Evaluating Transformer Architectures for Language Understanding

Aashish Yadavally
Institute of Artificial Intelligence
University of Georgia
Athens, GA - 30602
ayadavally@uga.edu

Sumer Singh
Institute of Artificial Intelligence
University of Georgia
Athens, GA - 30602
sumer.singh@uga.edu

Abstract

Natural Language Understanding (NLU) is a core domain of Artificial Intelligence (AI). In recent times, there has been a rapid advance in the NLU capabilities of AI systems. These systems have achieved excellent results on basic NLU tasks such as sentiment analysis. We study the performance of advanced AI systems on complex NLU tasks. Our study involves testing several different models, followed by fine tuning the best performing one. We choose reading comprehension as the task and use the SQuAD dataset for our primary task. For our secondary task, we test our model performance on 23 Graduate Record Examinations (GRE) questions.

1 Introduction

Reading Comprehension is a fundamental language skill possessed by humans. Any machine required to emulate human language tasks must have excellent reading comprehension ability. Specifically, a reading comprehension task includes two things - 1) a context, and 2) questions. The context contains all the information required to answer the questions and is normally in the form of one or more paragraphs of text. The questions are based on the given context and must be answerable using the contextual information. To successfully tackle a reading comprehension task, the machine (or human) requires several skills. Formally, the fundamental skills required in efficient reading comprehension include knowing meanings of words, understanding word context, draw inferences from a passage, ability to answer questions answered in a passage and ability to draw inferences about the writer (1).

As we can see, the first and most fundamental skill required in reading comprehension is to know the meaning of words. Currently the state-of-the-art in word meaning embeddings is transformer-based (2) models, such as BERT (3), XLNet (4), ALBERT (5), etc. These models are pretrained in an unsupervised fashion on very large amounts of text data. The pretraining is general and done for any language task. It allows the model to learn the basic and contextual meaning of words. Once pretraining is complete, the model is trained in a task-specific manner. The step, known as downstream training, will differ from task to task and is done in a supervised fashion.

2 Related Work

For words to be able to be used in machine learning frameworks, they need to be represented as a sequence of numbers. This led the NLP community to build vector representations using techniques such as word2vec(6) and GloVe(7). Using these techniques, pre-trained embeddings of words could be used to represent the words in the NLP task. The problem with using such word embeddings was that, each word could have different meanings when used in different contexts, and using the

same vector representation for the word in every context in the NLP task wouldn't capture much contextual information. Thus, pretraining techniques such as Embeddings from Language Models (ELMo)(8) were introduced, which would learn the vector representations for the words by looking at the entire sentence, with the help of bidirectional Long Short Term Memory Units (LSTMs)(9). Using contextualized word-embeddings like ELMo led to a significant step forward in the field of Natural Language Processing, towards using transfer learning for varied tasks - by pretraining on a massive dataset in the language, so as to use components of it for specific tasks.

2.1 Approaches to Transfer Learning

Throughout the history of the field, major improvements in the NLP tasks was observed through some forms of transfer learning: from semi-supervised learning for structural learning (10), to pretrained word-vector embeddings and pre-trained language models. Learning general language representations for further application to specific tasks was one of the more successful transfer learning approaches recently used in NLP. Techniques like ELMo relied on unsupervised approaches to learn context-sensitive feature representations of words, with the biggest difference with the previous bidirectional approaches being that they allowed sharing of parameters between the LSTMs in both the forward and backward directions. However, sequential transfer learning is the one that led to the biggest improvement in performance across tasks. (11) While the former approaches revolved around learning word representations (with and without context), this paradigm revolves around learning the language model for the data. In the process of learning language models, the models end up learning the syntax, semantics and information about the data for the task.

2.2 Transformers

The introduction of transformers by Vaswani et al(2) led to a major paradigm shift in the field of language understanding, with these sequence-transduction models outperforming complex recurrent networks and convolutional neural networks drastically. Mainly, they were able to learn the global long-term dependencies between the inputs and the outputs better than any other previously known networks. In a nutshell, transformers have two essential components - multiple encoders and decoders linked to each other. A third component which ties everything together is known as the attention mechanism. The attention mechanism is basically a set of weights, which tells the network how important an input is. This encoder-decoder structure made transformers very well suited, in particular, for machine translation tasks. As Radford et.al (12) observed, large gains on these language understanding tasks can be realized by generative pre-training of a language model on a diverse corpus of unlabeled text, followed by discriminative fine-tuning on each specific task. They used a language modeling objective on the unlabeled data to learn the initial parameters of a neural network model, subsequently adapting these parameters to a target task using the corresponding supervised objective. Thus, this was achieved by just using the decoder for language modeling, and building a stack of twelve decoder layers. However, this technique only took into consideration the forward direction, giving rise to Bert, and consequently, a number of it's variants, a few of which will be discussed in the further subsections.

