BERT-Chatbot

Sofoklis Kyriakopoulos
Willem Van de Mierop

CITY UNIVERSITY
LONDON

Contents

1 Introduction

In this paper we are going to fine-tune pretrained BERT (Bidirectional Encoder Representations from Transformers) models on the Cornell Movie-Dialogs Corpus dataset so that it can generate answers to the specific questions that are asked to the model (Devlin et al. 2018; Danescu-Niculescu-Mizil and Lee 2011). The objective is to make the questions and generated answers in a conversational structure so that the models can be combined to be used as a chatbot. We will fine-tune the original BERT model as well as SciBERT. By combining these models, we achieve a conversational model that is able to answer with appropriate intelligence depending on the question given.

One key difference that the BERT model has compared to other competitive NLP models such as GPT-2 is the fact that BERT is a Transformer (Vaswani et al. 2017) based model which is bi-directional, meaning that when given text, it predicts words considering both the context before and after the word. While the bi-directionality showed to provide an exceptional base for language classification and question answering tasks, it becomes difficult to use BERT for causal text generation, something vital for a conversational model. To overcome this, we adapt the training regime to the constraints of the BERT model, and utilize a unique generating algorithm that overcomes the need for uni-directionality. The code used for this report can be accessed at https://github.com/willemvdmierop/Chatbot_bert.

2 Dataset

The Cornell Movie-Dialogs Corpus database (Danescu-Niculescu-Mizil and Lee 2011) consists of $220,579$ conversational exchanges over $10,292$ dialogues between characters from movie scripts. The database was published in 2011 to signify the way that conversational participants tend to adapt their language and phrasing to each other. This provides an essential source of natural language text to which a conversational NLP model can be trained on. While the database holds many attributes and labeled data, the key components used for this report are the contents of the dialogues themselves, parsed from the files *movie_lines.txt* and the *movie_conversations.txt*. The *movie_lines.txt* file contains the line-id, the character-id, the movie-id, the character-name and the line text. The *movie_conversations.txt* contains the different line-ids of text from a particular conversation as a sequence of dialogue lines.

From the parsed text, we construct a dataset of phrase pairs from all the phrases and the responses produced in a dialogue. For example, if a dialogue has four phrases *A,B,C,D*, then the phrase pairs that are added to our dataset are *(A,B), (B,C),* and *(C,D)*. This is done for all dialogues in the Cornell database, with the exception of phrase pairs where either phrase is longer than 20 words. This was done to limit the pairs to a maximum length of 40 (necessary for the training process) without splitting the phrases, and potentially losing language context and nuances. This process resulted in a dataset of $171,022$ pairs.

3 Model

3.1 BERT

The BERT model, as proposed by Devlin et al. (2018), is a bi-directional transformer based natural language processing model. Specifically, the architecture of the BERT model used for this paper (BERT_{BASE}) consists of 12 Transformer Blocks (Transformer encoders), each containing feed-forward networks with hidden states of 768 and 12 self-attention heads. The

precise architecture of the Transformer Blocks are identical to the work done by Vaswani et al. (2017). A Transformer's encoder is comprised of a self-attention layer and a feed-forward neural network layer. The self-attention layer takes as input tokens of an input sequence and outputs vector embeddings representing the tokens and the context pertaining to those tokens. Following, the neural network layers take these embeddings and outputs a *hidden state*. The main difference between BERT and other transformer based models (e.g. GPT - Radford 2018) is the fact that BERT uses bi-directional self-attention, meaning that for predicting words it takes into consideration both the context to the left and the context to the right of that word.

The process of pre-training BERT (as well as the fine-tuning process done for this report) follows the same concept of training other Language Models (LM). This is the process where the hidden states are passed through a fully connected neural network (in the case of BERT, a one layer FCNN), with a softmax output over the size of the vocabulary, predicting words. Due to the bi-directionality of the self-attention models in BERT, though, the prediction of words could not follow the standard procedure, as predicting every word in a text would lead to an overlap of attention in the multi-layered context. Therefore, training was done by replacing only 15% of the tokens in a sequence with a *[MASK]* which in turn it would then predict, this process is called Masked Language Modelling (MLM). The cross-entropy loss is then back-propagated through the model.

3.2 Fine-Tuning

For the purposes of this report, we used the Pytorch implementation of BERT created by Hugging Face Inc. (Wolf et al. 2019). Huggingface provides the framework to easily load the BERT_{BASE} pre-trained weights into a BERT Language Model ready for fine-tuning, as well as the BERT Tokenizer. Before proceeding to training the model, we first needed to properly prepare the phrase pairs in our dataset in order to fit the input sequence requirements of the BERT model. This entailed using Huggingface's BERT Tokenizer's *encode_plus()* method. this method takes two text strings as input and returns six vectors: the *input_ids*, the *token_type_ids*, the *attention_mask*, the *overflowing_tokens*, the *num_truncated_tokens*, and the *special_tokens_mask*, all of which can be fed to the BERT model to train with.

In detail, the tokenizer encodes each phrase pair by concatenating the phrase and response with added special tokens: *[CLS]* at the beginning of the sequence and *[SEP]* in between the two phrases as well as at the end of the sequence. Then, the tokenizer encodes the sequence according to the WordPiece (Wu et al. 2016) vocabulary id's. However, because of the peculiar task of conversational text generation, we had to utilize the ability to add a *masked_lm_labels* vector to the input of the model. To ensure that the BERT model only predicts words of the response, while still taking the question into context, we set the positions of *masked_lm_labels* that correspond with the question of the phrase pair and any padding added to the sequence to -100 to indicate that these positions should not be accounted for by the LM loss. For the indexes which correspond to the response phrase, *masked_lm_labels* has the same values as *input_ids*.

The fine-tuning of the BERT and SciBERT models are essentially the same, with the only difference being the vocabulary of each model, which leads to different tokenizers, and different size LM head predicting tokens. Training was conducted using Huggingface's optimizer $AdamW$, with a learning rate of $1e-4$, weight decay of $1e-3$, a batch size of 200 (phrase pairs), and a maximum sequence length of 40. To oversee the training process, we generated text using the fine-tuned model forty-two times throughout a training epoch and evaluated the resulting text.

