Sentence Level Pretraining for Natural Language Inference

Sungjun Han
Group 5

Anastasiia Sirotina
Group 5

Abstract

In this paper we discuss a novel approach to improve the performance of transformer-based models for Natural language inference (NLI) task. We hypothesize the knowledge of the relationships between the premise and the hypotheses needed to be successful in NLI can be extracted from an unannotated corpus in a self-supervised manner. We propose two training objectives to achieve this: *Sentence Level Language Modelling* (SL-LM) and *Sentence Level Masked Language Modelling* (SL-MLM)[1]. To show the conceptual validity of this hypothesis we compare performance of transformer-based models with pretraining to the non-pretrained models on a chosen NLI task.

1 Introduction

The task of Natural Language Inference (NLI) is to determine whether the hypothesis s is entailed in the given premise o. The hypothesis s is said to be entailed in the premise o, denoted as $o \models s$, when in the world where the state or situation s has occurred, humans can reasonably conclude that o has also occurred. The entailment is uni-directional (i.e. $(o \models s) \not\Longrightarrow (s \models o)$), making the task very challenging to learn.

NLI has proved to be a rather challenging task for the existing models (Bhagavatula et al., 2019), (Nie et al., 2020). Usually, the most successful models are pretrained on a large collection of natural language data found in the Web, then fine-tuned on a specific NLI dataset. Through fine-tuning, the model extracts the relationships between the hypotheses and the premises given in the dataset. However, collecting such datasets is expensive and time-consuming. Furthermore, such models are only able to perform well on the specific domain which they were trained on.

In this paper we aim to learn to extract the premise-hypothesis relationships from freely available non-annotated text from the Web. Then, the model can be adapted in a few-shot or zero-shot manner on a specific NLI dataset. To achieve this, we pretrain two prominent transformer models over a large dataset in a self-supervised manner.

2 Sentence Level (SL) Pretraining

The ability to do inference on whether the sentence s is entailed in the given sentences o^L and o^R (one sentence to the left and one sentence to the right respectively), can be framed probabilistically as being able to properly compute the conditional probability $p(s|o^L, o^R)$. In order to properly compute this probability, it would be insufficient for the model to deal only with the semantics of the individual sentences. The model should also possess common sense knowledge to be able to judge the complex entailment relationships between sentences.

We hypothesize that the above mentioned probability distribution can be learned from the Web in a self-supervised manner. To learn to compute the probability, the dataset of tuples (s, o^L, o^R) can be easily collected by randomly sampling sentences around a chosen sentence in any document. Hence, we test whether a model pretrained in this way on a corpus collected from the Web is able to do better than a model without any further pretraining.

We derive two pretraining objectives: *Sentence Level Masked Language Modelling* (SL-MLM) and *Sentence Level Language Modelling* (SL-LM).

SL-LM decomposes the above mentioned probability as a product of conditional probabilities over the sequence of tokens of s: $p(s|o^L, o^R) = \prod_{i=1}^{|s|} p(s_i|o^L, o^R, s_{<i})$. We can also represent o^L and o^R as a sequence of tokens. The SL-LM objective is:

$$p(s|o^L, o^R) = \prod_{i=1}^{|s|} p(s_i|o^L_1, ..., o^L_{|L|}, o^R_1, ..., o^R_{|R|}, s_{<i}) \tag{1}$$

SL-MLM approximates the above mentioned probability by representing the sentence with a few masked tokens $M = \{s_j\}_{j=1}^{|M|}$. Following the same approach as before and assuming that the probability of one masked token does not depend on another, the SL-MLM objective is:

$$p(s|o^L, o^R) \approx \prod_{s_i \in M} p(s_i|o^L_1, ..., o^L_{|L|}, o^R_1, ..., o^R_{|R|}, s_{\notin M}) \tag{2}$$

