**Project Report:**

**Task 1: Sentiment Analysis on Movie Reviews**

Course: Project: NLP DLBAIPNLP01

By Dmitrii Shevchuk

Matriculation: 9213737

First attempt, 28.04.2025

Tutor: Simon Martin

## Table of Contents

## 1. Introduction

We are going to build a sentiment analysis (SA) model for movie reviews. SA is a Natural Language Processing (NLP) task aiming to generate sentiment marks for a given text, movie reviews in our case. Sentiment marks can be positive, negative or neutral. In this project we use only positive or negative marks.

SA research is useful not only for tracking market attitudes towards brands and products. It is also a first step to emotion recognition, better understanding of human speech and making generated speech more expressive (Cambria, 2016).

For training and testing we will use the “Large Movie Review Dataset” (IMDb dataset) available on Hugging Face. User-submitted movie reviews were collected from IMDb (Internet Movie Database) in 2011. Each review intinially had a movie rating from 1 to 10. Ratings ≥ 7/10 were marked as positive, ratings ≤ 4/10 as negative, the rest (neutral) were excluded from the dataset. Exactly 25K reviews are positive, and 25K negative, making the dataset well balanced. Also it has 50K unlabelled reviews to be used for Masked Language Modeling (MLM).

There are several model state-of-the-art (SOTA) SA models. Paramesha et al. (2023) used Bidirectional Encoder Representations from Transformers (BERT) as a backbone to test different traditional machine learning models for sentiment classification. They got the best accuracy of 95.5% on the document level for BERT-LARGE + Linear Classifier. Nkhata et al. (2025) fine-tuned BERT-BASE appended with Bidirectional Long Short-Term Memory (BiLSTM) to achieve 97.67% accuracy on IMDb dataset, which is the best result compared to 9 other SOTA deep learning (DL) models. Biplov et al. (2025) fine-tuned RoBERTa, an improved BERT model trained on a 10 times larger dataset, with Region-based Convolutional Neural Network (R-CNN) as a classification layer and got accuracy of 91.5%.

The reason why DL models based on transformers are so successful was noticed by Paramesha et al. (2023): “word expressing sentiments are highly sensitive to the domain and also within the domain, same words may express different opinions depending on their context”. Hence the model should be able to recognize context dynamically. For example, the word “terrifying” can refer to a good movie in the horror genre or a bad movie in the comedy one. For dynamic context we need transformers.

Straight forward approach then is to use well-trained Masked Language Model (MLM), like BERT or RoBERTa, and add smaller transformer classification head. We can also train our own MLM on domain specific movie reviews to make it better than more general BERT in understanding review language nuances.

However too complex models not always work better, this needs to be tested. Also such models need a lot of compute resources for training and inference. Having only RTX 2060 12GB GPU

available for limited time, we need to bootstrap. And because this is a student project, it would be better to train own MLM, than take a ready one.

Hence the design of our model: small custom transformer based MLM with a a very simple DL classification head. In the following sections we will describe the data preparation, training and evaluating of custom tokenizer, MLM, classification head, and discuss the results in conclusion.

### 2.1. Data preparation

IMDb dataset contains movie reviews in English. There are no missing texts or labels. Negative and positive reviews are perfectly balanced. But some cleaning is still needed.

We made all characters lowercase, removed HTML tags and URL's, control characters, math symbols and other rare symbols. We found 17 emojis in the dataset and cleaned them also, although if we had more of them, we might keep them as they are obviously connected to the text sentiment. However there were more than 1000 emoticons, which we kept. We also removed stand alone symbols like ":" or "/" and ":" right after a letter, as well as extra white spaces.

We kept punctuation, one letter words, words like "and", "the" and similar, because they are important for the context, and may significantly change the meaning or sentiment of a sentence. BERT/RoBERTa also have them in vocabulary. We ended up with 500K words total in the IMDb dataset after the cleaning.

### 2.2 Tokenizer

To convert text to numbers we need a tokenizer. BERT uses WordPiece and RoBERTa uses Byte-Pair Encoding (BPE) trained on their data. Tokenizer is basically a dictionary of most common words and subwords plus special tokens. WordPiece is considered better for smaller datasets of natural language texts with clear word boundaries. BPE is better for multilingual data or code and GPT-style generation.

