Sarcasm detection

Marcin Kądziołka[1], Marcin Podhajski[2], Illia Andrieiev[3]

{[1]marcin.kadziolka, [2]marcin.podhajski, [3]illia.andrieiev}@student.uj.edu.pl

Supervisor Andrii Krutsylo
andrii.krutsylo@uj.edu.pl

Abstract

Sarcasm detection is a challenging task in natural language processing, with applications in online communication, sentiment analysis, and virtual assistants. In this project, we focus on sarcasm detection in news headlines using transformer-based models. We compare our results with a baseline architecture and a state-of-the-art model. Our goal is to achieve comparable performance using a various transformer architectures. We experiment with different attention mechanisms, including default attention, wide attention, and a fast attention mechanism that reduces computational complexity. We also explore the impact of word embeddings on model performance. We evaluate our models on a dataset of sarcastic and non-sarcastic news headlines and use binary cross entropy as the loss criterion. Through our experiments, our aim is to improve sarcasm detection and contribute to the understanding of transformer-based models in NLP.

1 Introduction

Text classification is a classical task in Natural Language Processing and machine learning where the objective is to assign predefined categories or labels to a text based on its content. The task has been heavily researched in the past years and recently transformer-based models have been yielding significant results.

Sarcasm detection is a kind of text classification problem where the goal is to predict whether the text has been used sarcastically. Sarcasm is a form of expression that involves saying something contrary to the intended meaning for humorous reasons. Sarcasm is often characterized by the use of tone or intonation, therefore it is harder to recognize in a written text even for humans.

Quick and accurate detection would be useful in many areas. In online communication, it could help identify potential trolling or hate speech, and provide better moderation of online conversations. Sentiment analysis in market research would also be beneficial where accurate sarcasm detection can help distinguish between genuine positive or negative sentiments and sarcastic remarks, providing more accurate insights into consumer preferences

and opinions. Various virtual assistants and chatbots rely on user intent, which is crucial for good communication.

In our project we are going to work with the *News headlines dataset* which was introduced in [2]. The dataset consists of headlines from two news websites: *TheOnion* and *HuffPost*. The first one produces sarcastic articles about real events.

We use the architecture from [2] as a baseline and we want to compare our results with them. We will also compare to the recently published [7] which holds the current SOTA result.

Our goal is to achieve comparable results using a different architecture - currently commonly used in NLP transformers. We will try to fine-tune one of the available pre-trained models on the sarcasm dataset (probably GPT2 - depending on our computing resources). The second step is training a transformer from scratch and comparing results for different attention types.

2 Related work

There have been numerous methodologies employed for the detection of sarcasm in textual data across various datasets [8][9][2][10]. [1] addresses this issue by incorporating contextual information, which holds significance in the final classification outcome. The authors utilize a sampling technique from a unigram distribution estimated from posts contributed by all twitter users. By selecting the most frequently employed words as negative samples, the model encourages the representations to capture dissimilarities between an individual's word usage and commonly employed vocabulary. This enables the model to identify patterns indicative of sarcasm. The architectural design of the model involves the utilization of convolutions and linear layers for the purpose of classification.

In [2] the authors extend the convolutional architecture utilized in sarcasm detection on a Twitter dataset [1]. However, they applied it to the *News headlines dataset*. In this adaptation, they exclude user embeddings and introduce an LSTM component that generates attention values. These attention values are then concatenated with the output of the CNN. Another approach [6], encompasses attentional multi-reading to effectively incorporate information from both the utterance and the conversational context, by taking responses and answers from the SARC (reddit)[8] dataset, thus enhancing sarcasm detection. This methodology comprises four distinct steps: input embedding, attention mechanism, re-reading, and classification. The model architecture employed in this study involves the utilization of Bidirectional Long Short-Term Memory (Bi-LSTM) layers and linear layers.

[7] propose a novel method for sarcasm detection in text using a combination of Bidirectional Encoder Representations from Transformers (BERT) and Graph Convolutional Networks (GCN). Their approach leverages both semantic and structural understanding of the text. BERT is employed to encode the text into vector representations, while GCN captures syntactic dependencies using a graph structure. These representations are then fed into a classifier for sarcasm prediction. Other works that utilize transformer architecture are [11] which proposes a neural network methodology that combines a pre-trained transformer-based network architecture with a recurrent convolutional neural network (RCNN), [12] which uses LSTM and Transformers or [13] which deals with an imbalanced dataset by using

text augmentations.