3.3 Word Generation

For the purposes of a conversational interface, we are required to generate text in a causal manner (left to right), and as BERT is bi-directional, it is not inherently able to generate text in this way. However, BERT is able to predict masked words, similar to the training process for BERT. For this reason, we implemented a similar method as explained in "BERT has a Mouth, and It Must Speak: BERT as a Markov Random Field Language Model" (Wang and Cho 2019). With this method we add a [CLS] at the front of the input text, which is necessary as an input to BERT. Then we add a [SEP] indicating the end of our question, after that we add mask tokens until the predefined maximum length of the phrase, a maximum length of 40 is chosen for our implementation. Finally at the end of the input phrase a [SEP] token is added to indicate the end of the sequence.

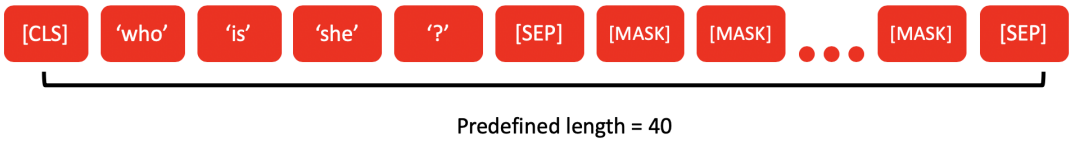


Figure 1: BERT word generation example

Following, the trained model is asked to sequentially predict each masked token, from left to right. As choosing a *pure sampling* strategy (always choosing the maximum softmax score) can lead to incoherent language unrelated to context (Holtzman et al. 2019), a better alternative is the *top-k sampling* generating strategy (Fan, Lewis, and Dauphin 2018). *Top-k sampling* consists of sampling from the k highest softmax scores to produce one token to generate. Improved results are achieved by combining *top-k* with *temperature sampling*, which effects the generating process by creating a more disbursed distribution of the output of the model's LM head. For training, we used *top-k*=50, and temperature of 1.5.

For the final conversational interface an additional processing of the generated text is implemented. The generated text, evidently due to performing causal generation with a bi-directional model, produces multiple sentences and somewhat confusing answers. For this reason, we use a BERT model with a *question-answering* (QA) head to distill the produced answer to a more clear and condensed answer. The BERT for QA predicts the segments of the generated text (by predicting the start and end index) with the best answers to the question originally presented to generate said text. The QA BERT is readily available through Huggingface's *transformers* library.

3.4 Accuracy metrics during training

During training we keep track of several NLP metrics to monitor the performance of our models. BertScore is the first metric that it used and this computes a similarity score for each token in the candidate sentence and compares it to a reference sentence (Zhang et al. 2019). The second metric used throughout training is the BLEU-score. BLEU-score or Bilingual Evaluation Understudy Score counts the number of n-grams in the generated text and compares it to a reference text (Papineni et al. 2002). The counting of the number of n-grams takes into account the occurrence of words in the reference text, by assuring that the score does not reward an abundance of reasonable words. This metric was originally proposed for machine translation but is also widely used for text generation.

Both of these metrics rely on a reference text being available, thus we decided to create a dataset with reference answers to predefined questions. This allowed us to simulate the flexibility of answers that a chatbot needs to generate, compared to a single possible answer. We started of by defining three general questions that are applicable for a chatbot; "who is she?", "Are you okay?" and "Why?". For each of these questions a F1 score is calculated compared to the questions that are available in our dataset of phrase pairs. If the question in the dataset has a F1 score higher than 0.9 this question can be seen as similar to our chosen question and the index is stored. From these indexes we then generate a dataset of possible answers appropriate for each of the three predefined questions. This dataset of answers will be used as a reference to calculate the BertScore and BLEU-score of the answers that the chatbot could generate for our questions.

During training we keep track of the BertScore and the BLEU-score. The model is trained with the standard loss function from the Huggingface library for a masked LM model. This calculates the cross-entropy loss of the predicted masked word. We found that the loss curves were less indicative than our metrics so the loss curves will not be plotted in our report.

4 Performance BERT, SCIBERT

In this section the pretrained BERT and SCIBERT models available from the Huggingface library are used with our word generation method to generate answers to our predefined question. These answers and scores will be used as a comparison to our models.

4.1 BERT$_{base}$

Table 2 shows that the generated text from pretrained BERT$_{base}$ LM model available from the Huggingface library has pretty decent scores. However, table 1 shows that the answers are not in a good conversational matter. The fine-tuning to Cornell Movie-Dialogs database should make it a better conversation.

Question	**Answer**
who is she?	what is she going to do? and what are them! who are the other five!?? [SEP]
Are you okay?	the realtor is looking in -'-, the receptionist. mike left the reception office shortly [SEP]
Why?	how do you feel - " ... this. this. him... it did... just it happened [SEP]

Table 1: BERT$_{base}$ word generation

Question	**BLEU**	**Precision**	**Recall**	**F1**
Q1	0.66	0.45	0.51	0.48
Q2	0.58	0.42	0.45	0.40
Q3	0.89	0.58	0.69	0.62

Table 2: BERT$_{base}$ scores

4.2 SCIBERT$_{base}$

Table 4 illustrates that the generated text from the pretrained SCIBERT$_{base}$ LM model is able to generate even higher scores than the BERT$_{base}$ model. However, table 3 shows that answers are not meaningful. This shows that the scores are not always a perfect representation to the quality of the generated answers.

Question	**Answer**
Who is she?	apparently in for and that of his who it? for who not and the is her who it? [SEP]
Are you okay?	rather as... that.. de thes... so in your real age.
Why?	from of on..... (from the table) (from e _ _. _ [SEP]

Table 3: SCIBERT$_{base}$ word generation

Question	**BLEU**	**Precision**	**Recall**	**F1**
Q1	0.65	0.44	0.50	0.46
Q2	0.77	0.60	0.69	0.64
Q3	0.60	0.54	0.68	0.56

Table 4: SCIBERT$_{base}$ scores

5 Fine-Tuning on Cornell Movie-Dialogs

5.1 BERT Baseline model

The following BERT model was trained for 60 epochs on the Cornell Movie-Quotes Corpus. Table 6 shows that the precision, recall and F1 scores of generated text are a bit worse than the BERT baseline scores shown in table 2. The tables mentioned above also show that there is a strong decrease in BLEU-scores after fine-tuning on this dataset. This shows that there is still room for improvement. Table 5 also shows that the fine-tuned model is able to produce [SEP] tokens this is beneficial for our chatbot context, this means that the bot will not always answer with a fixed length and shows that the model is learning to make the length of the answer flexible.