We could take a ready tokenizer trained for BERT with 30K tokens vocabulary, but training our own is also fine. In Table 1 you can see a comparison of our own WordPiece tokenizer with 30K vocabulary trained on IMDb dataset (training and unlabeled parts, 75K reviews), and BERT tokenizer. Our own vocabulary covered 100% of the text: 0 UNK tokens. BERT tokenizer produced 10 UNK tokens, meaning 10 words in the IMDb dataset were not in vocabulary. We could try even smaller vocabulary, allow some UNK tokens, but reduce dimensionality of the classification model and save compute resources.

As it is seen from Table 1, our custom tokenizer managed to represent the whole text with little bit shorter mean and median sequences. Hence if we choose 768 token context window for our model we cut off less text than in case of BERT tokenizer, which is good.

Table 1. Comparison of BERT and our tokenizers (sequence means one review in our case).

| Feature | BERT tokenizer | Our tokenizer |
|---|---|---|
| **Max sequence length** | 3257 | 3166 |
| **Mean sequence length** | 293.0 | 281.0 |
| **Median sequence length** | 219.0 | 210.0 |
| **95th sequence percentile** | 754.0 | 725.0 |
| **99th sequence percentile** | 1142.0 | 1102.0 |
| **UNK tokens** | 10 | 0 |

Source: our computation.

Of course, we could also increase the context window, say to 1024, but then we will need about 1.8x more VRAM and make the runtime about 1.8x slower due to quadratic scaling of attention. To avoid that we keep 768 context window and cut about 5% of sequences.

Last step here is tokenizing our dataset and adding special tokens like [CLS], [SEP], [PAD] and [MASK]. [CLS] - classification token placed in the start of the sequence, [SEP] - separator, used for marking dialogs, redundant in our case, but we keep it in the end of each sequence, [PAD] is used to fill every sequence up to our context window size, and [MASK] will be used in training MLM. We also produce an attention mask for each sequence - a vector of zeros and ones showing to the model where non [PAD] tokens are.

### 2.3 Masked Language Model (MLM)

MLM is a model for prediction of masked (missing) words in a sequence. If we have a sequence “I love cats, but not dogs” and mask word “cats” then MLM presented with the sequence “I love [MASK], but not dogs” should output high probability for the missing word “cats”. For the training of such a model we need to randomly mask some words in the sequence. We can do it statically by producing a new dataset with random masking, or dynamically by randomly masking every batch of sequences again and again for each epoch. Dynamic masking basically expands our effective training dataset many times with about 10% additional compute, which is very good for small datasets like our.

Other model parameters were chosen from memory constraints of our GPU (12GB) and up to 8 hours for training (Table 2). More memory and more time would allow more attention heads, layers and training time, and probably would improve the performance, but this needs to be tested.

Table 2. Comparison of BERT family models with our custom MLM model

| Feature | BERT-base | BERT-large | RoBERTa-base | Our Model |
|---|---|---|---|---|
| **Params** | 110M | 340M | 125M | 20M |
| **Layers** | 12 | 24 | 12 | 6 |
| **Hidden Size (d_model)** | 768 | 1024 | 768 | 512 |
| **Attention Heads** | 12 | 16 | 12 | 8 |
| **FFN Size** | 3072 (4×) | 4096 (4×) | 3072 (4×) | 2048 (4×) |
| **Max Seq Length** | 512 | 512 | 512 | 768 |
| **Vocabulary Size** | 30K | 30K | 50K | 30K |
| **Data (tokens)** | 3.3B | 3.3B | 30B | 0.05B |
| **Batch Size** | 256 | 256 | 8K | 64 |
| **Train epochs** | 40 | 33 | 1 | 24 |
| **Masking** | Static | Static | Dynamic | Dynamic |
| **FP** | 32 | 32 | 32 | Mixed(16/32) |

Source: Liu et al. (2019), Devlin et al. (2019).