3 Model

Our model is a transformer. We use token and positional embeddings. Then attention is applied, after which results are max pooled over time dimension. Lastly, the output is compressed by a linear layer to one value. Applying sigmoid returns values between 0 and 1, if the sentence is sarcastic or not.

3.1 Attention

Attention is a way of propagating only the most important information further. There are many different ways of implementing this mechanism. We treat attention from [3] as default. For each sentence keys, queries, and values are created. Then, a dot product of each query (independently) with each key is computed. The result is softmaxed, outputting attention values between 0 and 1. They are multiplied by values, which in consequence propagates only the information that is most important (has high attention value). The unit that computes that is called a head. But because single head may not be able to differentiate various types of information, multi-head attention is used. Now, each head may learn to linearly transform keys, queries, and values to better encapsulate some of the information. For example, there might be a "verb" head, where noun queries "ask" for any verbs and verbs keys "answer" (point in a similar direction)2. Formally, each head computes attention values as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

With multi-head attention, we get:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1,\ ...,\ \text{head}_h)W^O$$

$$\textbf{where } \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

and the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}, W_i^K \in \mathbb{R}^{d_{model} \times d_k}, W_i^V \in \mathbb{R}^{d_{model} \times d_v}, W^O \in \mathbb{R}^{hd_v \times d_{model}}$. We test a different number of heads h. Same as [3] for each head we use $d_k = d_v = d_{model}/h$ unless stated otherwise.

3.2 Wide attention

Reducing the dimension size for each head reduces the total computational cost, but also may reduce the ability to capture complex information. Therefore, we experiment with attention where $d_k = d_v = d_{model}$, and call it wide attention. Note that wide attention is identical to default attention when $h = 1$.

3.3 Fast attention

We implemented attention from [18]. Their work aims to reduce quadratic complexity to the input sequence length (because attention computes the dot-product at each pair of positions).

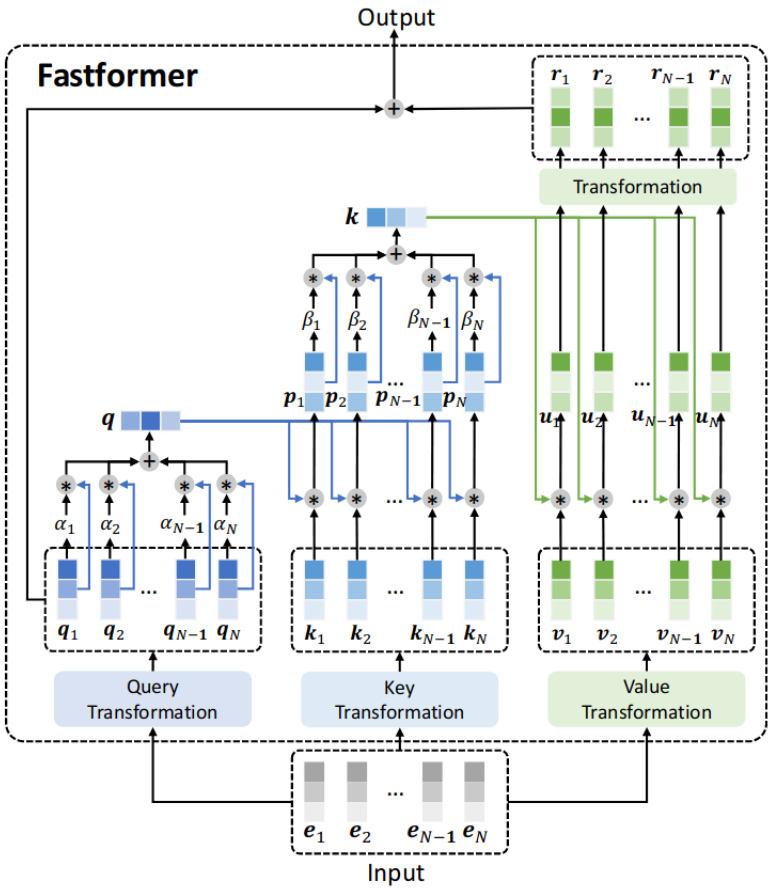

Figure 1: Attention head architecture from [18]

They use an additive attention mechanism to summarize information into global vectors, q and k as in figure 1. Next, the interaction between global query and keys, and global key and values are modeled by element-wise product. Lastly, context-aware values are linearly transformed and added with original query entries resulting in output. This approach greatly reduces complexity but also loses a lot of information. The speed increase is bigger the longer the sentence.

