Improving BERT for Natural Language Inference Task

NTU-CE7455 Project Final Report

Chang Jun Qing
School of Computer Science and Engineering
Nanyang Technological University
junqing001@e.ntu.edu.sg

Zhao Lin
School of Computer Science and Engineering
Nanyang Technological University
lin018@e.ntu.edu.sg

Abstract

Bidirectional Encoder Representation of Transformer (BERT) [1] is widely used nowadays in natural language tasks with Natural Language Inference(NLI) being one of the use cases. Training of these state-of-the-art models are often computationally expensive and done on datasets not available to the public. We present four main methods of improving BERT for NLI by finetuning using both data augmentation methods as well as architectural changes. Our model demonstrates an improvement on MultiNLI corpus test set in Match and Mismatch accuracy. Through this experiment, we found that both loss computation and choice of activation function are potential areas to improve model accuracy. There is still much space to improve for BERT model not only in pre-training but finetuning for downstream tasks.

1 Introduction

Transformer architectures like BERT, XLNet [2], and XLM [3] have brought significant improvements to many areas of natural language processing including NLI. Training of such models are however computationally expensive as it often requires long hours of pre-training on graphics processing units (GPU) with large amount of memory. The data that these models are pre-trained on also tend to be large and are usually private. This makes it hard for most to improve on these model as not many have access to such resources.

Inference on entailment and contradiction is a valuable benchmark test for semantic representation. Natural language inference can be applied to tasks such as information retrieval, semantic parsing and commonsense reasoning[4]. The major task of NLI is to determine whether a "hypothesis" is true (entailment), false (contradiction) or neutral given a "premise".

We argue that improving models for task such as NLI should not be limited by computational resources. Therefore in this paper we present 4 main approaches to improve BERT by finetuning. Our contributions are as follows:

- We demonstrate how by augmenting the dataset there is potential to improve the performance of a given task
- We show that even at finetune level, BERT can have some architectural change that can further improve the performance
- The code is available at `https://github.com/junqingchang/natural-language-inference`

2 Related Work

Robustly optimized BERT approach (RoBERTa) [5] is a variant of BERT that investigates 4 main changes to improve the results of BERT. Firstly, they investigated the use of dynamic masking instead of static masking, which prevents the same data from repeating multiple times with different mask, reducing pre-training time. Secondly, they investigated if removing the Next Sentence Prediction (NSP) loss from the training would improve the model as they found that inclusion of this loss will result in worse results in downstream task (such as NLI). Thirdly, they made use of larger batch sizes (compared to the original BERT model) as larger batch sizes have proven in recent papers to be beneficial for models [6]. Lastly, they investigated a different approach of Byte-Pair Encoding (a hybrid between character and word level representation) [7].

Another upgraded BERT model is called ALBERT, which is A Lite BERT for self-supervised learning of language representation. It improved the 12 NLP tasks state-of-the-art performance[8]. The success of ALBERT relies on the following optimization strategies. First, it allocates the model's resources more efficiently. It specifies the input-level embedding which has relative low dimension to learn the context free representation while the hidden layer embedding need to further confine the representation in context-dependent manner. Next, ALBERT uses Sentence Order Prediction (SOP) as pre-training loss comparing with BERT which uses Next Sentence Prediction (NSP) loss for pre-training. Sentence Order Prediction is a good task for constructing coherence and cohesion in discourse. According to the paper, sentence ordering is more challenging and efficient in certain downstream tasks than Next Sentence Prediction task. The paper presents that using SOP pre-training model for downstream task of "SQuAD1.1", "SQuAD2.0","MNLI" and "RACE", the performances are better than NSP pre-training. But in "SST-2" task, NSP is slightly better. The key takeway is that different pre-training loss task affects the finetuning result on downstream task, which is why we tried to implement it in our model. [9]

