Is It Good Enough to Use Only the Output of [CLS] Token for Sentence Classification in BERT?

Kefan Wang
kefan.wang.18@ucl.ac.uk

Zhebin Sun
zhebin.sun.21@ucl.ac.uk

Zerui Li
zerui.li.21@ucl.ac.uk

Wangxuan Zhao
wangxuan.zhao.21@ucl.ac.uk

Abstract

BERT, one of the most powerful pre-trained models in the field of Natural Language Processing, is often considered to be the state-of-the-art. It has promoted a great deal of relevant research in this field since the beginning of its birth. At the moment, researchers tend to rely only on the output of [CLS], which is the first input token of BERT, for the task of sentence classification, without using the output of the other tokens. To verify whether it is good enough and whether simple modifications can lead to better results, we introduce two simple variants of BERT, namely BERT-ALL and BERT-QA. Through experiments on a sentiment analysis task, we found that BERT-QA outperforms plain BERT in a variety of aspects and it shows tremendous potential for sentence classification tasks. In contrast, BERT-ALL doesn't perform as well as plain BERT, and we propose several potential ways to improve it.

1 Introduction

Pre-trained models such as ELMO (Sarzynska-Wawer et al., 2021) and GPT (Brown et al., 2020) have shown impressive success in a wide range of NLP tasks (Nguyen et al., 2020). BERT (Devlin et al., 2018), stands for Bidirectional Encoder Representations from Transformers, is one of these pre-trained models and has achieved remarkable performance in language understanding. It is often considered as the state-of-the-art pre-trained language model. When using pre-trained models, the information used is different depending on the specific task, with some requiring word representations, such as the named entity recognition (NER) task, while for some other tasks such as classification and sentence semantic matching, sentence representations will be more important. [CLS], the first token in BERT, is designed to aggregate information from the input sentence(-pair) (Liu, 2019), and its output is usually considered to be a representative vector of the entire input. When doing sentence classification, researchers tend to just take the output of [CLS] token and fine-tune the entire BERT based on the classification performance, such as the works carried out by Munikar et al. (2019); Adhikari et al. (2019); Lee and Hsiang (2019). However, are the results obtained in this way good enough?

In this paper, we investigate whether BERT's sentence classification capability can be improved simply. Note that the term "sentence" in "sentence classification" throughout this paper refers to any original text input that produces only one [SEP] token after BERT tokenisation, so it can refer to a word, a sentence, a paragraph or even a passage, as long as it is not paired (i.e. generates two [SEP] tokens after BERT tokenisation). To achieve this, we introduce two simple variants of BERT for sentence classification, BERT-ALL and BERT-QA. BERT-ALL utilises the output from all of the input tokens for sentence classification, rather than only the output of [CLS] token. BERT-QA converts the original classification task into a Question-Answering system, which selects the candidate answers with the highest probability. We aim to show that both methods can outperform the plain BERT in classification tasks. By experimenting with a sentiment analysis task, a well-known branch of sentence classification, we show that BERT-ALL cannot beat plain BERT, and discuss potential problems and improvements. However, we find that BERT-QA can outperform plain BERT in many aspects, even a partially fine-tuned one can beat a completely fine-tuned plain BERT in some aspects. Moreover, we show that the intuition behind BERT-QA is very robust, and BERT-QA with carefully selected candidate answers has the potential to boost performance greatly.

2 Related Works

Over the years, neural-based models have achieved impressive performance in sentence-level tasks such as text classification and Machine Translation. The basic structure of these models is to transform each word in a sentence into a word vector and use these vector representations as input to the model to extract sentence-level features (WANG et al., 2020). Word embedding mechanisms such as Word2Vec use low-dimensional distributed representations instead of the traditional highly sparse one-hot vectors. Sentence-level embedding is then typically obtained using convolutional or recurrent networks. For example, Melamud et al. (2016) used a bi-directional LSTM based on Word2Vec-generated word vectors to efficiently obtain generic context embedding. Recently, with the popularity of attention mechanisms, lots of research on sentence-level embedding have been based on this technique combined with the previously mentioned models. For example, Jang et al. (2020) proposed an attention-based Bi-LSTM-CNN hybrid model for text classification.