3 Relevant Works

There are many works that attempted to extract information on a sentence level from a large amount of data collected from the Web. Skip-thought (Kiros et al., 2015) attempted to learn general sentence embeddings by predicting the tokens in the sentences around each sentence using an encoder-decoder architecture (Sutskever et al., 2014). BERT (Devlin et al., 2018) learned shallow alignments between contiguous sentences by predicting whether the second sentence is a contiguous sentence from the first sentence or not. Struct-BERT (Wang et al., 2019) extended BERT's next sentence prediction objective to the previous, next and random predictions. This

[1]The code is to be found in our GitHub repository: https://github.com/hansungj/NLPLab

resulted in improvements in various down-stream tasks. CNTRL (Keskar et al., 2019) tried to learn the relationship between the meta-information and the information contained in the sentences by learning the probability of the sentences conditioned on control tokens. Recently, (Ennen et al., 2021) learned to predict the next user response given the future and the past system responses using dialogue datasets. They observed drastic improvement in test perplexity and BLEU score (Papineni et al., 2002).

4 Dataset

4.1 Pretraining

We test the conceptual validity of our hypothesis using a corpus called BookCorpus (Zhu et al., 2015) for further pretraining. BookCorpus consists of books in a wide variety of genres. Therefore, it is rich in both high-level semantic information, such as how one is feeling and how this evolves through a story, and low-level factual information, such as what objects look like. We believe that books hold more rich and coherent relationships between the sentences than texts in other domains such as Wikipedia. Furthermore, the corpus is relatively small (5Gb) compared to other popular web-crawled corpora. This best suited us given the very limited availability of computational resources and time.

The data is prepared as follows: given a sentence s_i, we defined a local context window to the left l_i and to the right r_i according to the hyperparameters $L \in \{1, 2, ...\}$ and $R \in \{1, 2, ..\}$. L and R define the number of sentences to consider to the left of s_i and to the right respectively. For example with $L = 2$, the left context is $l_i = \{s_{i-2}, s_{i-1}\}$. We randomly sampled from each context independently to acquire o_i^L and o_i^R which are the sampled sentences from the left and the right respectively. For each tuple (s, o^L, o^R) from the dataset, we concatenated this tuple to form the input where the sentences were separated by a separation token ($[SEP]$ here): $[o^L; [SEP]; o^R; [SEP]; s]$. The identity of the sentences were made explicit by prepending control codes: $"before:"$ for o^L, $"after:"$ for o^R, and $"hypothesis:"$ for s.

4.2 NLI Task

We chose to work with ART dataset for the NLI task (Bhagavatula et al., 2019) as it was shown to be an exceptionally challenging dataset for multiple models, including BERT (Devlin et al., 2018) and GPT-2 (Radford et al., 2019). The task setup in ART is as follows: given two premises o_1 and o_2 and two hypotheses h_1 and h_2, the model has to predict, which of the hypotheses is more plausible in being the answer for the question *"What happened between the premises?"*.

5 Experiment Details

5.1 Sentence Level (SL) Pretraining

5.1.1 Language Modelling

In order to test the effectiveness of the proposed language modelling objective for further pretraining for NLI, we use GPT-2 (Radford et al., 2019). GPT-2 is a transformer (Vaswani et al., 2017) decoder with layer normalization added at the beginning of each sub-layer. We chose GPT-2 to test SL-LM, as this is one of the widely known pretrained language model. We used the freely available pretrained *GPT-2-base* model released by Huggingface (Wolf et al., 2019).

We used "|" as the segmentation token in GPT-2. In order to make the difference between the premises and the hypothesis explicit, we also introduced segmentation ids: 0 was given to the sub-word tokens of o^L and o^R and 1 was given to the sub-word tokens of s. We employed cosine learning weight schedule with warm-up (Loshchilov and Hutter, 2017) for 16k steps with the learning rate of $4e^{-5}$. Due to the lack of computational resources, we only pretrained for 1.3M steps on two GPUs with batchsize of 8 using Adam (Kingma and Ba, 2015) by minimizing the negative log likelihood with SL-LM. This is about quarter of a single pass of the corpus and it took about 6 days to train. We accumulate gradients for 8 batches to approximate training with a larger batch size of 64. Also, we truncate the input length of each context to 128 and the target to 92.