3.4 Word embeddings

We learn embeddings for each word from scratch. Embedding size plays a crucial role as it decides how much information can each word hold. However, there exists a point, after which increasing the dimension size isn't conveying more information or it's insignificant. To find the optimal value we perform ablation experiments, where we test different embedding

dimensions having only one attention head. Results from figure 3 suggest that a dimension of size 384 is most suitable, so later on this is the default value.

3.5 Positional embeddings

In language, a sequence is important and can decide the meaning of the sentence. The attention mechanism doesn't take this into account and produces the same output independently of the order of the words. Therefore we need to propagate information about positions to the model. We do that by forwarding each word position and projecting it into the embedding dimension. We allow the model to learn the projections. Then they are added to the word embeddings.

4 Experimental set-up

4.1 Dataset

Our dataset was divided into training, validation, and testing sets using a split ratio of 0.8, 0.2. The test set was further divided into validation and final test subsets. The training set consists of 21,366 examples, with 9,356 instances labeled as sarcastic and 12,010 instances labeled as non-sarcastic. The validation set comprises 2,669 examples, with 1,149 instances labeled as sarcastic and 1520 instances labeled as non-sarcastic. The final test set contains 2,671 examples, with 1,218 labeled as sarcastic and 1,453 labeled as non-sarcastic.

4.2 Hyperparameters

We considered several factors during the training process. We used a batch size of 256. The learning rate was set to 0.001, unless specified otherwise, to control the model's learning speed. To prevent overfitting, we applied weight decay with a value of 0.0001. We also experimented with the number of attention heads and network depth. For our binary classification task, we used binary cross entropy as the loss criterion.

4.3 Optimizer

We perform further experiments to find the best optimizer. Optimizers we test are: Adam[15], AdamW[16], Adadelta[17], SGD. We've found that we achieve the best results with SGD optimizer with a learning rate of 0.001. Because Adam and AdamW have high inertia their learning rate had to be lowered to 0.0001 or otherwise it was stuck, as shown in figure 12

4.4 Fine-tuning

Additionally, we fine-tune a few pre-trained popular transformed-based models and compare their results.

4.4.1 GPT-2

GPT-2 (Generative Pre-trained Transformer 2) [14] is a language model architecture that uses a multi-layer Transformer network. GPT-2 uses a deep stack of self-attention layers that allow it to efficiently process long-range dependencies and model contextual relationships between words or tokens.

The model uses an unsupervised learning method known as pre-training where it is trained on a huge text dataset. During initial training, the GPT-2 uses a masked language modeling objective where it learns to predict missing words in a sentence based on the surrounding context. During the pre-training phase, GPT-2 leverages a large-scale dataset comprised of diverse and unlabeled text from the internet, encompassing sources such as books, articles, and websites.

After initial training, the GPT-2 can be fine-tuned for specific end tasks using supervised learning. Tuning involves training the model on task-specific datasets, allowing it to adapt previously trained knowledge to the specific requirements of the target task.

We conducted fine-tuning experiments on GPT-2 on the *News headlines dataset*. The model was fine-tuned for 7 epochs with a learning rate of 2e-5 and the Adam optimizer. We achieved an accuracy of 85.3%, which was slightly higher than vanilla transformers' but lower than the baseline.

4.4.2 BERT + Classification head

BERT (Bidirectional Encoder Representations from Transformers)[4] is a pre-trained natural language processing model that revolutionized various language understanding tasks. Unlike previous models that processed text in a sequential manner, BERT introduced a bidirectional approach by training on both left and right context of each word, enabling it to capture deeper contextual meanings. BERT's architecture consists of multiple transformer layers that encode the input text and generate contextualized word embeddings. By pre-training on vast amounts of unlabeled data and then fine-tuning on specific downstream tasks, BERT exhibits exceptional performance in tasks like text classification, named entity recognition, question answering, and more. Its ability to comprehend complex sentence structures, handle ambiguity, and grasp contextual dependencies has made BERT a widely adopted model, advancing the field of natural language understanding and representation.

This model consists of two parts: BERT backbone and classification head. We use an enlarged version of BERT with 24 layers, 1024 hidden dimensions, 16 attention heads, and 336M parameters, which outputs $N + 1$, 1024-dimensional feature vectors, where N is the length of a tokenized utterance. As a classification head, we use a dropout layer with probability 0.1 and 2 linear layers without activation functions.