BERT (Devlin et al., 2018) not only brings high-quality pre-trained model weights but also a more representative word embedding than those previously generated by traditional algorithms such as Word2Vec. The special classification token [CLS] aggregates sequence information for next sentence prediction (NSP) (Devlin et al., 2018; Choi et al., 2021) during pre-training. Hence, this token is often used as the sentence-level vector representation for tasks that require sentence information nowadays. Although using only the output of [CLS] token for text classification tasks, lots of methods and strategies have been proposed to effectively fine-tune BERT (Sun et al., 2019b). However, Choi et al. (2021) has shown that many other methods of sentence-level embedding such as Pooled token embeddings, Sentence-BERT and sentence BERT with CNN architecture outperformed the plain BERT in semantic textual similarity task, and it has also suggested that average pooling the token embedding outputs can form a fixed-length sentence vector. Su and Vijay-Shanker (2020) proposed BERT-variants using all output vectors. In their first model, the output vectors are processed through a Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and concatenated with the output of [CLS] token to obtain a sentence-level representation. In their second model, they did the same thing but replaced LSTM with an attention mechanism (Vaswani et al., 2017). Both models outperform plain BERT in their experiments, where the one with attention achieved the best results. In our work, the BERT-ALL model is developed under a similar idea as Choi et al. (2021); Su and Vijay-Shanker (2020), but is not as complex as their processing, since the focus of our study is on "simple variants of plain BERT".

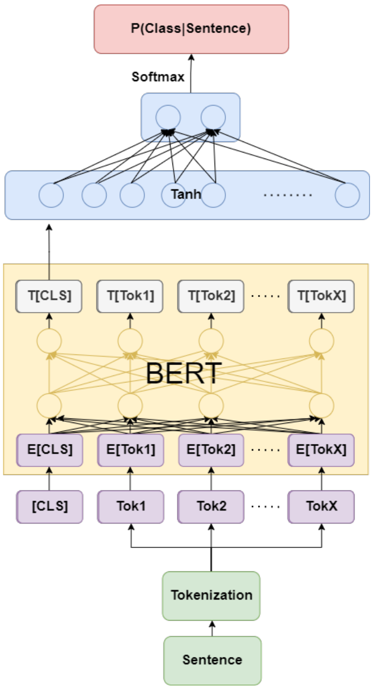

Figure 1: Architecture of BERT-Plain

Sun et al. (2019a) presented an interesting method for aspect-based sentiment analysis (ABSA). They transformed the ABSA question into a sentence-pair classification task using auxiliary sentences and achieved the state-of-the-art result by fine-tuning BERT. Our BERT-QA model is similar to this, where it transforms the single sentence classification task into a Question-Answering system by adding auxiliary candidate answers. Devlin et al. (2018) demonstrated that the pre-trained BERT model has impressive improvement on QA and Natural Language Inference (NLI) tasks.

3 Methods

3.1 BERT-Plain

BERT is a pre-trained language representation model based on the Transformer architecture (Vaswani et al., 2017). It combines several Transformer Encoder layers as its structure and emphasises a state-of-the-art pre-training approach to produce in-depth bi-directional language representations that goes beyond the traditional uni-directional language models and the shallow bi-