Hidden size is a dimensionality of word embedding vector representing one word to reflect its meaning relatively to other words. Word embeddings in transformers are trained to be dynamic as opposed to static embedding models like Word2Vec. Due to the self-attention mechanism in transformers each embedding becomes a function of this word and other words in the sequence. Which makes this model so powerful.

Having a smaller embedding size reduces the ability of the model to represent more subtle meanings of words and contexts, but saves about 30% of compute resources. However if we needed to improve the model having more resources, we would probably increase hidden size as a first step.

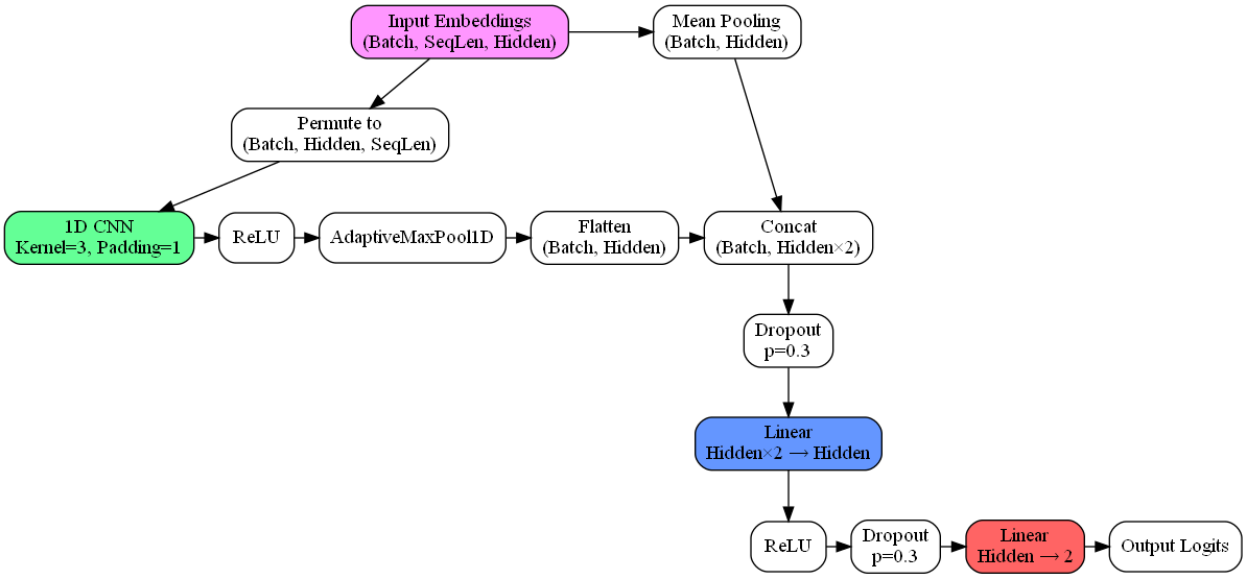
Training our MLM for 24 epochs ended up with validation accuracy of 0.305, which is reasonable. With more resources we might train it 30-40 epochs.

## 2.4 Classification model

We trained our MLM to guess the missing words in a sequence. For sentiment classification we need to update it with a new “head” producing sentiment guesses out of token embeddings tensors.

Classification head can be R-CNN like in Biplov et al. (2025), BiLSTM like in Nkhata et al. (2025), various traditional ML or DL models like in Paramesha et al. (2023). We will try a lightweight hybrid architecture designed to combine local (CNN) and global (Mean Pooling) information (Figure 1).

Figure 1. Architecture of the classification head.




Process starts with embeddings from MLM in shape [batch_size, seq_len, hidden_size]. We feed them to the CNN layer (nn.Conv1d) after permuting them to [batch_size, hidden_size, seq_len]. We use kernel equal to 3 to catch phrases like “not very good”, but not bigger to stay local and conservative with compute resources. Padding is needed to preserve the output length. ReLU layer adds non-linearity for cases like "not bad" where we need to flip the sentiment because of the word “not”, and suppress non-relevant features. After that we apply max pooling to get the most prominent features for each channel and receive one embedding for each sequence flattened to the tensor [batch_size, hidden_size].