We fine-tuned this setup on *News headlines dataset* for 7 epochs with a learning rate of 2e-5, Adam optimizer, batch size of 32, and $80 - 20$ train/test split. We achieved a 94% accuracy.

4.4.3 Frozen BERT + Classification head

This model has the same parameters as the previous one, but BERT's weights were frozen so that we would train only the classification head and free some GPU resources.

We fine-tuned this setup on *News headlines dataset* for 15 epochs with a learning rate of 2e-5, Adam optimizer, batch size of 32, and $80-20$ train/test split. The highest accuracy that has been achieved is 67%, which is surprisingly smaller than expected.

4.4.4 DistilBERT + Classification head

DistilBERT[5] is a distilled version of the BERT model that aims to reduce its computational complexity while retaining a significant portion of its performance. By employing a process called knowledge distillation, DistilBERT compresses the knowledge learned by the larger BERT model into a smaller and more efficient architecture. This is achieved by training the smaller model to mimic the outputs of the larger model, enabling it to capture similar contextual representations. DistilBERT's reduced size and computational requirements make it more suitable for deployment in resource-constrained environments, such as mobile devices or edge devices, without sacrificing a considerable amount of performance. It serves as a valuable alternative for applications where efficiency and speed are critical, while still benefiting from the strong language understanding capabilities of the original BERT model.

DistilBERT has the same general architecture as BERT. The token-type embeddings and the pooler are removed while the number of layers is reduced by a factor of 2. In our case, it has 6 layers, 768 hidden dimensions, 12 attention heads, and 66M parameters. The classification head consists of a linear layer with ReLu, a dropout layer with a probability of 0.1, and another linear layer without an activation function.

We fine-tuned this setup on *News headlines dataset* for 15 epochs with a learning rate of 2e-5, Adam optimizer, batch size of 32, and $80-20$ train/test split. Training and validation time was 7 times smaller than with regular BERT and we achieved 85% accuracy.

5 Results

We conducted several experiments testing different attention types, depths, and sizes in the architecture of the transformer model and compared them to the published results.

Attention	Number of heads	Depth	Embedding size	Test accuracy
Default	10	10	400	**85.0%**
Default	6	6	384	84.3%
Default	6	8	384	84.2%
Default	1	6	384	83.9%
Wide	6	6	384	83.7%
Default	6	1	384	83.0%
Default	1	1	384	82.9%
Fast	1	6	384	80.3%
Wide	1	6	384	80.1%
Fast	6	6	384	80.1%

Table 1: Vanilla transformers comparison

Model	Test accuracy
Hybrid NN [2]	89.7%
BERT-GCN [7]	90.7%
fine-tuned GPT-2 (trained by us)	85.3%
fine-tuned BERT+Classification head (trained by us)	**94%**
fine-tuned Frozen BERT+Classification head (trained by us)	67%
fine-tuned DistilBERT+Classification head (trained by us)	85%
Best vanilla model (trained by us)	85%

6 Conclusions

The attention mechanism is well suited for text classification tasks. Simple architectures (in terms of the number of heads and depth) perform well despite their low complexity. Fast attention peformed the worst, which was expected, due to simplification of attention. Increasing the size of architecture allows to achieve slightly better results. Pretrained models perform significantly better than vanilla transformers. Further research could implement and test other attention types. Experiments could be conducted if pre-trained word embeddings improve the overall accuracy of the model.

References

[1] Silvio Amir et al. (2016), *Modelling Context with User Embeddings for Sarcasm Detection in Social Media.*
Available: https://arxiv.org/abs/1607.00976

[2] Rishabh Misra, Prahal Arora (2023), *Sarcasm detection using news headlines dataset.*
Available: https://www.sciencedirect.com/science/article/pii/S2666651023000013

[3] Ashish Vaswani et al. (2017), *Attention is all you need.*
Available: https://arxiv.org/abs/1706.03762

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova (2018), *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.*
Available: https://arxiv.org/abs/1810.04805

[5] Victor Sanh, Lysandre Debut, Julien Chaumond, Thomas Wolf (2019), *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.*
Available: https://arxiv.org/abs/1910.01108

[6] Reza Ghaeini, Xiaoli Z. Fern, Prasad Tadepalli (2018), *Attentional Multi-Reading Sarcasm Detection.*
Available: https://arxiv.org/abs/1809.03051

[7] Anuraj Mohan, Abhilash M Nair, Bhadra Jayakumar, Sanjay Muraleedharan (2023), *Sarcasm Detection Using Bidirectional Encoder Representations from Transformers and Graph Convolutional Networks.*
Available: https://www.sciencedirect.com/science/article/pii/S1877050922024991