Question	**Answer**
Who is she?	old lady charles old united light man guy, large to man ben bear general house, 16 life, [SEP]
Are you okay?	seven rocka radio to go to radio man man do [SEP] radio three radio jack radio major threea [SEP]
Why?	[SEP] money big general to call big united david [SEP] ben - man - guy on radio cross army - [SEP]

Table 5: BERT Baseline word generation

Question	BLEU	Precision	Recall	F1
Q1	0.41	0.42	0.46	0.42
Q2	0.37	0.43	0.47	0.45
Q3	0.65	0.43	0.46	0.42

Table 6: BERT Baseline scores

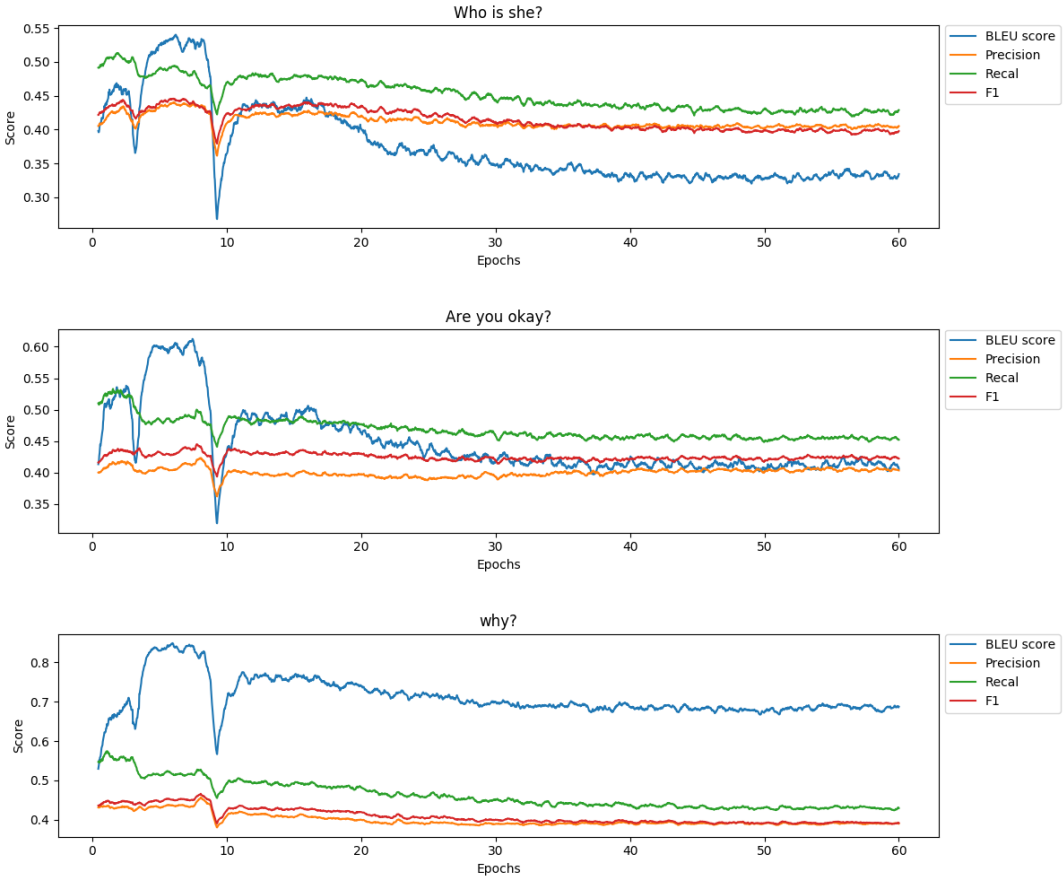


Figure 2: BERT baseline

5.2 BERT model with gradient clipping

In this section we implement gradient clipping on the BERT model during training, this is done because a common problem in natural language processing are exploding gradients, this means that it is possible for the weights to take on infinite values, rendering the network useless. Gradient clipping involves forcing the gradient values to a specific minimum value and maximum value (Brownlee 2019).

Table 8 shows that gradient clipping leads to an improvement in scores compared to our baseline model shown in table 6. However, we can see that the scores do not show an increase

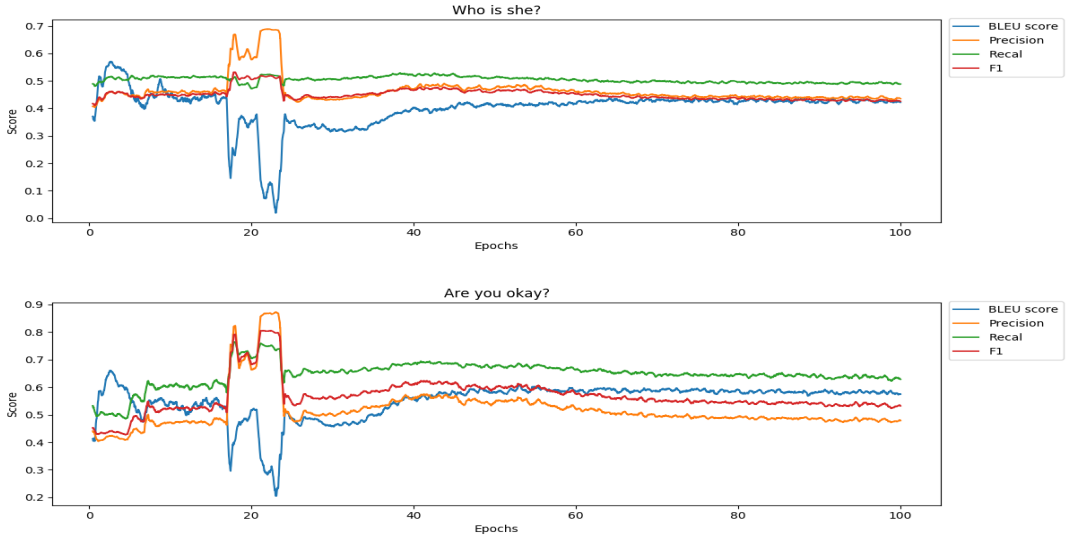
to the BERT$_{base}$ model performance shown in table 2 and that the answers are not in a more conversational structure.