3 Approach

3.1 Bidirectional loss for sentence pair

Typically, cross entropy loss is used for MNLI task where we calculate the loss from the predicted label against the targets. In MNLI, the data provided contains 2 sentences and the normal approach would be to use $sentence_1 \rightarrow sentence_2$ as the input to the model to predict the label for the pair of sentence. This can be written as the following:

$$L_{cross-entropy}(f(sentence_1 \rightarrow sentence_2), \hat{y}) = -\sum_i y_i \log(f(sentence_1 \rightarrow sentence_2)_i)$$

However, MNLI is different from Next Sentence Prediction where the order of the sentences matter. Due to the nature of our task, we want our model to avoid learning like a Next Sentence Prediction Model and believe that we can augment the dataset by utilising the sentence pairs in both orders with the same target. This allows us to have a sum cross entropy loss of both ways of ordering and can be written as:

$$L = -\sum_i y_i \log(f(sentence_1 \rightarrow sentence_2)_i) - \sum_i y_i \log(f(sentence_2 \rightarrow sentence_1)_i)$$

This enables us to ensure that our model learns to optimise in both orderings.

3.2 Word Activation Masking

BERT models make use of Byte-Pair Encoding(BPE) which is an in-between of word encoding and character encoding. This can be thought of as sub-word encoding, which allows handling vocabularies common in natural language even when certain full words may not be part of the model's vocabulary. This also opens up to potential learning of relations between sub-words instead of full words by the model. Because of this, we decided to include another layer of masking into the embedding layer of the model which activates the first sub-word of each full word. This enables our model to know where every next word of the sentence is. Similar to how BERT's Whole Word Masking implemented in May 2019 that improved the performance of the model, we believe that this will help reduce the impact of relations between subwords. Table 1 shows an example of the masking.

Sentence	Lorem		ipsum			dolor		sit	amet	
Tokenised	Lo	##rem	i	##ps	##um	do	##lor	sit	am	##et
Masking	1	0	1	0	0	1	0	1	1	0

Table 1: Sample Word Activation Masking

3.3 Pre-finetuning using other datasets

Given the idea that BERT uses NSP, ALBERT uses SOP and RoBERT does not use any task to compute pre-training loss, we came up with the idea to experiment on different prediction task to pre-train BERT model. The task are not exactly language inference tasks but similar tasks that the model is required to learn and understand the context. Two corpuses are used for experiment in this session. They are Dialogue NLI (DNLI) corpus [10] and Microsoft Research Paraphrase (MSRP) corpus[11]. Both corpuses are well prepared for classification tasks. The details for corpuses will be introduced later. Since we do not have enough resources to pre-train the whole system. To implement this experiment, we treat them as additional finetuning corpuses before the actual finetuning on MNLI corpus. We hope the system can learn the parameters from more relevant resources before finetuning on specific downstream task.

3.4 Changing activation for attention layer and linear layer

BERT model is in general the encoder part of transformer. It contains the attention layer to select the context that better fit the current node instead of fitting all the contents into a vectors. In attention layer, softmax activation function plays an important role for attention mechanism. In the attention layer, the decoder does not use the inputs from all encoders. Instead, it selects the state that is best matched with current node by calculating compatibility score. To convert these scores into valid attention weights, we need to normalize them and sum the values to 1. This is achieved by softmax function. Softmax function also helps to enlarge the higher component and minimize the lower component. The CrossEntropy cost function is applied to compute the output from softmax function for classification or regression. There are studies that reveal the bottleneck for softmax activation. In the paper from [12], they proved that there is representational capability issue in language model when using softmax.They demonstrated that the softmax-based model is restricted by the length of the hidden nodes in output layer. [13]. Therefore, a few methods are proposed to replace softmax. They are 1)sigsoftmax which combine softmax with Sigmoid function. 2) weighted sum of softmax 3) ReLU and Sigmoid combination. We plan to try out sigsoftmax first followed by weighted sum of softmax.

Another activation function is located in pooler layer when BERT embedding and encoder are completed.The BERT pooler layer picks up only the first element from hidden state and pass it to the tanh() activation function. This step maps the encoder sequence from [batch_size, seq_length, hidden_size] to [batch_size, hidden_size] shaped vector. The "pool" action is performed by selecting the hidden state that corresponding to the first token. With the activation function transformation, it can do classification task for Masked LM and Next Sentence Prediction in pre-training using learned weights and biases. In practice, there is another linear layer after the pooler layer for downstream task finetuning. We plan to experiment on other activation functions.