[8] Mikhail Khodak, Nikunj Saunshi, Kiran Vodrahalli (2017), *A Large Self-Annotated Corpus for Sarcasm.*
Available: https://arxiv.org/abs/1704.05579

[9] Silviu Oprea, Walid Magdy (2020) *iSarcasm: A Dataset of Intended Sarcasm.*
Available: https://aclanthology.org/2020.acl-main.118/

[10] Shereen Oraby, Vrindavan Harrison, Lena Reed, Ernesto Hernandez, Ellen Riloff, Marilyn Walker (2017) *Creating and Characterizing a Diverse Corpus of Sarcasm in Dialogue.*
Available: https://arxiv.org/abs/1709.05404

[11] Rolandos Alexandros Potamias, Georgios Siolas, Andreas - Georgios Stafylopatis (2019) *A Transformer-based approach to Irony and Sarcasm detection.*
Available: https://arxiv.org/abs/1911.10401

[12] Amardeep Kumar, Vivek Anand (2020), *Transformers on Sarcasm Detection with Context.*
Available: https://aclanthology.org/2020.figlang-1.13/

[13] M. Abdullah, J. Khrais and S. Swedat (2022), *Transformer-Based Deep Learning for Sarcasm Detection with Imbalanced Dataset: Resampling Techniques with Downsampling and Augmentation.*
Available: https://ieeexplore.ieee.org/document/9811196

[14] Radford, Alec and Wu, Jeff and Child, Rewon and Luan, David and Amodei, Dario and Sutskever, Ilya (2019), *Language Models are Unsupervised Multitask Learners.*
Available: https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[15] Diederik P. Kingma, Jimmy Ba (2014), *Adam: A Method for Stochastic Optimization.*
Available: https://arxiv.org/abs/1412.6980

[16] Ilya Loshchilov, Frank Hutter (2017), *Decoupled Weight Decay Regularization.*
Available: https://arxiv.org/abs/1711.05101

[17] Matthew D. Zeiler (2012), *ADADELTA: An Adaptive Learning Rate Method.*
Available: https://arxiv.org/abs/1212.5701

[18] Chuhan Wu et al. (2021), *Fastformer: Additive Attention Can Be All You Need.*
Available: http://arxiv.org/abs/2108.09084