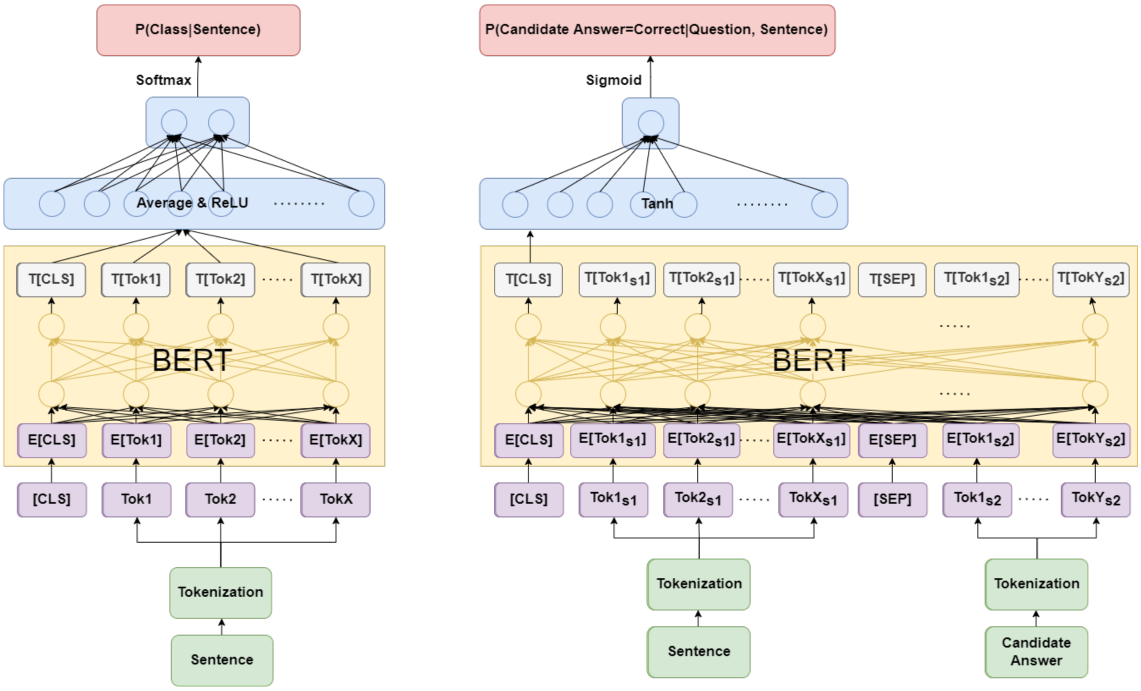

Figure 2: Architectures of BERT-ALL (Left) and BERT-QA (Right)

directional language models. It is pre-trained via a Masked Language Model (MLM) task and a Next Sentence Prediction (NSP) task on the BooksCorpus (800M words) (Zhu et al., 2015) and English Wikipedia (2,500M words). By simply adding an additional output layer to the pre-trained BERT and fine-tuning the entire BERT with a small number of epochs, state-of-the-art performance was achieved in a variety of downstream tasks at the inception of BERT.

We make use of the "bert-base-uncased" version of BERT implemented by HuggingFace[1] (Wolf et al., 2020) in our work. In the implementation, the output of [CLS] token is passed to a pooler layer, which is a linear layer with a tanh activation function[2]. For classification problems, the output of the pooler layer is then passed to another linear layer with a softmax activation function. The architecture of BERT used for classification is shown in Figure 1. We refer to this version of BERT as BERT-Plain and it is treated as the baseline. Both BERT-ALL and BERT-QA are built upon this model.

3.2 BERT-ALL

Due to the high complexity of the structure of BERT and the huge size of the corpus used for pre-training. We suspect that the output of all tokens in BERT may be more representative than the output of only [CLS] token in classification tasks. BERT-ALL is developed to investigate whether the former can perform better, the architecture is shown on the left of Figure 2.

Denote D_i^m as the output vector of token i, $D^{n \times m}$ as the output matrix consisting of the output vectors of all tokens, where n is the number of tokens and m is the dimension of each output vector. In BERT-ALL, $D^{n \times m}$ is averaged across the dimension n, resulting in the new sentence-level representative vector D_{Avg}^m. A Rectified Linear Unit (ReLU) (Agarap, 2018) activation function is then applied to it for adding non-linearity and aiming to generate better results. Afterwards, it is connected to a linear layer $W^{c \times m}$, followed by a softmax activation function, where c is the number of classes. By denoting Avg as the function for averaging out dimension n, the process of BERT-ALL can be represented as the following formula.

$$out = Softmax(W^{c \times m}(ReLU(Avg(D^{n \times m}))))$$

3.3 BERT-QA

One of the BERT pre-training tasks, Next Sentence Prediction, is a sentence-pair classification task.