4 Experiments

4.1 Data

The main dataset that we are using is the Multi-Genre NLI corpus [14]. This dataset focuses on natural language inference where the inputs are a sentence pair (the premise and the hypothesis) and the expected targets are 3 classes - neutral, contradiction and entailment. These 3 classes reflect if the sentences within a pair do not contradict each other, contradict each other, or if they support each other. Examples can be seen in Table 2.

Two additional datasets are used for pre-finetuning task. Dialogue NLI (DNLI) corpus focus on improving the consistency of dialogue model. It consists of sentence pairs labeled as entailment,

Premise	Label	Hypothesis
The Old One always comforted Ca'daan, except today.	neutral	Ca'daan knew the Old One very well.
yes now you know if if everybody like in August when everybody's on vacation or something we can dress a little more casual or	contradiction	August is a black out month for vacations in the company.
At the other end of Pennsylvania Avenue, people began to line up for a White House tour.	entailment	People formed a line at the end of Pennsylvania Avenue.

Table 2: Sample Examples from MultiNLI Website

Label	Sentence 1	Sentence 2
Negative	We did too but working in real estate for 12 years sucked up a lot of time .	We did too but working in real estate for fifteen years sucked up a lot of time.
Positive	I work with a lot of kids in the healthcare industry .	I work in the healthcare industry .
Natural	sounds very important . you must be a people person .	i am naturally a crabby person .

Table 3: Sample Examples from DNLI corpus

neutral or contradiction. [10] The paper demonstrates the dialogue consistency problem can be mapped to a natural language inference model and they train the proposed model using DNLI corpus. We believe that this DNLI dataset can make contributions to our model with NLI task. The original corpus contains 310110 pairs of sentences which takes very long to train. A filter is applied to select only sentences pairs that both length are lower than 32 characters as training dataset. After filtering, 58604 pairs are left to form the new training dataset. Examples of DNLI can be found in Table 3

Another corpus chosen for pre-finetuning is Microsoft Research Paraphrase (MSRP) Corpus[11]. It gives a pair of sentences to classify them as paraphrases or not paraphrases. The corpus consists of 4076 sentence pairs with 67.5% are positive (paraphrases). Samples of MSRP Corpus can be found in Table 4

Label	Sentence 1	Sentence 2
1	Amrozi accused his brother, whom he called "the witness", of deliberately distorting his evidence.	Referring to him as only "the witness", Amrozi accused his brother of deliberately distorting his evidence.
0	Yucaipa owned Dominick's before selling the chain to Safeway in 1998 for $2.5 billion.	Yucaipa bought Dominick's in 1995 for $693 million and sold it to Safeway for $1.8 billion in 1998.

Table 4: Sample Examples from MSRP corpus

Lastly, we also made used of the Cornell Newsroom dataset [15] for pre-training purposes. The Newsroom dataset provides news articles and their respective summaries and is often used for training and evaluating summarization systems. For our purpose, we are using the articles from the dataset to pre-train our edited BERT models as these articles provide us with paragraphs of sentences which are similar to the Wikipedia dataset and BookCorpus dataset used to pre-train the original BERT model.

Sizes of all the datasets we used can be found in Table 5

4.2 Evaluation method

For our evaluation, we used the Match Test Accuracy and Mismatch Test Accuracy. These can be calculated by finding the accuracy of our model with the provided Match Test dataset and Mismatch

Dataset	Size
Multi-Genre Natural Language Inference (MultiNLI)	433k Sentence Pairs
Filtered Dialogue NLI (DNLI)	58604 Sentence Pairs
Microsoft Research Paraphrase (MSRP) Corpus	4076 Sentence Pairs
Cornell Newsroom	4 GB (1.3m Articles)

Table 5: Dataset Sizes

Test dataset in MNLI. In addition, we are evaluating our model by comparing the performance with BERT as we expect our model to improve BERT in the area of natural language inference. One of the constraints we had was limited computational resources. As such, instead of comparing our model with the large BERT model (hidden size 1024), we compared our model with the base BERT model (hidden size 768).