7 Appendix

7.1 Illustration of the attention mechanism

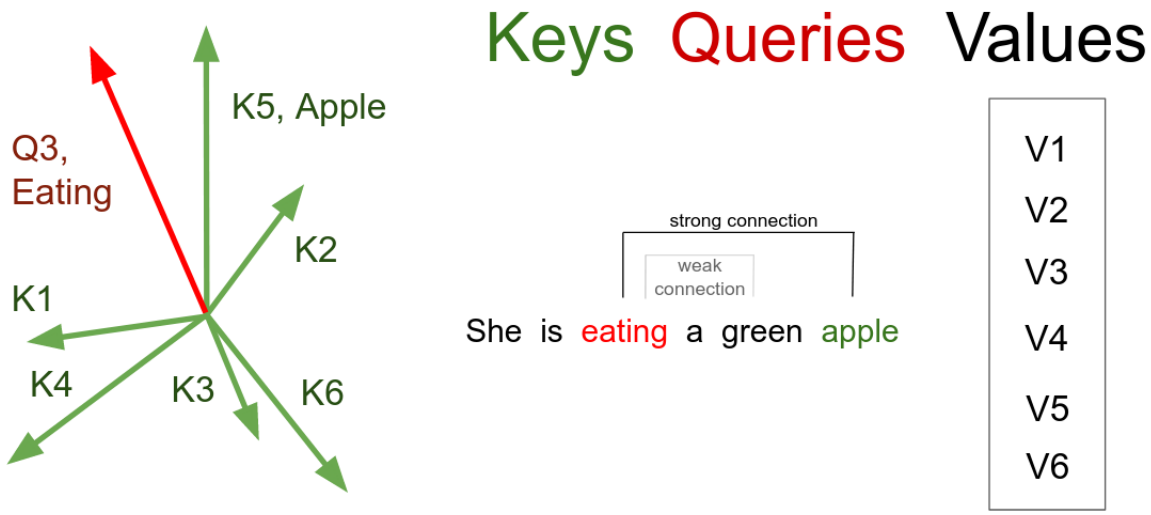

Figure 2: Attention mechanism

7.2 Accuracy of the model

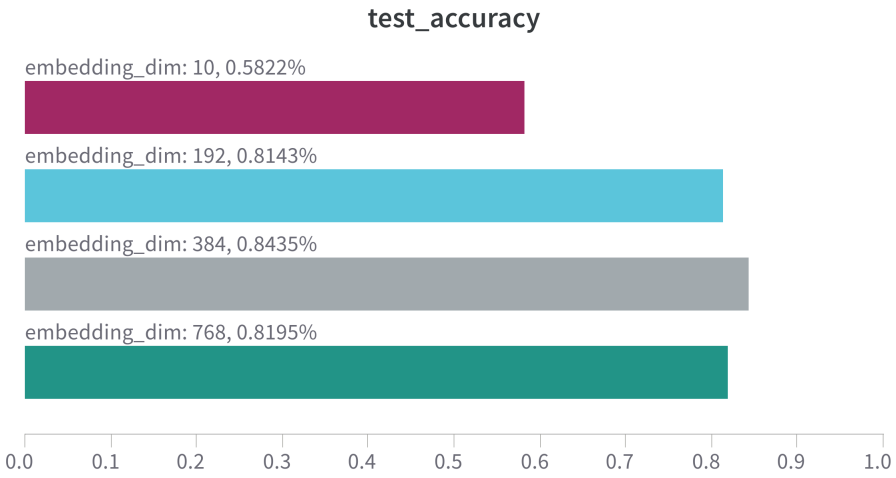

Figure 3: Accuracy of the vanilla transformer depending on the embedding dimensions

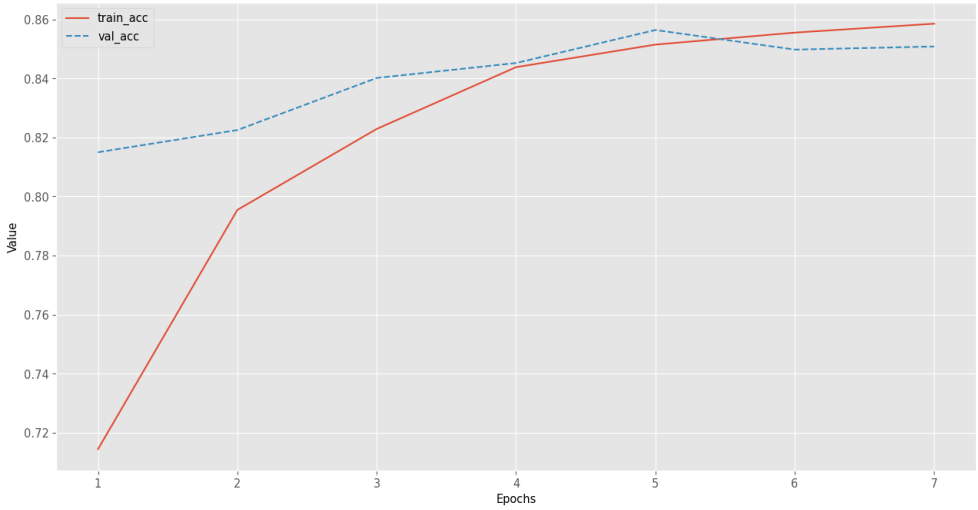

Figure 4: Training and testing accuracy of the fine-tuned GPT-2 over the 7 epochs of training

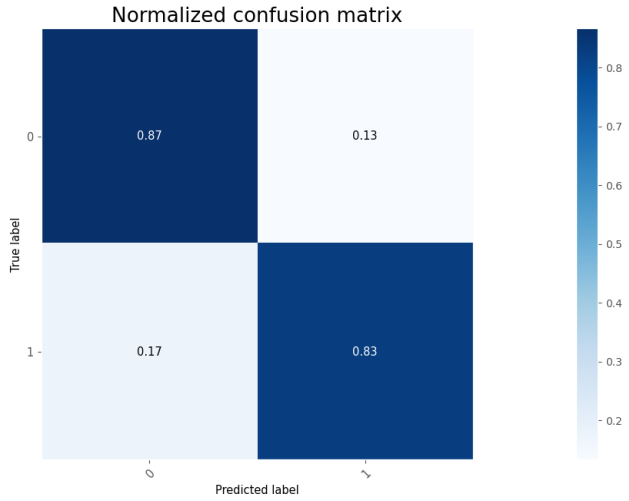

Figure 5: Normalized confusion matrix of the fine-tuned GPT-2 on the test dataset