Question	Answer
Who is she?	. hand i. solo bobby.. [SEP] other head head i arm out.. man fine love [SEP]
Are you okay?	case. other okay work go out go serious fine great work head english life love fine okay. i [SEP]
Why?	. general con small 20 now head. con fine other later president well hand back work now i okay [SEP]

Table 7: BERT with gradient clipping word generation

Question	BLEU	Precision	Recall	F1
Q1	0.57	0.46	0.48	0.45
Q2	0.62	0.43	0.58	0.46
Q3	0.87	0.48	0.49	0.49

Table 8: BERT with gradient clipping scores



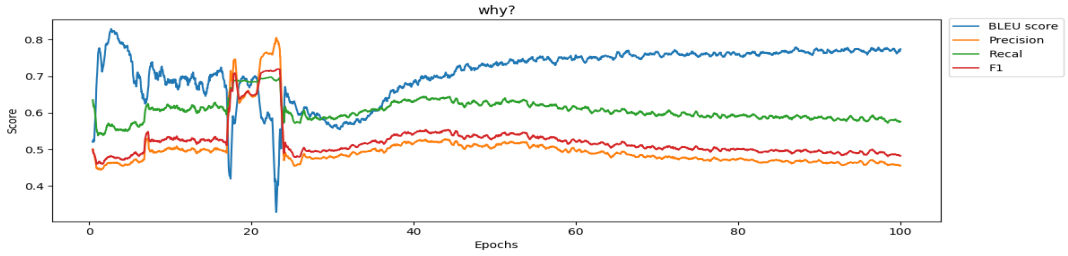


Figure 3: Bert with gradient clipping

5.3 Learning Rate Parameter tuning

The learning rate of a model determines how much the model is going to change its weights based on the calculated loss. A learning rate that is too large will result in unstable learning and a learning rate that is too small will result in a failure to learn. Figure 4 (and appendix B.1) show that the BERT transformer performs best with out original learning rate of 0.0001.

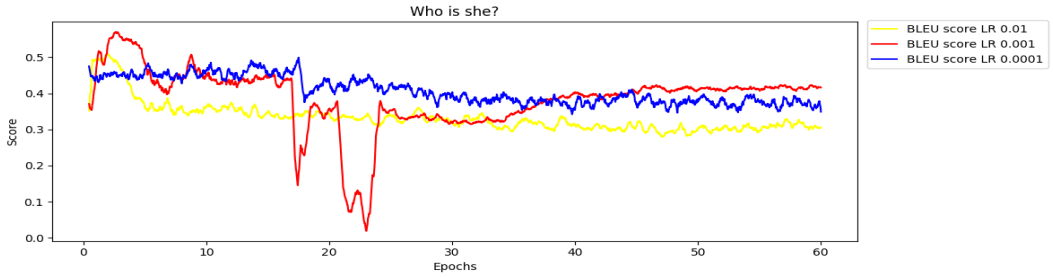


Figure 4: Learning rate parameter tuning

5.4 Weight Decay parameter tuning

Weight decay is used each update during training and prevents the weights of the model from growing too large. Figure 5 (and appendix A.1) show the BERT transformer trained with different weight decay values with gradient clipping implemented. We can see that with our dataset a weight decay value of 0.0001 is the quickest to converge but overall the model with a weight decay of 0.001 results in the best performance.

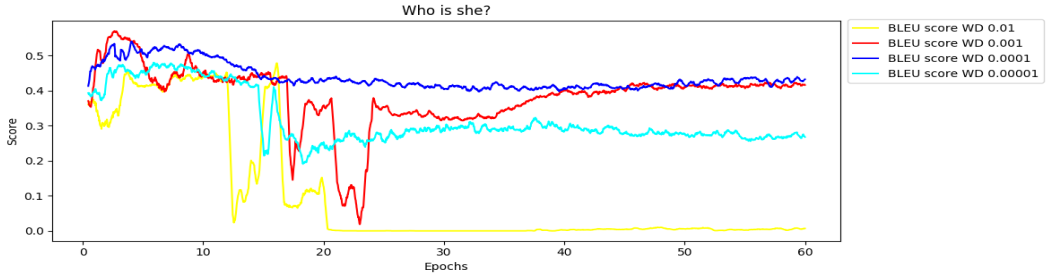


Figure 5: Weight decay parameter tuning

5.5 Learning Rate Schedules

In this section two learning rate schedules from the Huggingface library were tested to see if they would yield any improvement in performance. First a learning rate schedule with a cosine warmup is used where the number of warmup steps is equal to 10% of the training length. Next, a cosine with hard restarts schedule is implemented with the number of warmup steps still equal to 10% of the training length.

Figure 6 shows that the cosine warmup schedule does not perform well for our implementation and that the cosine hard restarts performs a bit better than the cosine warmup but still the original constant learning rate performs best. This shows that the *ADAMW* optimizer from Huggingface library that is used in our original models is a powerfull optimizer.