4.3 Experimental details

For our experiments, we built on top of huggingface's transformers library as they have classes made for BERT models that we can inherit and make changes from. All our finetuning experiments, unless otherwise specified, ran with the "bert-base-cased" model configuration which has output of 768 hidden dimensions, learning rate of $3e-5$ for 3 epochs and an Adam optimizer. We are also using a batch size of 8 for all our experiments due to limited GPU memory.

For our hardware configuration, we made use of instances on Paperspace. Each of these instance have 8 vCPUs from an Intel Xeon, and are equipped with an Nvidia Quadro P5000 with 16GB of memory.

4.3.1 Baseline BERT

For our baseline BERT, we used a pre-trained "bert-base-cased" model from huggingface and added a linear layer on top of it to get our desired 3 outputs. This model was then finetuned on the MNLI dataset to obtain the baseline results for comparison.

The total training time taken was approximately 3 hours an epoch (9 hours in total).

4.3.2 Bidirectional loss for sentence pair

Bidirectional loss for sentence pair requires no changes to the model itself, but instead we augmented the data. For this experiment, we made our dataset provide the tokenised inputs, token type ids, and attention mask for both $sentence_1 \rightarrow sentence_2$ and $sentence_2 \rightarrow sentence_1$ at the same time with the given target for the sentence pair. Next, predictions were obtained for both inputs and the cross-entropy loss is calculated and added together. Finally, the model updates its weights by summing the losses.

The total training time taken was approximately 5 hours an epoch (15 hours in total).

4.3.3 Word Activation Masking

Word Activation Masking requires us to make changes to the BERT architecture. We did this by changing the embedding layer of BERT. The original BERT has three main components to it's embedding layer - word embedding, position embedding, and token type embedding. These embeddings are summed and returned. We added an additional word activation embedding that is also summed to the other three embeddings. For us to obtain the word activation mask, we also altered the tokeniser such that for every input that we take in, we find each first subword and make that position 1 for the word activation mask. Every other subword is a 0.

The total training time for finetuning with the architectural change was approximately 3 hours an epoch (9 hours in total).

For this experiment, we also attempted to pre-train our own model. It was trained on the Cornell Newsroom Dataset, where we take the articles and split them into sentences. These sentences are then sent into the model as input for pre-training.

For pre-training, we define 1 iteration as each batch of 8 data sent to the model. We tested the model at 1 million iterations as well as 6 million iterations. The total pre-training time was approximately 240 hours for 6 million iterations.

4.3.4 Pre-finetuning using other datasets

This experiment requires a linear layer for classification. Changes need to be made for the classes since original finetuning using MNLI corpus requires 3 classes. But MSRP dataset requires only 2 classes. Sentences were pre-processed using same embeddings method as MNLI dataset. We trained both dataset using same configurations as other experiments. After the training, we saved the training parameters and loaded these parameters again for MNLI dataset finetuning.

The total training time taken was approximately 2 hours to pre-train on the MSRP dataset, 5 hours for DNLI, and 9 hours an epoch to finetune for MNLI after.

4.3.5 Changing activation in attention layer and pooler layer

As explained in section 3.4, we did the experiments for different activation function at attention layer and pooler layer. For pooler layer, we replaced the tanh function with Sigmoid and ReLU function. Tanh and Sigmoid are widely adopted activation functions. They own similar characteristic. Tanh has deeper gradient. ReLU becomes more and more popular in deep learning. It is more computationally effective. ReLU works most of time as a general approximator.

For attention layer activation, we implemented the sigsoftmax function with a reference from [16]. As the first step, we tried to replace softmax with only Sigmoid activation. The training loss kept high and non-decreased with more training iteration. Match and Mismatch accuracy for MNLI corpus are very low.

The total training time taken was approximately 3 hours an epoch (9 hours in total).