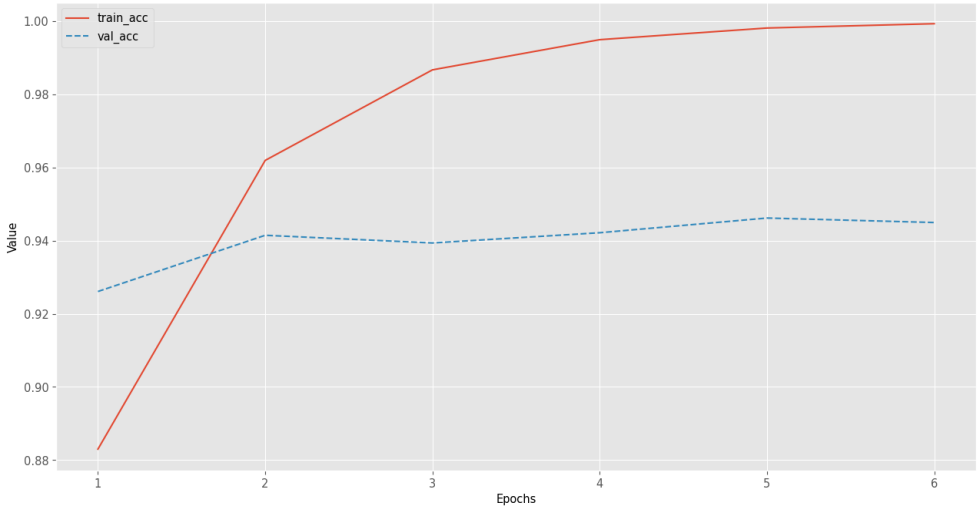

Figure 6: Training and testing accuracy of the fine-tuned BERT+classifcation head over the 7 epochs of training

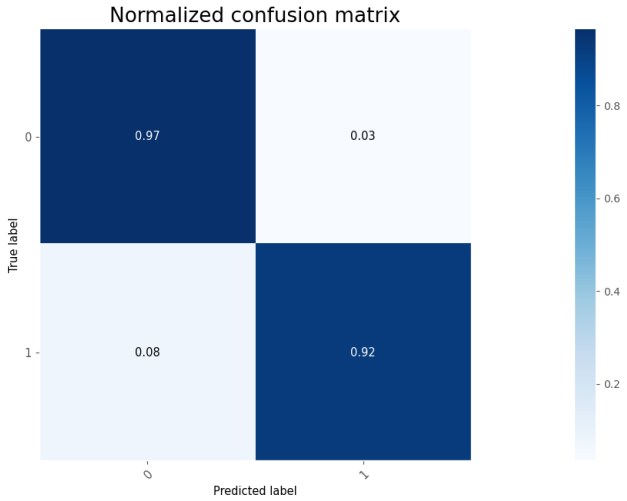

Figure 7: Normalized confusion matrix of the fine-tuned BERT+classification head on the test dataset

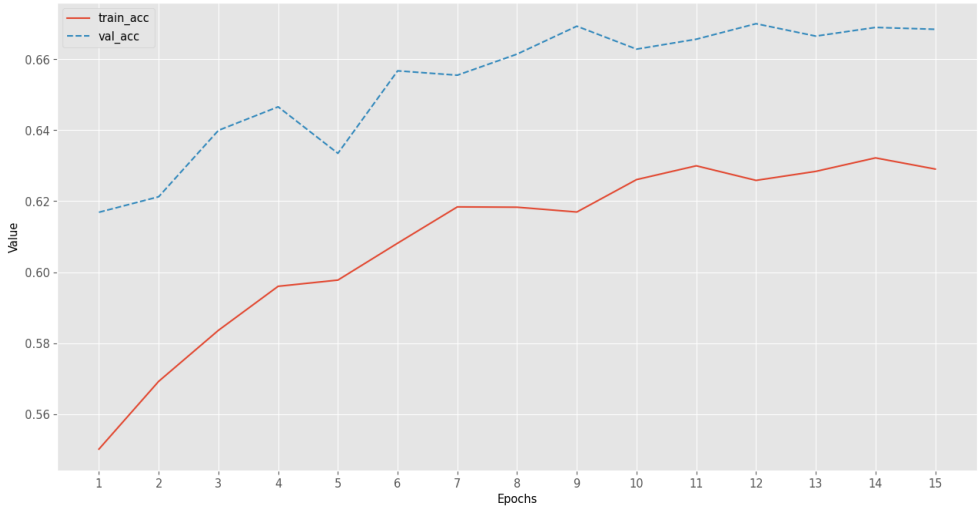

Figure 8: Training and testing accuracy of the fine-tuned Frozen BERT+classifcation head over the 15 epochs of training