2.2.1 BERT

BERT, which stands for 'Bidirectional Encoder Representation from Transformers' makes use of the attention mechanism in the Transformers architecture to learn the contextual relations between words in a text. Since BERT's primary goal is to generate a language model, it is sufficient to use the encoders in the attention mechanism. While the directional models read the text in a particular direction (left-to-right or right-to-left, or a combination of both), BERT inputs the whole text as a sequence, because of which it can be considered bidirectional. In a nutshell, the BERT architecture replaces a certain percentage of the tokens in the text sequence with masks, with the goal of predicting the words replaced by the masks in the context of the remaining unmasked words.

The authors proposed two variants of BERT, BERT-base and BERT-large, with the difference in both the models being the number of parameters involved due to the difference in the number of attention layers. The former has 12 attention layers with up to $110M$ parameters, while the latter has 24 attention layers with up to $340M$ parameters. In order to finetune BERT for other classification tasks,

an additional classification/softmax layer can be added on top of the [CLS] node in the transformer. The introduction of this architecture was considered to be revolutionary, with the BERT-large model outperforming all the other models in most NLP benchmark tasks.

2.2.2 DistilBERT

Sanh et al (13) proposed DistilBERT, operating with a constrained computational training, achieving 97% of BERT's language representational capabilities and training 40% faster. The authors use the concept of knowledge distillation, to compress and train a compact model (referred to as student) reproducing the behavior of the larger model (referred to as teacher). While the architecture of DistilBERT retains the general architecture of BERT, the token-type embeddings and pooler layers are removed and the number of layers is reduced by a factor of 2. The former is done on the basis of investigations that the last dimension of the tensor has less impact on the computational efficiency. The architecture of DistilBERT also takes advantage of the common dimensionality between the student and teacher networks. Furthermore, DistilBERT is distilled on very large batches leveraging gradient accumulation, using the concept of dynamic masking and without the next sentence prediction objective, which is the basis of BERT pretraining. The authors also propose a general purpose distillation technique as compared to the task-specific distillation techniques followed by Tang et. al (14) and Chatterjee (15).

2.2.3 XLNet

XLNet (4) is a BERT-like model, which is a generalized autoregressive pretraining method which caught the attention of the NLP community by outperforming BERT on 20 NLP downstream tasks. XLNet is an autoregressive model which predicts the next word depending on a context word, with the context word being present in either direction, i.e, forward and backward. It aims to reconstruct the original data from the input in which [MASK] is used to replace the original token during the pretraining phase. Despite the superior performance, XLNet has it's fair share of disadvantages as well. First, the [MASK] token which is used to replace the original tokens during the pretraining phase is not present during the finetuning process for the downstream task, resulting in a pretrain-finetune discrepancy. Second, XLNet assumes that the predicted tokens are independent of each other given the unmasked tokens.

2.2.4 ALBERT

While it has commonly been observed that increasing the size of the model at the pretraining step leads to an improvement in the learning process of language representations over a wide array of downstream tasks, due to the GPU/TPU memory limitations, there is a necessity to find networks which require fewer parameters. ALBERT, proposed by Google Research and still in review at ICLR 2020, caters to this requirement by employing a variety of techniques to enable performance enrichment, alongwith a drastic decrease in the number of parameters, and consequently finetuning time. One of the techniques that ALBERT follows towards decreasing the number of parameters is factorized embedding parameterization. This technique involves decomposing the large vocabulary embedding matrix into two small matrices, thus separating the size of the hidden layers from the size of the vocabulary embedding. Another technique that ALBERT incorporated is that the network supports cross-layer parameter sharing, resulting in smoother transitions from one layer in the network to another.

As noted earlier, in BERT as well as XLNet, the size of the WordPiece embeddings and the size of the hidden layers were tied together, i.e, they were maintained to be the same across the network. However, for ALBERT, the authors untie this relationship and maintain the length of the WordPiece embeddings to be much smaller than the number of hidden units.. This helped the authors make efficient use of the number of parameters. They use a factorization of the embedding parameters matrix by projecting it into a lower dimension, and maintaining it across all the word pieces.