4.3.6 Combination

For our final experiments, we ran two combinations of our individual experiments. Our first combination consists of using the Bidirectional Loss, Sigmoid Activation, as well as the Word Activation Masking. For this experiment, we made use of our edited BERT embedding layer, edited BERT pooler layer, and made used of the augmented dataset used for Bidirectional Loss of sentence pair.

Our second combination was running only Bidirectional Loss for sentence pair with Sigmoid Activation. Similar to the previous experiment, we made use of edited BERT pooler layer as well as augmented dataset.

The total training time was approximately 5 hours an epoch (15 hours in total) for each composite experiment.

4.4 Results

The results for experiments can be found in Table 6

In general, the four tasks achieve an improvement in accuracy comparing to BERT(base) model in terms of Match Accuracy, with Sigmoid Activation improving Mismatch Accuracy as well. This is more than our expectation. According to the table, the best result is produced using finetuning on MNLI corpus with Sigmoid activation function in BERT pooler layer. Computing sentence pair bidirectional loss and word activation masking slightly improve the Match accuracy. Pre-finetuning using additional corpuses shows the accuracy difference in MSRP and DNLI. It indicates this is a potential area to improve the model on.

We expect to obtain the best result when combining all the useful actions together in finetuning part as all of them are outperformed individually. However, the results indicate otherwise. We believe this is due to too many changes to the model such that finetuning itself is insufficient for the model to learn and improve the results further.

Model	Match Accuracy	Mismatch Accuracy
BERT (base model)	79.50	80.79
Finetuning with + Bidirectional Loss	**80.10**	80.35
Finetuning with + Word Activation Masking	**80.02**	80.30
pre-training Word Activation Masking (1000000 iterations)	31.82	31.82
pre-training Word Activation Masking (6000000 iterations)	35.45	35.22
Finetuning with + DialogueNLI Corpus	78.67	79.56
Finetuning with + MSRP Corpus	**80.08**	80.45
Finetuning with + Sigmoid Activation	**80.20**	**80.91**
Finetuning with + ReLU Activation	**80.11**	79.72
Finetuning with + Bidirectional Loss + Sigmoid Activation + Word Activation Masking	**80.19**	80.19
Finetuning with + Bidirectional Loss + Sigmoid Activation	**80.05**	80.46

Table 6: Experiment Results

5 Analysis

Among the four tasks, there are some interesting findings that can be further analyzed.

1. Why did most finetune task improve the Match Accuracy but not Mismatch Accuracy?
The difference between Match Accuracy and Mismatch Accuracy is that Match Accuracy comes from dataset which are from the same source as those in the training set while Mismatch test set has sources that are different from those in the training set. [14] This suggests that most of our experiments in finetuning helped the model fit better to the training set but makes it less robust to sources it has yet to see.

2. Why did the remaining two combination not included in the edited loss?
Bidirectional Loss can be further improved to include the remaining combinations. However, we wanted to test out our hypothesis with the simplest base case. We believe that the optimal loss function would be the following:

$$L = -\sum_{j}^{2}\sum_{k}^{2}\sum_{i} y_i \log(f(sentence_j \rightarrow sentence_k)_i)$$

3. Would pre-training the Word Activation Masking be better?
As mentioned in section 4.3.3, we attempted to pre-train the word activation masking. Pre-training has the potential to improve the performance. However as seen from our experiments, even at 6 million iterations, the model was only around 35% accuracy. We believe that to fully test this hypothesis, we need to use the same dataset used to train the original BERT as well. We believe that by finetuning on this modified architecture, we can demonstrate the potential of identifying full words.

4. What causes the non-decreasing loss when changing activation in attention layer comparing to changing activation in pooler layer?
As discussed in section 3.4, BERT pooler layer is an addition to the system for NSP and Masked LM task. It makes use of the completed set of pre-trained weights and biases. Therefore, even it is modified, the system are still able to make use of the trained parameters to continue training. However, for the activation in attention layer, there are two possible causes of failures. Firstly, some sentence pairs modify the trained parameters to an uncontrollable range. Another reason could be the attention layer is closely related to pre-training the encoder. Therefore, the changes in activation function will disrupt the trained parameters.