5.1.2 Masked Language Modelling

We chose BERT to test SL-MLM pretraining objective as BERT is one of the most widely known model pretrained with MLM and has shown be successful in multiple NLI benchmarks. We used the freely available pretrained *BERT-base-uncased* model released by Huggingface (Wolf et al., 2019).

We used *[SEP]* as the segmentation token for BERT. We followed the masking strategy for Masked Language Modelling described in (Devlin et al., 2018). We randomly chose 30% of all the tokens for masking. This masking probability is much higher than the probability used in original BERT pretraining. This is done as BookCorpus was already used during its original pretraining for BERT. Thus, we wished to make the task harder and push the model to focus in using the information from the provided contexts for the predictions. The model was pretrained by minimizing the negative log likelihood with SL-MLM for 600k steps on one GPU with batch of size 8 using Adam and this took around 3 days to train. We applied cosine learning weight schedule with warmup for 10k steps with learning rate of $1e^{-5}$. The lengths of the sentences were truncated and the same segmentation ids were introduced in the same way as the SL-LM pretraining.

5.2 Fine-tuning

5.2.1 Baseline

We have implemented several baseline models and we briefly introduce them here. The first baseline model is the Bag-of-Words (BoW) model for NLI introduced in (MacCartney, 2009). The model calculates the conditional probability $P(h|p)$ of a hypothesis h given a premise p. Expressing this probability in terms of costs for convenience, it decomposes to:

$$cost(h|p) = \sum_j weight(h_j) \cdot cost(h_j|p) \quad (3)$$

$$cost(h_j|p) = \min_i -\log sim(h_j, p_i) \quad (4)$$

where h_j and p_i are the tokens of the hypothesis and the premise respectively. For each hypothesis word h_i, the model finds the most similar word to align in p. $sim(h_j|p_i)$ can be estimated using a string-similarity measure such as *Levenshtein distance* (Miller et al., 2009) or *cosine similarity* from distributional vector-space models. The final score is given by the sum of these individual scores weighed by how important each word h_i is. We use *inverse-document-frequency* as this weighting function. For ART, we calculated the total score for each hypothesis and subtract the two scores. Then this was passed to a classifier of choice for classification. We used Perceptron (Rosenblatt, 1958) and Max-Ent (Berger et al., 1996) classifier. For further details on BoW, please refer to (MacCartney, 2009).

We also implemented simple deep learning models as baselines. In these models, the hypothesis h and the premise p are converted into embeddings through a shared embedding matrix, which is initialized using GloVe (Pennington et al., 2014). Then the embeddings are pooled by an encoder into a single sentence vector separately for the premise and hypothesis. The encoder can be implemented by CNN (Kim, 2014), RNN (LSTM (Hochreiter and Schmidhuber, 1997)) , or FFN with mean-pooling. We explored all three encoder options. Then the sentence vectors are concatenated. For ART, we passed each hypothesis through an weight-tied encoder. In order to distinguish the difference between the premise and the two hypotheses, we used the following concatenation scheme: $[p; h_1; h_2; p - h_1; p - h_2]$. Then this vector was put through a multi-layer FFN for classification. We distinguish these models by their choice of the encoder: *FFN, RNN, CNN*.

5.2.2 BERT and GPT-2

We describe the fine-tuning setup on ART for both BERT and GPT-2 here. We prepared the inputs by concatenating the premises and the hypothesis separated by separation tokens in the same way as done during the pretraining. We also added the $[CLS]$ token for classification. This was inserted in the beginning and the end for BERT and GPT-2 respectively. We made the identities of the premises and the hypothesis explicit as done in pretraining by adding control codes, see Section 5.1.