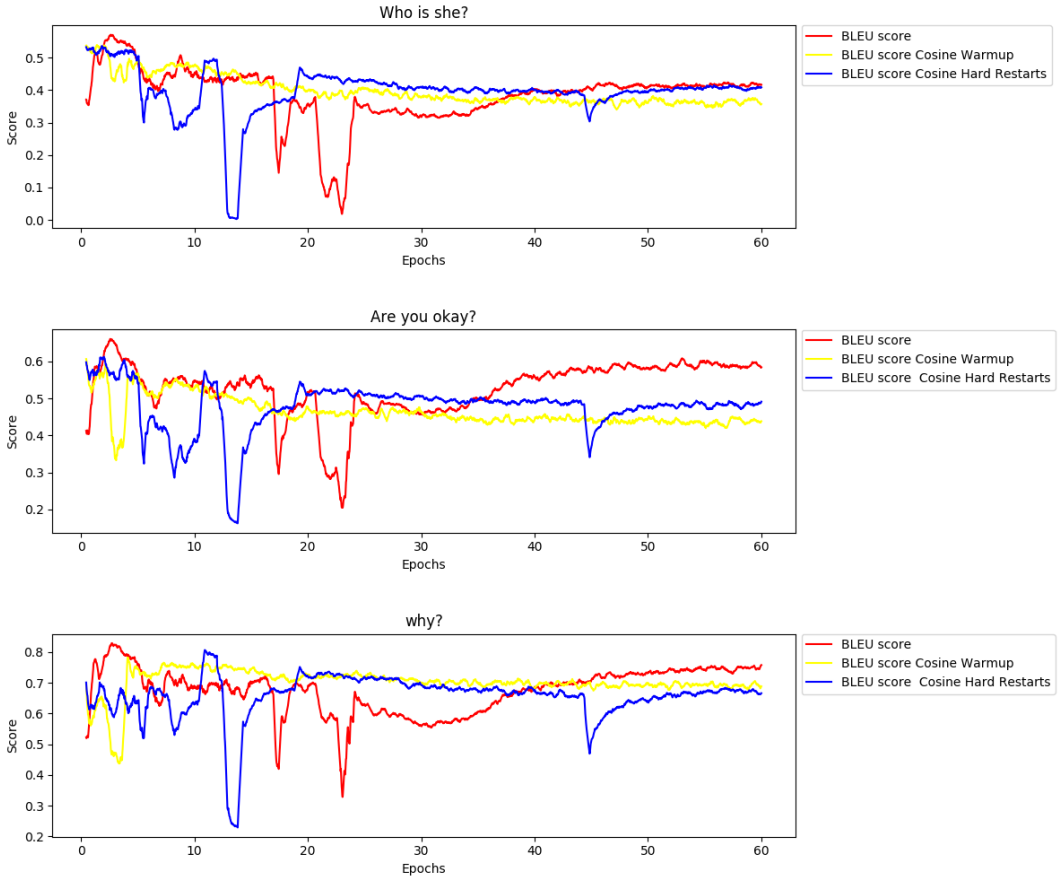


Figure 6: Learning rate schedules with a learning rate of 0.0001

5.6 SciBERT fine-tuning

The process of fine-tuning SciBERT followed the same structure as with BERT, and even though we expected SciBERT to train best with the same optimal values for BERT, we found that SciBERT trains best with a bigger learning rate of 0.001 and a smaller weight decay of 0.0001 (see Appendix C.1). It was also noted that while SciBERT increases precision,

recall, and F1 scores during training, the BLEU score drastically decreases(Figure 7). We hypothesise that this occurs due to the fact that SciBERT is pre-trained on scientific papers and the language nuances of these papers do not match the context of the Cornell dataset that we are fine-tuning the model on, leading to conflicts of language interpretation.

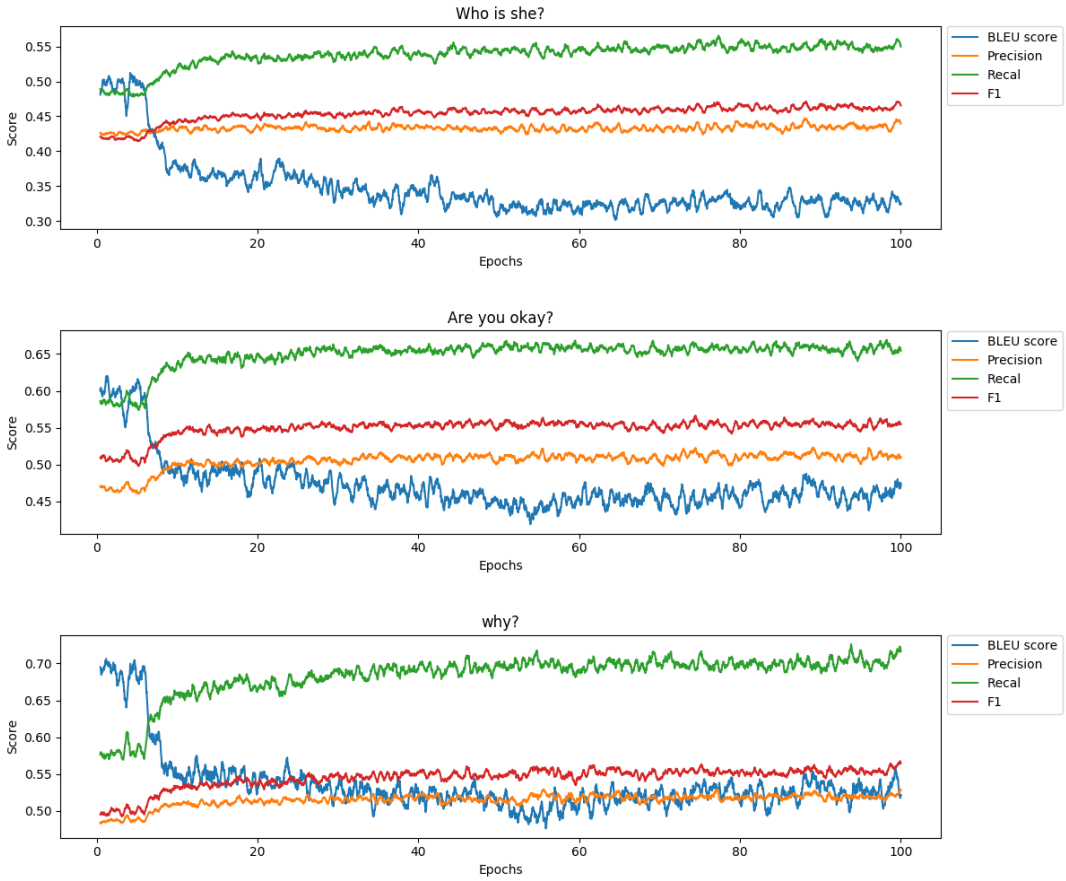


Figure 7: SciBERT performance

6 Word generation parameter tuning

Figure 8 shows that our fine-tuned BERT and SciBERT models perform best with different word generation values. BERT is most powerful with a *top-k* value of 80 and with a *temperature sampling* value of 1.5. SciBERT is most with a *top-k* value of 120 and with a *temperature sampling* value of 4.