These changes in the architecture allowed the authors to scale up the number of hidden layers and units in the model. The authors primarily worked with three versions of ALBERT, i.e, ALBERT-large, ALBERT-xlarge and ALBERT-xxlarge. For example, ALBERT-large has 18M parameters as compared to the 334M parameters in BERT-large, with $H = 2048$ number of hidden units.

For a similar configuration, BERT-large would end up having ~1.27B parameters. Additionally, ALBERT-xlarge has ~60M parameters for a hidden layer with 2048 units, and ALBERT-xxlarge has ~233M parameters for a hidden layer with 4096 units. Despite having much fewer parameters than BERT-large, ALBERT-xxlarge is still more computationally expensive (in terms of time), and the authors propose extending ideas such as sparse attention and block attention to improve the computational expense of ALBERT.

2.3 Datasets for Language Understanding

2.3.1 SQuAD-2.0

	Version 1	**Version 2**
Train		
Total examples	87599	130319
Negative examples	0	43498
Total articles	442	442
Articles with negatives	0	285
Development		
Total examples	10570	11873
Negative examples	0	5945
Total articles	48	35
Articles with negatives	0	35
Test		
Total examples	9533	8862
Negative examples	0	4332
Total articles	46	28
Articles with negatives	0	28

Table 1: Dataset Statistics of both iterations of SQuAD.

The Stanford Question Answering Dataset (SQuAD) (16) is a Reading Comprehension dataset. It consists of crowd-sourced questions based on Wikipedia articles. The answers to the question must be a span of the article (i.e. a subset of words from the article itself). No paraphrasing is required. The first iteration/version of SQuAD consisted 100,000+ question-answer pairs on 500+ articles. The second iteration/version added 50,000 'negative' questions. Negative questions are questions which do not have an answer. Thus the model must refrain from answering. If any answer is given, it will be deemed incorrect. This update makes the dataset highly challenging and requires the model to correctly identify unanswerable questions. Details are displayed in Table 1.

2.3.2 RACE

	Train	**Dev**	**Test**	**Total**
passages	25137	1389	1407	27933
questions	87866	4887	4934	97687

Table 2: Dataset Statistics of RACE

RACE is a large-scale, multiple-choice, reading-comprehension dataset proposed by Lai et al (17). It consists of almost 100,000 questions based on 28,000 articles. The material is collected from English exams in China and consists of two levels - middle and high school. As the dataset is collected from school examinations, it has the benefit of being created by domain experts. There are 4 choices as an answer for each question, and each question has exactly one answer. The options are not required to be a span of the article. Details are displayed in Table 2.

There are two main differences between SQuAD and RACE. Firstly, answers in the SQuAD dataset are a span of the article, while in RACE they can be paraphrased. Next, SQuAD questions might not have an answer, while all questions in RACE have an answer.

2.3.3 GRE

The Graduate Record Examination (GRE) is an examination all graduate students must give. The verbal part of GRE consists of several reading comprehension questions, although their difficulty is much higher than the RACE questions. GRE reading comprehension questions contain 5 options, with exactly one correct answer. Thus, the format is very similar to questions from the RACE dataset. We constructed a short GRE dataset using material available online with 23 questions based on 5 articles/passages. The shortest article in the dataset consists of 138 words and the longest article consists of 453 words.

3 Methodology

3.1 Primary Task

We explored several state of the art architectures, such as BERT(3), XLNet(4), DistilBERT(13) and ALBERT(5). Initially, we tested the models with the default hyperparameters. If the model showed promising results, we performed hyperparameter-tuning to improve it's performance. The hyperparameters tuned include:

1. Learning Rate - decides how big a step the model would take during gradient descent. A large learning rate can cause the model to not converge, while a slow learning rate could cause the model to converge very slowly.
2. Max Sequence Length - decides how many tokens from the input will be processed. An input larger than max sequence length with be truncated. A higher max sequence length will capture more information but will also use more memory.

The metrics used to test the performance of the SQuAD dataset are Exact Match (EM) and F1-Score (F1). These scores are calculated between the predicted span and the actual span. EM for a sample will be 1 if the predicted span exactly matches the true span (otherwise 0), while F1-Score will range between 0 and 1 depending on the number of correctly predicted words. Naturally, EM score will be lower or equal to the F1-score.