5. Why is the result from pre-finetuning with DNLI corpus worse than MSRP?
Dialogue NLI is another type of NLI which is expected to have a better result than MSRP. However, from Table 5 we can see the result is even lower than BERT base model. During loading of dataset, a filter is applied to select sentence pairs where both sentences' length are shorter than 32 characters. The reason to do filtering is that the original dataset contains 310110 training pairs which is computationally expensive. After filtering, 58604 pairs are left to form the new training dataset. Although all the three classes take roughly the same percentage in the training set, the sentence's length are generally shorter compared to MNLI or MSRP corpus. This is one possibility that Match and Mismatch accuracy on MNLI with pre-finetuning on DNLI are lower than pre-finetuning on MSRP or BERT base model.

6. Why Sigmoid outperforms tanh and ReLU in BERT Pooler layer?
Sigmoid has similar characteristics with tanh function. Sigmoid function belongs to range [0,1] and tanh function has the range [-1,1]. Both method have the gradient vanishing issue. ReLU has the advantage of less computation load compared with Sigmoid and tanh. However, because of the zero value region in the left, it means the learning is not happening in that area as negative values are not mapped appropriately, and it loses some accuracy. In this experiment, Sigmoid shows better accuracy result than tanh. One possible reason is the classification task needs a smoother transition. And BERT model is probabilistic which would benefit more from Sigmoid as compared to tanh.

6 Conclusion

In this work, we researched the basic BERT operation, analyzed two state-of-the-art models - RoBERTa and ALBERT. After investigating their successful factors, we came up with four possible areas that have potential for experimentation. They are (1) bidirectional loss of sentence pair (2) word activation masking (3) pre-finetuning with relevant corpus (4) changing activations in Attention and Pool layer. We expect to replicate the accuracy stated in BERT paper, RoBERTa paper then generate better Match and Mismatch accuracy on MNLI corpus after taking the above measures.

We encountered the resource issue as the original BERT(base) is pre-trained on 16 TPU chips and BERT(large) model is pre-trained with 64 TPU for 4 days for each training. In addition, the unsupervised pre-training is conducted on large number of dataset that is not entirely available online. Therefore, we experimented with methods that are independent to pre-training. Table 6 shows the improvement in Match and Mismatch accuracy.

Through this experiment, we found that BERT model still has much area to improve in both pre-training part and finetuning part. We believe that the finetune experiments we ran demonstrated that there is potential for data augmentation and architectural changes to future improve the BERT model. Furthermore, we are only able to run the experiments in relatively smaller batch sizes which would likely affect the results slightly as compared to larger batch sizes. As future works, if there is no resource limitation, we believe that pre-training with changes of activation in attention layer and word activation masking are good experiments to run.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[2] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv:1906.08237*, 2020.

[3] Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *arXiv:1901.07291*, 2019.

[4] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[5] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv:1907.11692*, 07 2019.

[6] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv:1904.00962*, 2020.

[7] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv:1508.07909*, 2016.

[8] Zhenzhong Lan Radu Soricut. Google ai blog: Albert: A lite bert for self-supervised learning of language representations, dec 2019.

[9] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2019.

[10] Sean Welleck, Jason Weston, Arthur Szlam, and Kyunghyun Cho. Dialogue natural language inference. *CoRR*, abs/1811.00671, 2018.

[11] Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*. Asia Federation of Natural Language Processing, January 2005.

[12] Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. Breaking the softmax bottleneck: A high-rank rnn language model. *ArXiv*, abs/1711.03953, 2017.

[13] Sekitoshi Kanai, Yasuhiro Fujiwara, Yuki Yamanaka, and Shuichi Adachi. Sigsoftmax: Reanalysis of the softmax bottleneck. 05 2018.

[14] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018.

[15] Max Grusky, Mor Naaman, and Yoav Artzi. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 708–719, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[16] Pankesh Bamotra. Pytorch implementation of sigsoftmax, may 2019.