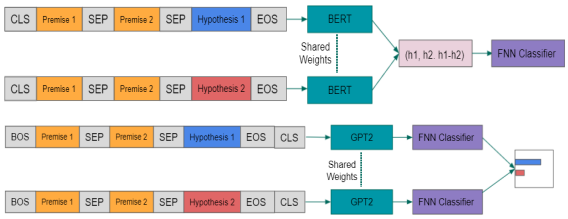

Figure 1: Dual Encoder architecture for BERT (top) and GPT-2 (bottom)

We used a dual-encoder architecture for both BERT and GPT-2 where the two hypotheses are put into the model separately with the weights of the two models tied, see Figure 1. The dual encoder architecture has shown to be effective for NLI for BERT in (Reimers and Gurevych, 2019). We observed that simply concatenating all the premises and hypotheses as the input for BERT and classifying on [CLS] gives a comparable result, see *BERT-concat* in Table 1. However, we used this architecture to make BERT comparable with GPT-2. We observed experimentally that simple concatenation does not work very well for GPT-2, thus we used the dual architecture. In this architecture, both BERT and GPT-2 takes the last layer's hidden representation of the *[CLS]* token from the encoders are put through a three-layer FFN classifier with LayerNorm (Ba et al., 2016) and GELU (Hendrycks and Gimpel, 2020) activation for classification with residual connections. BERT concatenates them in the following way: $[h_1, h_2, h_1 - h_2]$. GPT-2 instead keeps a separate classifier for each hypothesis and the final logits are concatenated and put through a softmax layer.

We trained using a cross-entropy loss defined over the dataset for both models. For GPT-2, following the observation by (Radford et al., 2018) that auxiliary language modelling objective improves the classification, we introduced this auxiliary objective $L_{LM}(\theta)$ only for the correct hypotheses,

$$L_{LM}(\theta) = -\sum_{i=1}^{|D|}\sum_{t=1}^{T}\sum_{j=1}^{2} logp(h_{j_t}|o_1^i, o_2^i, h_{j_{<t}})I_{j=y^i} \quad (5)$$

where $I_{j=y^i}$ is an indicator function. The relative contribution of L_{LM} in the total loss from is controlled by the hyperparameter λ. Without this auxiliary objective, we observed a drastic drop in performance for GPT-2. We set $\lambda = 0.5$. GPT-2 was trained for 10 epochs and BERT for 5 epochs. Both used cosine learning rate decay schedule with warm-up for 0.1 of the total steps.

5.3 Evaluation

For BERT and GPT-2 models, we test the model on the provided entire validation set at each fine-tuning epoch and report this validation curve. This was done

Model	Accuracy
single-class baseline (predicts only 1)	50.97
BoW + Perceptron + Levenshetein	50.97
BoW + MaxEnt + Distributional	51.52
FNN w/ mean-pooling	52.73
RNN	55.10
CNN	56.13
BERT-concat	61.82
GPT-2-base*	**63.5** $\pm$ **0.01**
GPT-2-SLLM*	**63.0** $\pm$ **0.01**
BERT-base*	**60.6** $\pm$ **0.02**
BERT-SLMLM*	**61.6**$\pm$ **0.02**

Table 1: Accuracy on the validation set (higher is better). We evaluated on the validation set after a certain number of training epochs for the baseline models. *For GPT-2 and BERT, we state accuracy averaged over all epochs for easy comparison.

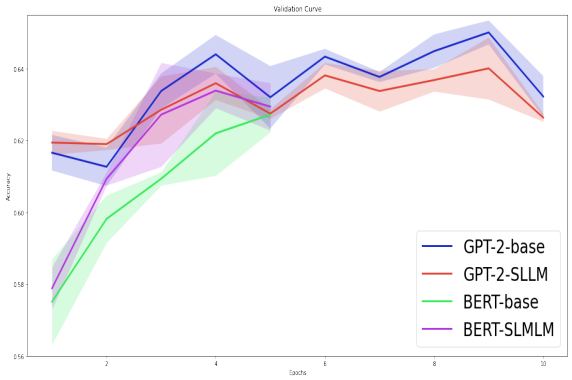

Figure 2: We report accuracy at each epoch on the validation set for GPT-2 and BERT. The shaded region represents 1 standard deviation. See Appendix A for the data for the graph.

as the test set was not easily accessible. Also, since the validation set of ART was chosen explicitly to have a different distribution than the train set, doing model selection by setting out part of the train set was not a viable option. We ran each model three times and report on their statistics.