3.2 Secondary Task

As the SQuAD dataset only contains options which are a span from the passage, which is not the case with GRE, for which the answers are multiple choice - it isn't possible to test the performance of the models trained on this dataset on the GRE questions. Thus, we used the RACE dataset for training on this task. Another issue is that RACE has 4 possible answers for each question, while the general format of the GRE Reading Comprehension questions have 5 options. To tackle this, we randomly delete one incorrect option from our GRE dataset, to make it in line with the RACE dataset. For modelling purposes, we choose the best performing model from the primary task. Once the selected model is trained on the full RACE dataset (high + middle school questions), we used transfer learning and test it on GRE.

The metric used in this task is Accuracy. For a sample, if the predicted option matches the correct option, a score of 1 is given, otherwise 0.

4 Experimentation and Results

4.1 Setup

Our initial models, where we used the default hyperparameters as described in the introductory publications of these models, were run locally on a GTX 2080Ti (11GB CUDA memory). After identifying the better performing models, larger models were used in the finetuning stage, which also had more memory requirements. Thus for these models, we made use of a Titan V (16GB CUDA memory) through Google Cloud Platform.

4.2 Gradient Accumulation

Gradient Accumulation means accumulating the gradient updates of a fixed number of mini-batches before actually updating the gradient. It is extremely useful when very small batch sizes have to be used due to the memory limitations. Our models use gradient accumulation. Thus, we could retain the performance of the models despite using smaller batches for finetuning for the downstream task.

4.3 Primary Task

As a starting point, we test our model with the base version of BERT, base version of DistilBERT, base version of DistilBERT with distillation during fine tuning (DistilBERT-base-distilled-squad), base version of XLNet and base version of ALBERT. We use the following default parameters for all these models:

1. learning rate - $5 * 10^{-5}$
2. batch size - 8
3. maximum sequence length - 384
4. number of epochs - 3

Model	**Parameters**	**EM / F1**	**Speedup**
Bert-base	110M	69.61 / 73.02	1.00x (3.5 hours)
DistilBERT-base	66M	57.08 / 60.69	1.3x
DistilBERT-base-distilled-squad	66M	57.52 / 61.23	1.3x
XLNet-base	112M	36.07 / 41.29	0.7x
ALBERT-base	12M	76.1 / 79.0	1.4x

Table 3: Performance of base models with default parameters on SQuAD dataset

The results are shown in Table 3. Out of these five models, only BERT and ALBERT beat the baseline score of 63.4% EM and 66.3% F1, while the other three performed poorly. Additionally, we observe three key advantages of ALBERT over BERT. First, ALBERT uses almost 10x fewer parameters which means it has much better memory requirements. Next, it offers a speedup of ~1.4x over BERT. Lastly, in spite of using less memory and training faster, it also has higher performance. Keeping these three factors in mind, we went ahead with tuning the parameters of ALBERT for achieving an improved performance.

4.3.1 Fine Tuning

Model	**Parameters**	**Learning Rate**	**Max Sequence Size**	**Epochs**	**EM / F1**
ALBERT-L	18M	$5 * 10^{-5}$	384	3	77.1 / 79.2
ALBERT-L	18M	$5 * 10^{-5}$	400	3	77.3 / 80.1
ALBERT-L	18M	$5 * 10^{-5}$	420	3	77.9 / 80.9
ALBERT-L	18M	$5 * 10^{-5}$	500	3	78.4 / 81.3
ALBERT-XL-V1	60M	$5 * 10^{-5}$	80	2	72.3 / 74.7
ALBERT-XL-V1	60M	$5 * 10^{-5}$	100	2	71.6 / 72.9
ALBERT-XL-V1	60M	$1 * 10^{-5}$	80	2	80.3 / 83.1
ALBERT-XL-V1	60M	$1 * 10^{-5}$	100	2	81.6 / 83.3
ALBERT-XL-V1	60M	$1 * 10^{-5}$	120	2	82.1 / 85.1
ALBERT-XL-V2	60M	$1 * 10^{-5}$	120	2	83.05 / 86.3

Table 4: Results of hyperparameter-tuned ALBERT models on the SQuAD dataset

For fine tuning, we start off with the 'Large' version of ALBERT (ALBERT-L). ALBERT-L has 50% more parameters than ALBERT base. We observe an increase of ~1 point in EM and ~0.2 points in F1 over ALBERT-base. We note that increasing max sequence size corresponds to an increase in EM and F1 scores. Unfortunately, we run into out of memory issues when max sequence size is greater than 500. Albert-L with max sequence size of 500 gives an increase of ~1.3 points EM and ~2.1

points F1 over ALBERT-base. We believe that performance of ALBERT-L can further be increased if more memory was available.