[1] https://huggingface.co/bert-base-uncased

[2] https://huggingface.co/docs/transformers/main/en/main_classes/output

We hypothesise that it may be beneficial to convert single sentence classification tasks to sentence-pair classification tasks, as it mimics the original pre-training process. BERT-QA is developed to verify this hypothesis. The architecture of BERT-QA is shown on the right of Figure 2.

[CLS] x [SEP]	z
[CLS] I'm very happy. [SEP]	1
[CLS] The film isn't good. [SEP]	0

Table 1: Examples BERT inputs of sentiment analysis task

[CLS] x [SEP] $\mathbf{y}$ [SEP]	$\mathbf{g}$
[CLS] I'm very happy. [SEP] True [SEP]	1
[CLS] I'm very happy. [SEP] False [SEP]	0
[CLS] I'm not happy. [SEP] True [SEP]	0
[CLS] I'm not happy. [SEP] False [SEP]	1

Table 2: Examples of converting the original inputs in Table 1 to BERT-QA inputs, where BERT-QA implicitly asks "Does sentence x show positive emotion?"

BERT-QA works as a Question-Answering (QA) system, as revealed by its name. For a sentence x, BERT-QA asks a question about x implicitly and picks the candidate answer with the highest probability. For a sentence x and its original label z, we introduce a set of candidate answers $\mathbf{y}$ and generate new labels $\mathbf{g}$ accordingly, where g_i indicates whether y_i is the correct answer to the question posed by BERT-QA. Consider a sentiment analysis task, where 1 is defined as positive sentiment and 0 is defined as negative sentiment. Examples of conversions from the original input into a BERT-QA input can be seen in Table 1 and Table 2. This can be interpreted as BERT-QA implicitly asking "Does sentence x show positive emotion?", for $x =$ "I'm very happy.", the candidate answer "True" is correct and the candidate answer "False" is incorrect. Similarly, for $x =$ "I'm not happy." the opposite $\mathbf{g}$ are the correct answers.

The architecture of BERT-QA is similar to that of BERT-Plain. Instead of building the last layer with c (i.e. the number of classes) hidden neurons and a softmax activation function, BERT-QA constructs the last layer with one hidden neuron and a sigmoid activation function. The output of BERT-QA has a nice interpretation, namely

$$P(y_i = Correct | Question, Sentence)$$

4 Experiments

4.1 Dataset

The Standford Stanford Sentiment Treebank v2 (SST2) (Socher et al., 2013) is employed for comparing the classification power of BERT-Plain, BERT-ALL and BERT-QA. The task, sentiment analysis, is a computational study of people's views, emotions and attitudes towards objects such as products, services, and topics and their attributes (Zhang et al., 2018). This dataset contains mainly long-form movie reviews with binary labels as positive and negative comments. We utilise a pre-processed subset[3] of it, which has a total number $10,000$ reviews. In this pre-processed subset, 0 represents positive sentiment and 1 represents negative sentiment. As the dataset has already been shuffled, it is sequentially split into training/validation/test sets at a ratio of 60%/20%/20%.

4.2 BERT-QA Candidate Answers Selection and Input Conversion

In order to conduct BERT-QA, the question posed by BERT-QA needs to be proposed and a set of candidate answers needs to be identified accordingly. We experiment with two different sets of candidate answers to BERT-QA to find out whether the choice of them will have a significant impact.

In the first version, we let BERT-QA ask "Is sentence x a negative emotional sentence?". To do this, we simply converting every x with $z = 0$ into x with $y_1 = True, g_1 = 0$ and x with $y_2 = False, g_2 = 1$. g_1 and g_2 are exchanged for every x with $z = 1$. In the second version, we let BERT-QA ask a potentially complex question, "What is the opposite of the sentiment of sentence x?". Similar as before, every x with $z = 0$ is converted into x with $y_1 = Positive, g_1 = 0$ and x with $y_2 = Negative, g_2 = 1$. g_1 and g_2 are exchanged for every x with $z = 1$. In later sections, We refer to the first version as BERT-QA-TF and the second as BERT-QA-PN.