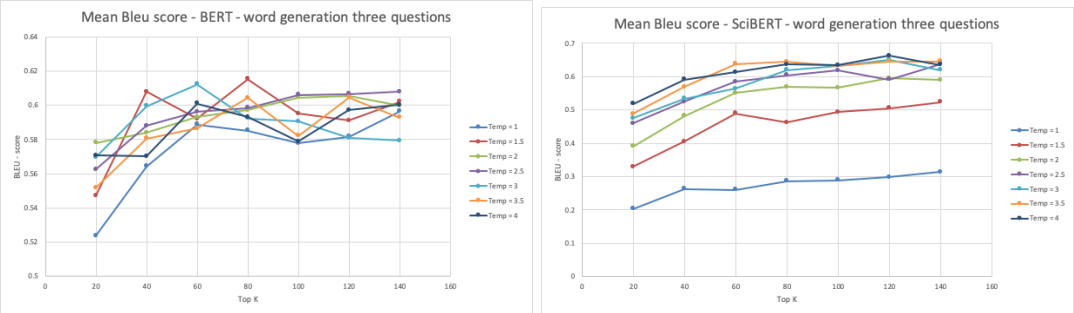


Figure 8: Word generation parameter tuning

7 Final BERT and SciBERT models

The final fine-tuned models of BERT and SciBERT were trained for a 100 epochs. BERT used a learning rate of 0.0001, a weight decay value of 0.001, a batch size of 200, an AdamW optimizer, and gradient clipping. SciBERT used a learning rate of 0.001, a weight decay value of 0.0001, a batch size of 200, an AdamW optimizer, and gradient clipping. For word generation, BERT uses a temperature of 1.5 and top-k of $k = 80$, while SciBERT uses a temperature of 4 and top-k of $k = 120$. Using the above parameters, the output and scores of each model can be seen below.

Question	**Answer**	**BLEU**	**Precis.**	**Recall**	**F1**
Who is she?	solo [SEP] major crazy head race arm letter president head flight. now life life.. major chief	0.41	0.44	0.48	0.43
Are you okay?	.. hand night. okay.. head president president chief. small man job fine now president west	0.55	0.53	0.70	0.60
Why?	. other life later business outside hand hand major work great business head back other president safe. president fire	0.81	0.41	0.61	0.48

Table 9: BERT Fine-tuned word generation and scores

Question	**Answer**	**BLEU**	**Precis.**	**Recall**	**F1**
Who is she?	you what now are? the all about so nothing great! isy pittsburgh never well north why why	0.56	0.42	0.52	0.46
Are you okay?	in absolutely why al him do where breaking thankss with [SEP]. columbia toah anything great where please	0.58	0.41	0.48	0.435
Why?	like whaty anything. see that about it he. you hu nothing north yes me he well that	0.94	0.48	0.50	0.46

Table 10: SciBERT Fine-tuned word generation and scores

These values do not show a great improvement compared to our baseline scores. However, when we use a BERT model with a *question-answering* (QA) head as discussed in section 3.3 for the Graphical User Interface in order to select from the generated answers a more clear and condensed answer. Then, table 11 shows that with the *QA* head the generated answers are meaningful and show that the GUI will be able to behave as a basic chatbot.

Question	**BERT fine-tuned answer**	**SciBERT fine-tuned answer**
Who is she?	chief general	nice general fine flight
Are you okay?	i fine	fine last safe hand night. i fine
Why?	business	my large money

Table 11: Answers models with QA

8 Graphical User Interface

The graphical user interface (GUI) combines both of our models in an easy to use user interface. In order to discern whether the answer should be generated by the BERT or the SciBert model, the GUI begins by calculating the readability score of the asked question. The readability score indicates how difficult a passage is to understand by focusing on textual content such as lexical, semantical, syntactical and discourse cohesion analysis (Benzahra 2019). The score is calculated by using the average sentence length and the average word length of the sentence. A lower score indicates that the passage is more difficult to read.

If the readability score is lower than 50, meaning the passage is college level, then the GUI recognizes that the question is difficult to read and that a more scientific answer is needed. Thus, the answer is generated by the SciBERT trained model. The SciBERT trained model can also be called manually by clicking the "*SciBERT answer*" button. Once the text is generated, the text is fed through the *BERT for QA* model, so that the particular segment of text best correlating to the question asked is chosen as the final answer. In figure 9, you can see the format of the GUI.

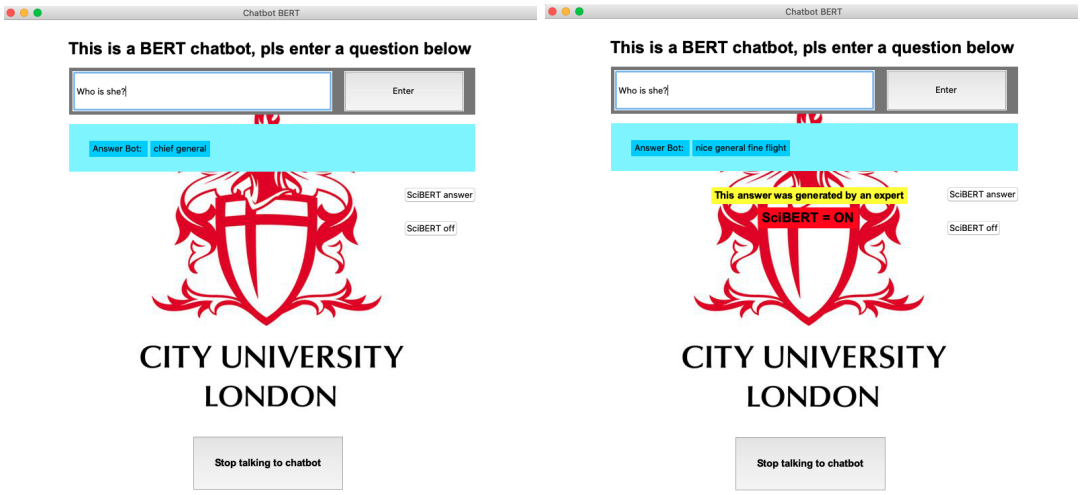


Figure 9: Graphical User Interface

9 Conclusion

In this paper we introduce a novel structure for word generation with BERT and introduce a training procedure to fine-tune BERT's bidirectional behaviour to the conforms of causal word generation, needed when developing a chatbot. In order to evaluate the performance of the trained models, both during and after the fine-tuning process, we used metrics such as BERTScore and BLEU-score, as loss was not properly indicative of performance. For this reason, we chose three questions and created a list of appropriate answers from the Cornell Movie Dialog dataset to which we could use as references for evaluation.