6 Result

We first observe that ART is a very challenging dataset. BoW's token based similarity matching between the premise and the hypothesis performed just above the baseline that predicts only one label, see Table 1. The deep learning baselines did a bit better but they are also not far from chance. GPT-2 performed similar to BERT. This was somewhat surprising as GPT-2 is a model that is order of magnitude larger than BERT. However, this result could be explained by GPT-2 only considering the left context unidirectionally, not bidirectionally like BERT.

6.1 SL-LM

The GPT-2 pretrained with SL-LM (GPT-2-SLLM) performed worse than the model without SL-LM (GPT-2-base) by around 0.5% averaged over all 10 epochs, see Table 1. Looking at the validation curve in Figure 2, we see that this performance gap due to GPT-2-SLLM doing consistently worse after the third epoch. In order to test the statistical significance of these differences, we performed Student's t-Test (Student, 1908) for each epoch. None of the differences were significant as all $p > 0.1$ with average $p = 0.31$, see Appendix A.

Despite this result, it is worth mentioning that SL-LM seems to be helping with convergence in fine-tuning as it is able to do better in the earlier epochs. To test this, we trained the model with a different learning rate for 5 epochs over 2 runs. In these runs, we again observe that GPT-2-LM is able to consistently do better than GPT-2-base in the first 3 epochs. Considering fine-tuning the model on new datasets is usually expensive for a large model such as GPT-2, it is promising result as a pretrained model could quickly adapt to new domains.

6.2 SL-MLM

The BERT pretrained with SL-MLM (BERT-SLMLM) performed better than the model without SL-MLM (BERT-base) with 1% gain averaged over all epochs, see Table 1, and higher accuracy in all epochs, see Figure 2. However, we again did not find a significant statistical difference between the two models with all $p > 0.1$ with average $p = 0.38$, see Appendix A. However, considering SL-MLM pretraining was only run for a short amount of time, covering only around 10% of BookCorpus, the improvement we observe is promising. This is more surprising considering BookCorpus was already used in the original pretraining for BERT. This suggests that the model was extracting some genuine relationships between the sentences with SL-MLM. Also, similar to GPT-2 we observed that BERT with SL-MLM converged faster in fine-tuning with $p = 0.18$ averaged over the first four epochs.

7 Discussion and Conclusion

There are many possible reasons behind the insignificant statistical difference between the models trained with and without the SL pretraining. First, due to the lack of computational resources and time, both pretraining objectives could only be run for a relatively short amount of time, covering less than half of BookCorpus for both models. Second, we believe that the truncation of sentences for the pretraining had a significantly negative impact on the models particularly for GPT-2. GPT-2 was originally pretrained using contiguous sentences of length 512, hence further pretraining with truncated sentences might have detrimentally altered the parameter space leading to catastrophic forgetting. (Popel and Bojar, 2018) also noted that using truncation leads to significantly worse results for transformer models. However, truncation was needed to fit the model even for the smallest batch size of 8. Third, transformer models are

notoriously sensitive to the hyperparameter settings. We closely followed the hyperparameter settings specified in the original works for pretraining, but these settings were for pretraining the model *from scratch* not further pretraining the already pretrained model. Fourth, both BERT and GPT-2 were already pretrained on BookCorpus. Hence, newly learnable relationships could have been small compared to other datasets new to the models.

We proposed new pretraining objectives to learn useful relationships between premises and hypotheses in a self-supervised manner from unannotated corpora for NLI. The results were inconclusive, but some promising results could be seen for both GPT-2 and BERT. In the future research, we wish to test the SL objectives with more computational resources and a more diverse set of datasets. Specifically, we wish to use the datasets with rich relationships between their sentences like BookCorpus, but not used in pretraining BERT or GPT-2 and train the models with the SL objectives for much longer.