At training time, the conversion of the inputs is done by directly transforming the dataset according to the rules described in the previous paragraph. At test time, the original data is used and the conversion is completed in RAM for each mini-batch. The candidate answers with the highest probability is selected and the corresponding z is returned as the predicted class. For example, in BERT-QA-TF,

[3] https://github.com/yyxx1997/pytorch/blob/master/bert-sst2/sst2_shuffled.tsv

when $y = True$ (i.e. BERT-QA-TF says "Sentence x is a negative emotional sentence."), the corresponding $z = 1$.

4.3 Configuration and Model Training

To be compatible with the embedding layer of the pre-trained BERT model, we use the same WordPiece-based tokeniser (Wu et al., 2016) as in the original BERT. We use the "bert-base-uncased" version of BertTokeniser implemented by HuggingFace[4] (Wolf et al., 2020).

In general, BERT-Plain and BERT-ALL solve multi-class classification problems, while BERT-QA solves binary classification problems. As we wish to minimise the predicted distribution and the target distribution of the labels, Cross-Entropy Loss is employed in all of our model training.

All model training consists of two phases, namely training a new classifier and fine-tuning the original BERT model. Therefore, in order to allow the classifier to learn sufficiently and to prevent excessive corruption of the original parameters in the pre-trained BERT, we utilise two different Adam optimisers with $\beta_1 = 0.9, \beta_2 = 0.999, eps = 1e^{-8}$, and set $lr_{BERT} = 2e^{-5}, lr_{classifier} = 1e^{-3}$ differently, where Adam with lr_{BERT} optimises the entire BERT and Adam with $lr_{classifier}$ optimises the classifier.

Furthermore, in order to find out if our models are strong enough to require fine-tuning only a part of BERT, we experiment with two versions of fine-tuning, one called complete fine-tuning and the other called partial fine-tuning. In complete fine-tuning, the whole BERT is fine-tuned. In partial fine-tuning, only the output layer and the pooler layer of BERT are fine-tuned. In later contexts, we denote -C as completely fine-tuned models and -P as partially fine-tuned models.

We train each model endlessly until it overfits the training data. Specifically, for every model, both training loss and validation loss are evaluated at each epoch, and the model is only saved if it reaches the lowest validation loss to prevent it from overfitting the training set. Each model is initially trained for 10 epochs, and the checkpoint is saved at that point. If the model hasn't overfit for 5 consecutive epochs, training will be resumed and the model will be trained for another 10 epochs. The process is repeated until the model overfits.

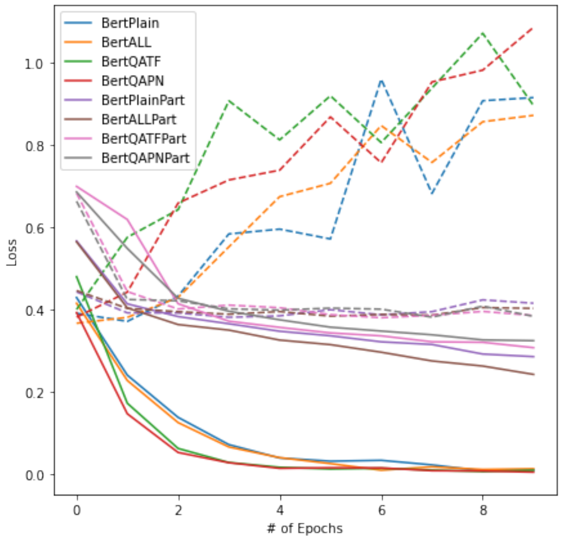

Figure 3: Loss graph. Solid lines represent training loss while dashed lines represent validation loss.

5 Results

5.1 Qualitative Analysis of Loss Graph

Figure 3 shows the training loss and validation loss of each model during its training phase. All models overfit the training data within the first 10 epochs. The best models (i.e. models with the lowest validation loss) are saved and used for evaluation. It is obvious that the training loss of the completely fine-tuned models drops sharply (close to zero) as the number of epochs increases, but the validation loss increases to around 0.8, implying that these four models achieve severe overfitting during training. As a result, only the very first few epochs (about 1 to 2) are the exact useful epochs when training these models. In contrast, the loss of the partially fine-tuned ones steadily decreases and the models start to overfit after around 5 epochs, indicating that better generalisation capabilities can be expected in these models.