The final result of our fine-tuning process, while showing comparative results with the BERT_{base} model performances, still do not generate fully understandable answers. This can be attributed both to the fact that BERT's bi-directional self-attention does not adapt well to causal generation, and also due to the fact that more training, or a larger dataset, is necessary to fully adapt to conversational nuances and context. To deal with this weakness, we utilize Huggingface's Bert for QA model, to segment the generated answer into a concise, and more appropriate answer, leading to good results.

Another approach to increasing the performance of the BERT chatbot, which we leave as suggestions for future work, is to incorporate the metrics we use for evaluation, namely the BLEU score, into the loss function used for training. The exact process in which this could lead to better training would need further analysis and research. In conclusion, while the BERT model has not been commonly linked to NLP tasks of left to right text generation, we see through this report that is can be used for such tasks with only certain extensions to the core model structure.

References

Benzahra, Marc (2019). *How to Evaluate Text Readability with NLP*. URL: https://medium.com/glose-team/how-to-evaluate-text-readability-with-nlp-9c04bd3f46a2. (accessed: 15.05.2020).

Brownlee, Jason (2019). *How to avoid exploding gradients with gradient clipping*. URL: https://machinelearningmastery.com/how-to-avoid-exploding-gradients-in-neural-networks-with-gradient-clipping/. (accessed: 23.04.2020).

Danescu-Niculescu-Mizil, Cristian and Lillian Lee (2011). *Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs.*

Devlin, Jacob et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805. arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

Fan, Angela, Mike Lewis, and Yann Dauphin (2018). *Hierarchical Neural Story Generation*. arXiv: 1805.04833 [cs.CL].

Holtzman, Ari et al. (2019). *The Curious Case of Neural Text Degeneration*. arXiv: 1904.09751 [cs.CL].

Papineni, Kishore et al. (July 2002). "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://www.aclweb.org/anthology/P02-1040.

Radford, Alec (2018). "Improving Language Understanding by Generative Pre-Training". In:

Vaswani, Ashish et al. (2017). *Attention Is All You Need*. arXiv: 1706.03762 [cs.CL].

Wang, Alex and Kyunghyun Cho (2019). "BERT has a Mouth, and It Must Speak: BERT as a Markov Random Field Language Model". In: *CoRR* abs/1902.04094. arXiv: 1902.04094. URL: http://arxiv.org/abs/1902.04094.

Wolf, Thomas et al. (2019). "HuggingFace's Transformers: State-of-the-art Natural Language Processing". In: abs/1910.03771. arXiv: 1910.03771 [cs.CL].

Wu, Yonghui et al. (2016). *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. arXiv: 1609.08144 [cs.CL].

Zhang, Tianyi et al. (2019). "BERTScore: Evaluating Text Generation with BERT". In: *CoRR* abs/1904.09675. arXiv: 1904.09675. URL: http://arxiv.org/abs/1904.09675.

