes model.

7.3 Text Preprocessing

The results in this experiment was gathered from text which had not been altered much before being converted to numerical data. The only manual preprocessing before tokenization was removing html

line breaks. Removing stop words, lemmatization of words, lowercasing or punctuation removal could all be considered for attempting to improve performance for CNN-GloVe or the Naive Bayes model. These models assign values to terms statically, so reducing the number of non-important words and reducing word variants to the same common base form could create higher quality data and improve model performance. The reason this was not done is due to the fact that these preprocessing steps is not usually done for transformer models, as they benefit from the full context in the text. Since we want to compare the models using the same content, minimal preprocessing was therefore done, though this might have hurt the results of the Naive Bayes model and the CNN-GloVe model.

7.4 Dataset Assumptions

The dataset used for this project included positive or negative reviews, while neutral reviews were left out completely. Including these types of reviews into the dataset would provide a more realistic scenario, as not all movie reviews in real life are polarizing. Predicting neutral reviews might prove challenging for the models, as these reviews tend to use milder language comparatively, which might be harder to classify.

Additionally the dataset originates from 2011, so the newest reviews are approximately 13 years old. Typical language used for movie reviews might have evolved considerably since then, which could influence the difficulty of predicting today's movie reviews. Future research could compare the language of movie reviews from different periods, or update the dataset with more recent reviews to explore potential changes to model performance.

8 Conclusion and Future Work

This study has demonstrated the application of Convolutional Neural Networks (CNNs) for sentiment analysis on the Large Movie Review Dataset, focusing on both binary sentiment classification and numerical score prediction. The results indicate that while CNNs are capable of effectively extracting and analyzing sentiment from textual data, the performance significantly varies depending on the type of word embeddings used. Specifically, CNNs utilizing RoBERTa embeddings (CNN-RoBERTa) showcased superior performance over those using GloVe embeddings (CNN-GloVe), highlighting the importance of embedding quality and contextual awareness in sentiment analysis tasks. The results show that CNNs outperform the Naive Bayes model considerably in both tasks, but get beaten itself by the RoBERTa model. This highlights the strength of advanced transformer models in natural language processing. Future projects should explore the impact of using different embedding lengths and incorporating new and different data, as these suggestions could have a large impact on model performance.

References

Simran Arora, Avner May, Jian Zhang, and Christopher Ré. Contextual embeddings: When are they worth it? In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2650–2663, Online, July 2020. Association for Computational Linguistics. doi:10.18653/v1/2020.acl-main.236. URL https://aclanthology.org/2020.acl-main.236.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023.

Matthew Honnibal and Ines Montani. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. https://spacy.io, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

George H. John and Pat Langley. Estimating continuous distributions in bayesian classifiers, 2013.

Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. In Rada Mihalcea, Joyce Chai, and Anoop Sarkar, editors, *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies, pages 103–112, Denver, Colorado, May–June 2015. Association for Computational Linguistics. doi:10.3115/v1/N15-1011. URL https://aclanthology.org/N15-1011.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi:10.1109/5.726791.

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.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. ISBN 9780521865715.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi:10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

9 Formulas

9.1 TF-IDF

Term Frequency (TF) measures how frequently a term occurs in a document. It is calculated as:

$$TF(t,d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d} \tag{1}$$

Inverse Document Frequency (IDF) measures how important a term is. Unlike TF, IDF considers the terms in other documents, such that more rare terms are weighed more heavily. IDF is calculated as:

$$IDF(t,D) = \log\left(\frac{\text{Total number of documents } |D|}{\text{Number of documents with term } t \text{ in them}}\right) \tag{2}$$

To find the TF-IDF value for a term in a document, these two terms are multiplied together:

$$TF - IDF(t,d,D) = TF(t,d) * IDF(t,D) \tag{3}$$

9.2 Precision

Precision (P) is defined as the number of true positives (TP) over the number of true positives plus the number of false positives (FP):

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4}$$

9.3 Recall

Recall (R), also known as Sensitivity or True Positive Rate, is defined as the number of true positives over the number of true positives plus the number of false negatives (FN):

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5}$$

10 Plots

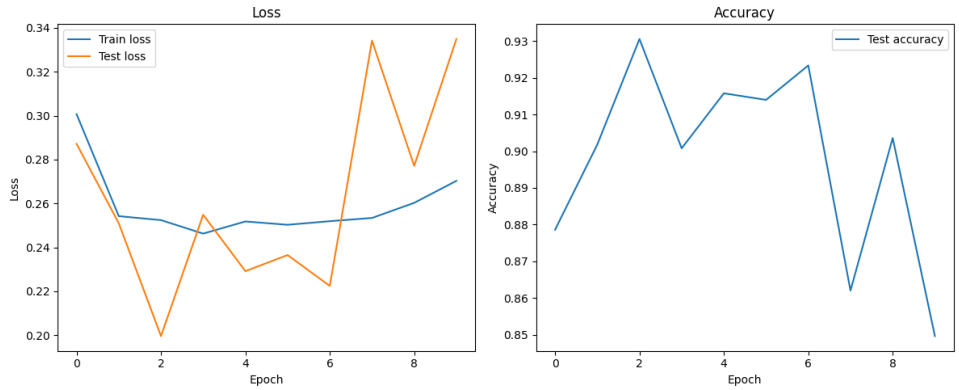


Figure 3: CNN-RoBERTa binary sentiment training history[a]

[a]Only 20% of the test set was evaluated after each epoch in this plot due to memory limitations.

11 Hyperparameters

11.1 RoBERTa

Table 5: Training Hyperparameters RoBERTa

Parameter	**Value**
Floating Point Precision (fp16)	True
Number of Training Epochs	10
Batch Size per Device (Training)	32
Gradient Accumulation Steps	1
Batch Size per Device (Evaluation)	32
Warm-up Steps	500
Weight Decay	0.01
Evaluation Strategy	'epoch'