5.2 Quantative Analysis of Model Evaluation

Table 3 summarises the evaluation of these models. The results indicate that BERT-QA-PN-C achieves the best performance in terms of precision, accuracy and F1 score while maintaining a similar recall to BERT-Plain-C. It's also worth noting that BERT-QA-TF-C also achieves better performance than BERT-Plain-C in those aspects, with an even closer recall. However, it is clear that BERT-QA—TF-C performs much worse than BERT-QA-PN-C in the other three aspects, but it still achieve the second highest accuracy and F1 Score. Unexpectedly,

[4] https://huggingface.co/docs/transformers/model_doc/bert#transformers.BertTokenizerFast

Model	Precision	Recall	Accuracy	F1 Score
BERT-Plain-C	0.852	**0.884**	0.870	0.868
BERT-ALL-C	0.818	0.882	0.854	0.849
BERT-QA-TF-C	0.869	*0.883*	*0.877*	*0.876*
BERT-QA-PN-C	**0.887**	0.880	**0.883**	**0.883**
BERT-Plain-P	0.828	0.851	0.841	0.839
BERT-ALL-P	0.835	0.865	0.853	0.850
BERT-QA-TF-P	0.853	0.845	0.849	0.849
BERT-QA-PN-P	*0.877*	0.852	0.863	0.864

Table 3: Evaluation of the models. The largest value of each metric is bolded, the second largest value of each metric is italicised.

BERT-ALL-C doesn't perform as well as BERT-QA-C, it fails to beat BERT-Plain-C in any aspect. Although BERT-Plain-C doesn't perform well in precision, accuracy and F1 score, it still achieves the highest recall. By comparing the partially fine-tuned models, all of our models outperform BERT-Plain-P, which is a good result as if one wants to partially fine-tune BERT-Plain, one can expect better results on classification tasks by easily extending BERT-Plain to BERT-ALL or BERT-QA. Surprisingly, BERT-QA-PN-P achieves the second-highest precision even compared to the completely fine-tuned ones, the only one that achieves a better precision than it is its completely fine-tuned version, BERT-QA-PN-C.

5.3 Failure of BERT-ALL and Potential Improvement Methods

Undesirably, BERT-ALL fails in all aspects. It was hypothesised at the beginning that utilising all of the output might lead to better results than using only the output of [CLS] token when doing sentence classification, but the result shows the opposite. Also, BERT-Plain-C still holds the highest recall. It is therefore likely that BERT-Plain is not so bad and the output of [CLS] token represents better sentence information than the output of all tokens.

However, this does not necessarily mean that the hypothesis is wrong. It is still possible that the architecture of BERT-ALL is not good enough. One possible improvement would be to change the choice of activation function after averaging. ReLU activation function is used in our case for non-linearity and a more easily computed gradient. One might try to use the GELU activation function as inside the BERT architecture, or try using the tanh activation function as to how the output of [CLS] token is processed in HuggingFace's implementation. Another potential improvement would be to change the way of summarising sentence information. By simply making an average, information from smaller output may be lost, while information from larger output may be enhanced. For a shorter sentence, when predicting or training with longer ones (i.e., they are processed in the same mini-batch), the shorter sentence will have a lot of padding tokens. Thus, after averaging, the vector representation may be dominated by the output of the padding tokens. For tackling this, one might discard the padding outputs when doing the average. On the other hand, one might try to use a Recurrent Neural Network based architecture (Medsker and Jain, 2001) or an Attention mechanism (Vaswani et al., 2017; Niu et al., 2021) to summarise the output, as Su and Vijay-Shanker (2020) did in their work. However, the idea of "simple variants of plain BERT" might be violated.