Appendices

A Weight Decay parameter tuning

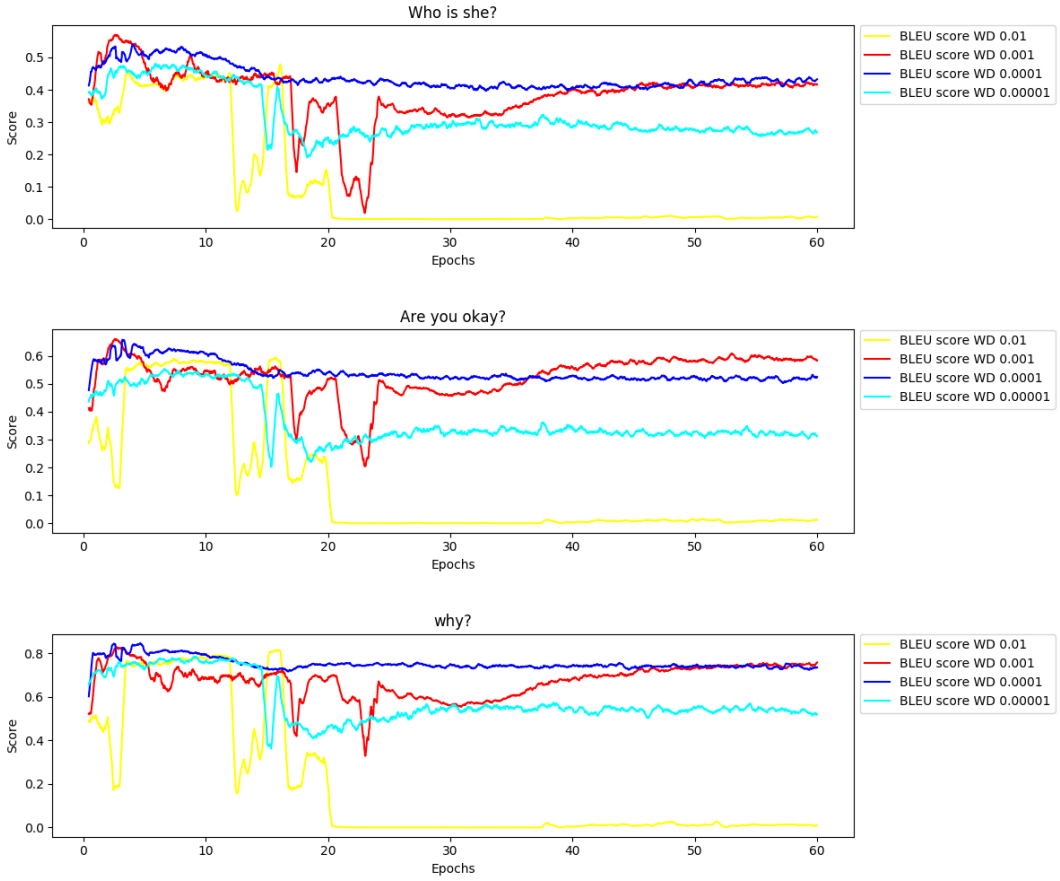


Figure A.1: Weight Decay parameter tuning

B Learning Rate Parameter tuning

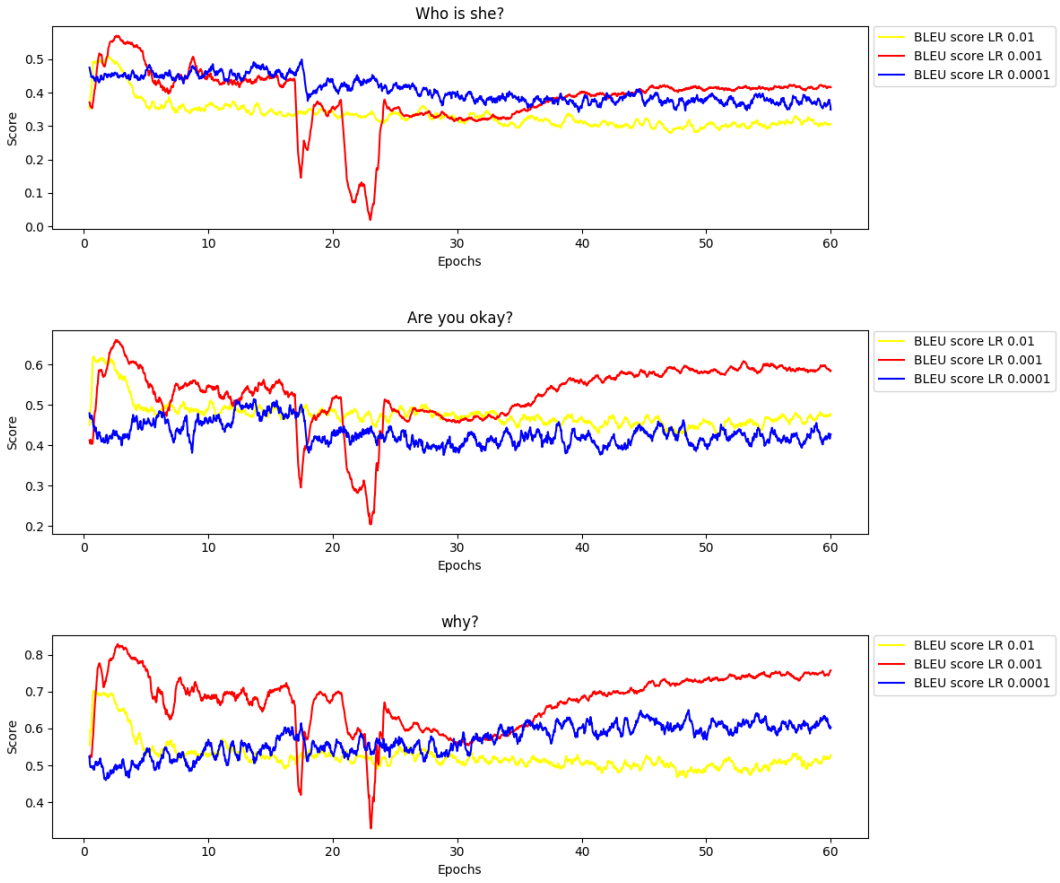


Figure B.1: Learning Rate Parameter tuning

C SciBERT parameter tuning

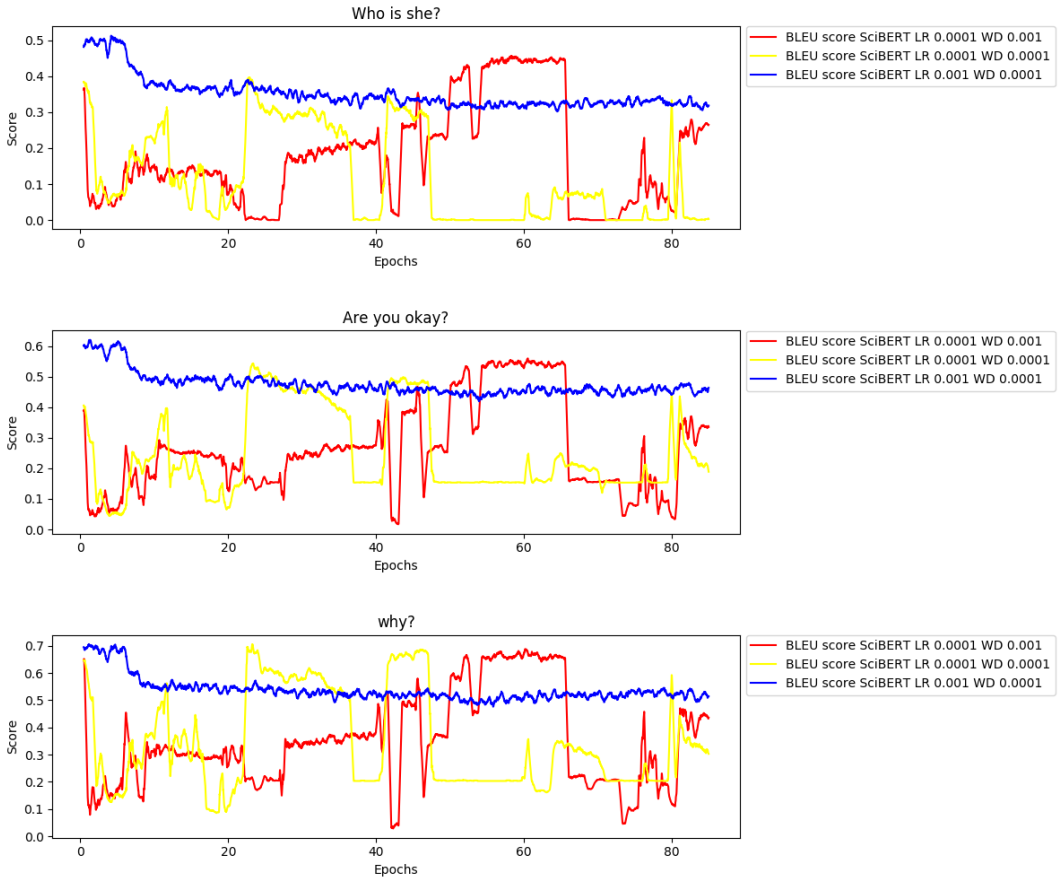


Figure C.1: SciBERT parameter tuning