Contributions

The code is to be found in our GitHub repository: https://github.com/hansungj/NLPLab

The work was splitted in the following way:
1) Baseline models were implemented in parallel by both Sungjun Han and Anastasiia Sirotina
2) GPT2 based models + pretraining scripts were implemented by Sungjun Han
3) BERT based models + pretraining scripts were implemented by Anastasiia Sirotina
4) All GPT-2 pretraining was conducted by Sungjun Han
5) All BERT pretraining was conducted by Anastasiia Sirotina
6) All fine-tuning experiments for BERT and GPT-2 were done by Sungjun Han
7) Analysis for comparing further pretrained vs. without further pretraining was done by Sungjun Han.
In the code release, the relative contributions are also noted in each script.

The work on paper was splitted in the following way:
Sungjun Han worked on sections: 1;2;3;4.1;5.1;5.2.2;5.3; 6;7;Tables;
Anastasiia Sirotina worked on sections: Abstract, 1, 4.2; 5.2.1;Figures

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Wen-tau Yih, and Yejin Choi. 2019. Abductive commonsense reasoning. *arXiv preprint arXiv:1908.05739*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Philipp Ennen, Yen-Ting Lin, Ali Girayhan Ozbay, Ferdinando Insalata, Maolin Li, Ye Tian, Sepehr Jalali, and Da shan Shiu. 2021. Towards a universal nlg for dialogue systems and simulators with future bridging.

Dan Hendrycks and Kevin Gimpel. 2020. Gaussian error linear units (gelus).

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

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors.

Ilya Loshchilov and Frank Hutter. 2017. Sgdr: Stochastic gradient descent with warm restarts.

Bill MacCartney. 2009. *Natural language inference*. Stanford University.

Frederic P. Miller, A. Vandome, and John McBrewster. 2009. Levenshtein distance: Information theory, computer science, string (computer science), string metric, damerau?levenshtein distance, spell checker, hamming distance.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia,

A Appendix - Results

Epoch	1	2	3	4	5	6	7	8	9	10
GPT-2-base: mean	0.6166	0.6127	0.6338	0.6440	0.6320	0.6433	0.6377	0.6449	0.6501	0.6322
GPT-2-base: std	0.0049	0.0053	0.0048	0.0053	0.0086	0.0021	0.0014	0.0046	0.0033	0.0056
GPT-2-SLLM: mean	0.6194	0.6190	0.6285	0.6359	0.6275	0.6381	0.6338	0.6368	0.6401	0.6264
GPT-2-SLLM: std	0.0033	0.0015	0.0093	0.0045	0.0013	0.0035	0.0056	0.0031	0.0086	0.0011

Table 2: GPT-2 on ART evaluated on validation set at each epoch for 10 epochs. GPT-2-base refers to the model without any further pretraining with SL-LM and GPT-2-SLLM refers to the model with further pretraining with SL-LM.

Epoch	1	2	3	4	5
BERT-base: mean	0.5706	0.5952	0.6049	0.6220	0.6272
BERT-base: std	0.0073	0.0100	0.0048	0.0117	0.0048
BERT-SMLLM: mean	0.5788	0.6093	0.6272	0.6339	0.6294
BERT-SLLM: std	0.0058	0.0018	0.0144	0.0048	0.0065

Table 3: BERT on ART evaluated on validation set at each epoch for 10 epochs. BERT-base refers to the model without any further pretraining with SL-MLM and BERT-SLMLM refers to the model with further pretraining with SL-MLM.

Epoch	1	2	3	4	5	6	7	8	9	10
p-value	0.5393	0.1835	0.5220	0.1823	0.5027	0.1498	0.3947	0.1117	0.2001	0.2256
statistic	-0.670	-1.606	0.7007	1.6117	0.7355	1.7789	0.9526	2.0337	1.5324	1.4309

Table 4: p-values from Student's t-Test comparing the results between GPT-2-base and GPT-2-SLLM

Epoch	1	2	3	4	5
p-value	0.2861	0.1215	0.1072	0.2546	0.718
statistic	-1.229	-1.959	-2.070	-1.328	-0.387

Table 5: p-values from Student's t-Test comparing the results between BERT-base and BERT-SLMLM