5.4 Deeper Analysis of BERT-QA

Although BERT-ALL fails in both completely fine-tuned and partially fine-tuned cases, BERT-QA gives surprising and insightful results. Two plausible explanations of why BERT-QA works can be proposed. One is that, as previously hypothesised, BERT-QA mimics the original pre-training process. Therefore, better results can be expected. The other is that its mechanism can be seen as a way of data augmentation. When doing a k-class classification with k candidate answers, the dataset can be seen to be expanded k times. In addition, when doing a k-class classification, the number of candidate answers is not limited to k and one can give more correct candidate answers to a class. Theoretically, the dataset can be expanded to be as large as possi-

Sentence	Label	BERT-Plain	BERT-ALL	BERT-QA-TF	BERT-QA-PN
1	0	1	1	0	0
2	1	0	0	0	0
3	1	1	0	0	0
4	0	1	1	1	1
5	1	0	0	0	1
6	0	1	1	0	1
7	0	1	0	0	0
8	0	1	0	1	1

Table 4: Example sentences with model predictions. 1: The movie addresses a hungry need for pg-rated, nonthreatening family movies, but it doesn't go too much further. 2: You can thank me for this. I saw Juwanna Mann so you don't have to. 3: Well, it does go on forever. 4: It's hard to imagine Alan Arkin being better than he is in this performance. 5: An inexperienced director, Mehta has much to learn. 6: Rarely, a movie is more than a movie. Go. 7: You might not buy the ideas. But you'll definitely want the t-shirt. 8: Exhilarating but blatantly biased.

ble. In this case, two interpretations of BERT-QA may be reasonable. One is that BERT-QA asks several questions at the same time, it determines the most important question for the current sentence x and selects the most appropriate candidate answer for that question, where the two steps are performed in parallel. The second is that several candidate answers are valid for answering the question posted by BERT-QA. These findings suggest that BERT-QA has the potential to augment the dataset to a very large case. As common knowledge in the machine learning community, models trained with more data tend to lead to better results. Hence, it could conceivably be hypothesised that BERT-QA can generate supreme results with reasonably large size of candidate answers.

The results of BERT-QA-TF and BERT-QA-PN show that the choice of predefined candidate answers is important. By replacing True/False candidate answers with Positive/Negative candidate answers, the performance of BERT-QA improves in most aspects, implying that Positive/Negative candidate answers are superior to True/False candidate answers. It is hypothesised that this difference is raised by the variation in their word embedding space. In general, candidate answers are hyperparameters and BERT-QA shows the potential to improve the model performance to a higher extent when candidate answers are tuned and selected carefully.

5.5 Qualitative Analysis of Completely Fine-Tuned Models

In this section we attempt to identify the common features behind these models, and the patterns that one algorithm may perfer over another. We assume that similar pattern recognition exists for the partially fine-tuned models as for the completely fine-tuned ones. Therefore, we only use the completely fine-tuned versions to give comparison. Eight sentences in the test set with correct labels and predictions from each model are given in Table 4. The full list can be found here[5].

The first property we can notice is that all the models seem to be "naive". In sentence 2, with the exception of the word "don't", few words show negative sentiment. However, the whole sentence shows the implication that that people don't have to watch this film, i.e. the reviewer does not recommend it. Similarly, in sentence 4, although negative word like "hard" exists, the reviewer states that Alan Arkin does a good job. Models seem to have difficulty understanding sentences with deep meaning, such as turning phrases in sentence 2 and 4.

In sentence 8, only BERT-ALL classifies it correctly. Since in that sentence the length is short and the words are rare. It is hypothesised that BERT-ALL is likely to achieve better results on sentences with rare words or shorter length.

It seems that BERT-Plain treats negative emotional words with more weights, so when words like "don't, no, rare" present in the sentence, BERT-Plain has a high probability of classifying that sentence as negative.

The predictions given by BERT-QA-TF and BERT-QA-PN seem to be close to each other. They seem to classify longer sentences better than the other 2, as for sentence 1. In addition, only

[5] https://www.dropbox.com/s/ygo8hk889gjh342/pred.csv?dl=0

BERT-QA-PN correctly classifies sentence 5 and only BERT-QA-TF correctly classifies sentence 6. These two sentences can be considered as relatively difficult cases, failing in each of the remaining 3 algorithms. Thus, it could be argued that BERT-QA with different candidate answers tend to give similar predictions for easier problems, but their pattern recognition may be different and specific in some cases, of which the cases remain to be analysed in depth.