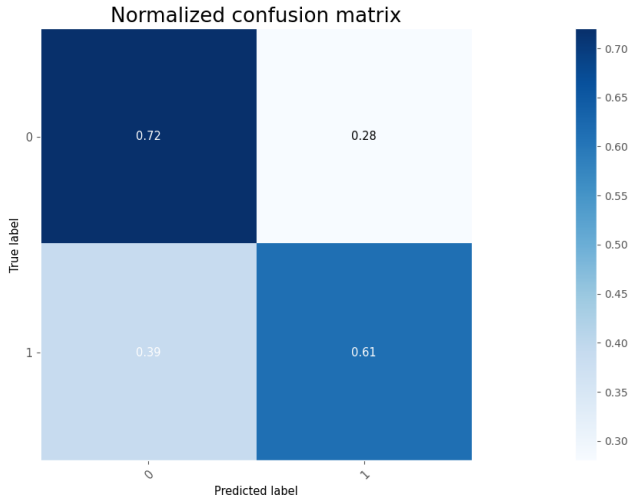

Figure 9: Normalized confusion matrix of the fine-tuned Frozen BERT+classification head on the test dataset

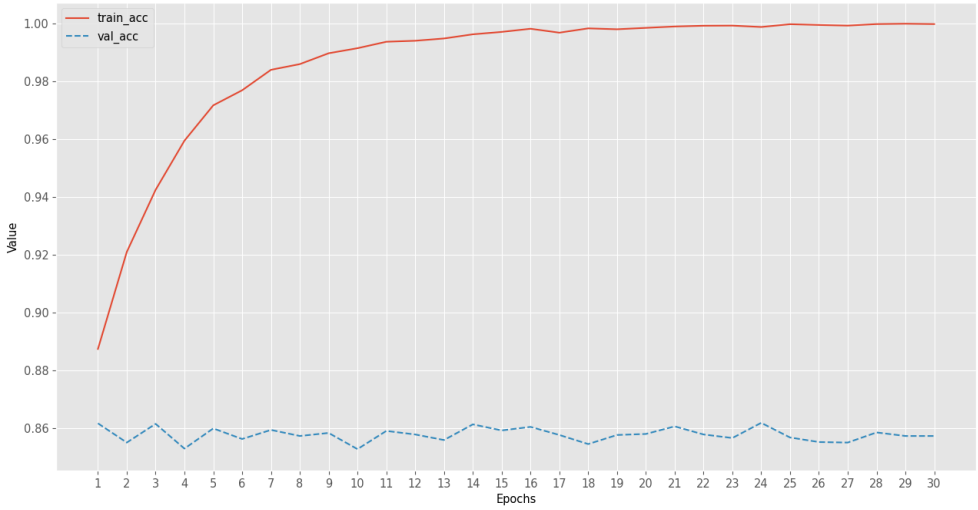

Figure 10: Training and testing accuracy of the fine-tuned DistillBERT+classifcation head over the 15 epochs of training

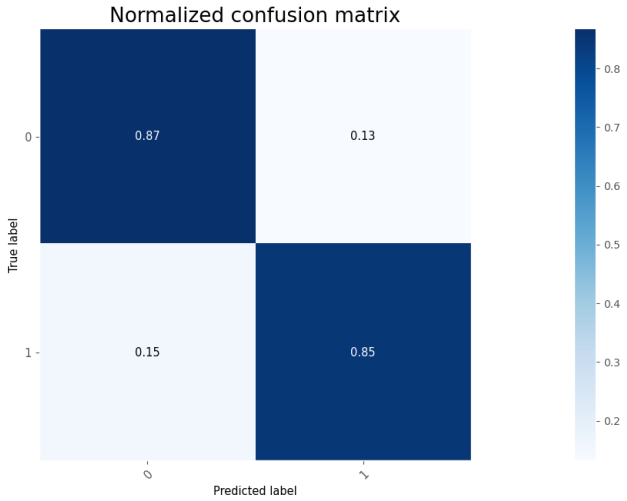

Figure 11: Normalized confusion matrix of the fine-tuned DistillBERT+classification head on the test dataset

7.3 Training loss

Figure 12: Training loss of the vanilla transformer with different optimizers