On the next step we create a merged vector [flattened_maxpool, mean_pooled]. Flattened_maxpool contains important local information and mean_pooled - global one, averaged out overall tone of the sequence. Random dropout of 30% prevents the model from overly trusting a handful of feature pairs. This makes the model more robust to the presence of strong words and forces it to take in account also other words before choosing the sentiment.

Then we use nn.Linear and ReLU layers to combine most important global and local information to get back to [batch_size, hidden_size] shape and dropout again. Last Linear layer is trained to convert an embedding into sentiment class logits of size 2, which we can use to compute probabilities of each sentiment via softmax.

Fine-tuning this model on 25K labeled reviews is much faster than MLM training, after 100 epochs we got valuation accuracy 88%.

Inference on the test IMDb dataset (25K reviews) showed slightly lower 86% accuracy, which is below SOTA DL models, but on the level of the best traditional ML models (Paramesha et al., 2023, Table 6). In Table 3 we report accuracy of our and benchmark models by Nkhata et al. (2025) for 4 datasets: IMDb movie reviews test subset, Movie Review Data (MR) containing 2000 reviews (Pang and Lee, 2004), Stanford Sentiment Treebank (SST-2) consisting of 70K movie reviews (Pang and Lee, 2005), and Amazon dataset containing 747.8K movie and TV shows reviews from 1996 till 2023 (Hou et al., 2024). Last two datasets we truncate to 25K balanced datasets to save compute resources (evaluation was done on CPU).

Table 3. Accuracy of benchmark and our models on unseen review data.

| Model | IMDb test | MR | SST-2 | Amazon |
|---|---|---|---|---|
| **BiLSTM** | 90.42 | 90.5 | 91.12 | 92.18 |
| **BERT** | 93.81 | 94.29 | 93.55 | 94.78 |
| **BERT+BiLSTM-SA** | 97.67 | 97.88 | 97.62 | 98.76 |
| **Our model** | 86.10 | 77.15 | 44.50 | 50.80 |

Source: Nkhata et al. (2025) and our calculations.

## 3. Conclusion

The aim of the project was to build an NLP system able to recognise sentiment (positive or negative) of a movie review. For training we used the IMDb dataset. Architecture of the model was chosen based on limited compute resources, but with intention to make it domain specific. Hence we trained our custom tokenizer, small transformer MLM, and fine-tuned it with a hybrid classification head.

Our model showed accuracy on the level of the best traditional ML models, but inferior to the SOTA DL models on the unseen IMDb data, and fared even worse on three other movie review datasets. To improve the result we need probably both more elborate MLM and classification head.

## Bibliography


Biplov, P., Bipul, T., & Bishwash, P. (2025). Sentiment analysis of movie reviews: A flask application using CNN with RoBERTa embeddings. Systems and Soft Computing, 7, 200192. https://doi.org/10.1016/j.sasc.2025.200192

Cambria, E. (2016). Affective computing and sentiment analysis. IEEE Intelligent Systems, 31(2), 102–107. https://doi.org/10.1109/MIS.2016.31

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (pp. 4171–4186). Association for Computational Linguistics.

Hou, Y., Li, J., He, Z., Yan, A., Chen, X., & McAuley, J. (2024). Bridging language and items for retrieval and recommendation. arXiv. https://arxiv.org/abs/2403.03952

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. arXiv. https://arxiv.org/abs/1907.11692

Nkhata, G., Anjum, U., & Zhan, J. (2025). Sentiment analysis of movie reviews using BERT. arXiv. https://arxiv.org/abs/2502.18841

Pang, B., & Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity. Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004) (pp. 271–278). https://aclanthology.org/P04-1035

Pang, B., & Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005) (pp. 115–124). https://doi.org/10.3115/1219840.1219855

Paramesha, K., Gururaj, H. L., Nayyar, A., & others. (2023). Sentiment analysis on cross-domain textual data using classical and deep learning approaches. Multimedia Tools and Applications, 82, 30759–30782. https://doi.org/10.1007/s11042-023-14427-9