6 Conclusion

In this work, we investigate whether the performance of sentence classification with BERT can be improved in a simple way by introducing BERT-ALL and BERT-QA. Our results show that a completely fine-tuned BERT-QA outperforms BERT-Plain in terms of precision, accuracy and F1 score. Even a partially fine-tuned one outperforms completely fine-tuned BERT-Plain in terms of precision. The main reason for achieving this may be the nature of data augmentation behind BERT-QA. It is very exciting that such a large performance improvement can be achieved by making such a simple change to the original BERT. One can expect better results on their classification tasks by casting BERT-Plain as BERT-QA. On the other hand, from the failure of BERT-ALL, we can see not only the potential sub-optimal design of it, but also the performance of BERT-Plain is not so bad. We also show the potential patterns that each algorithm prefers to identify. It appears that BERT-ALL performs better with short sentence and rare words, BERT-Plain gives negative words higher weights, and BERT-QA tends to give similar predictions with different candidate answers, but each set of candidate answers can form a different sentence specific pattern recognition rule. However, all models fails to recognise hard cases such as sentences with turning phrases.

In the future, one could modify BERT-ALL according to the proposed approaches discussed in Section 5.3 and investigate whether its poor performance in this experiment is due to a structural flaw. On the other hand, only a limited number of candidate answers have been experimented with BERT-QA in this work, in particular True/False and Positive/Negative. One can experiment with different choices, such as awful/nice, annoy/satisfy, etc. Furthermore, since both of them are 1-gram candidate answers, one can experiment with n-gram candidate answers by letting BERT-QA ask more complex questions, e.g. the candidate answers could be sentences or a mix of sentences and words. In addition, the data augmentation nature behind BERT-QA should be well studied, possibly by adding multiple correct candidate answers to a question, thus expanding the dataset more rapidly. Lastly, a more in-depth study on pattern identification should be studied, as only 8 sentences is analysed in our work.

References

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. Docbert: BERT for document classification. *CoRR*, abs/1904.08398.

Abien Fred Agarap. 2018. Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Hyunjin Choi, Judong Kim, Seongho Joe, and Youngjune Gwon. 2021. Evaluation of bert and albert sentence embedding performance on downstream nlp tasks.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

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

Beakcheol Jang, Myeonghwi Kim, Gaspard Harerimana, Sang-ug Kang, and Jong Wook Kim. 2020. Bi-lstm model to increase accuracy in text classification: Combining word2vec cnn and attention mechanism. *Applied Sciences*, 10(17).

Jieh-Sheng Lee and Jieh Hsiang. 2019. Patentbert: Patent classification with fine-tuning a pre-trained BERT model. *CoRR*, abs/1906.02124.

Yang Liu. 2019. Fine-tune bert for extractive summarization. *arXiv preprint arXiv:1903.10318*.

Larry R Medsker and LC Jain. 2001. Recurrent neural networks. *Design and Applications*, 5:64–67.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany. Association for Computational Linguistics.

Manish Munikar, Sushil Shakya, and Aakash Shrestha. 2019. Fine-grained sentiment classification using bert. In *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, volume 1, pages 1–5.

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. Bertweet: A pre-trained language model for english tweets. *arXiv preprint arXiv:2005.10200*.

Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. 2021. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62.

Justyna Sarzynska-Wawer, Aleksander Wawer, Aleksandra Pawlak, Julia Szymanowska, Izabela Stefaniak, Michal Jarkiewicz, and Lukasz Okruszek. 2021. Detecting formal thought disorder by deep contextualized word representations. *Psychiatry Research*, 304:114135.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Peng Su and K. Vijay-Shanker. 2020. Investigation of bert model on biomedical relation extraction based on revised fine-tuning mechanism.

Chi Sun, Luyao Huang, and Xipeng Qiu. 2019a. Utilizing bert for aspect-based sentiment analysis via constructing auxiliary sentence. *arXiv preprint arXiv:1903.09588*.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019b. How to fine-tune bert for text classification?

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

ZIWEN WANG, HAIMING WU, HAN LIU, and QIAN-HUA CAI. 2020. Bert-pair-networks for sentiment classification. In *2020 International Conference on Machine Learning and Cybernetics (ICMLC)*, pages 273–278.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27.