We then switch to Extra Large ALBERT (Albert-XL-V1), which contains 5 times more parameters than ALBERT-base. To conform to memory requirements, we drop the max sequence length to 80 and train for 2 epochs as each epochs takes more than 8 hours. Initially, a severe drop in performance is observed. ALBERT-XL-V1 loses ~3.8 points of EM score and ~4.3 points of F1 score compared to ALBERT-base. Increasing the max sequence to 100 further reduces performance. Thus, we revert to max sequence length of 80 and drop the learning to $1 * 10^{-5}$, which leads to a notable increase in performance. ALBERT-XL-V1 gains ~4.2 points of EM score and ~4.1 points of F1 score over ALBERT-base. We continue increasing max sequence length till memory permits. The best score of ALBERT-XL-V1 is 82.1 EM and 85.1 F1 with learning rate of $1 * 10^{-5}$ and max sequence length of 120.

Finally, we test ALBERT-XL-V2. This is the Extra Large version of Albert, but during pretraining no dropout is used. ALBERT-XL-V2 gives us our overall best score of 83.05 EM and F1 score 86.3.

4.4 Secondary Task

Model	**Dataset**	**Accuracy**
Albert-XL-V2	RACE (dev)	68.90%
Albert-XL-V2	GRE	43.47%

Table 5: Secondary Task Results on RACE and GRE datasets

Our secondary task consists of testing our model on 23 GRE questions. We select the best model from the previous task (ALBERT-XL-V2) and use the same hyperparameters. As a first step, we must retrain the model on the RACE dataset. To check if our model is training correctly, we evaulate it on the RACE (dev) set. It achieves a score of 68.9%. The current top score on the RACE dataset is 89.4%, achieved through an ensemble of ALBERT models. Thus we conclude our model is performing well enough.

	Article 1	**Article 2**	**Article 3**	**Article 4**	**Article 5**
Length	138	429	163	151	456
Total Questions	4	6	2	3	8
Correctly Answered	3	1	2	2	2
Accuracy	75.00%	16.67%	100.00%	66.67%	25.00%

Table 6: GRE dataset results

On the GRE dataset, our model gets 10 / 23 questions correct, thus scoring 43.47%. We believe this score is very good, considering the difficulty of GRE questions, in comparison to the high-school and middle-school level questions used in training the model. Additionally, no hyperparameter tuning was performed for this step, and it is possible to further improve the results.

We perform an article-wise analysis on the GRE datasets. Articles can be divided into two categories, short (between 138-163 words) and long (between 429 - 456) words. There are 3 shorts articles and 2 long articles. The model gets 7 / 9 questions correct on the short articles and 3 / 14 questions correct on the long articles. We can make an obvious conclusion from these results - the max sequence length is too small to capture enough information from the long articles. An increase in max sequence length should lead to better performance on the long articles.

5 Discussion

Our experiments highlight the effectiveness of transformer based NLP models, while also underscoring the hardware limitations of these techniques. BERT has proved highly effective in

the NLU domain, but comes with very high time and memory requirement. ALBERT significantly reduces the memory requirements but still has high time complexity. Due to this, our best performing model was only trained for 2 epochs (with each epoch taking around 8 hours) and with a max sequence length of 120. We believe that this model is still underfitting the data and results can further be improved, given more time and better hardware.

The effect of a crucial parameter - max sequence length - is clearly observed on the primary, as well as the secondary task. Intuitively, this makes sense, because by truncating sequences which are larger in length than the max sequence length, we might risk losing contextual information. On the other hand, too big a max sequence length can add more noise than information, because shorter sentences will be padded with zeros. We believe there exists a certain threshold after which increasing max sequence length will start degrading the performance. Given the computational resources and time, it would be interesting to observe this change. In addition, we observe promising results on the GRE dataset. In spite of an overall accuracy of 43.47%, we see a much better performance (77.77%) on the shorter articles. The lengths of the shorter articles (138-163 words) is much closer to the max sequence length of 120, thus facilitating a fairer evaluation.

6 Conclusion

This project studied the performance of several different architectures, including BERT, DistilBERT, XLNet and ALBERT on the SQuAD Dataset. We find ALBERT to perform significantly better than the other, while also having lower memory requirements. Through the course of the project, we tried to understand the trade-off between memory requirements and performance. A key factor in this trade-off was found to be the model size and the max sequence length (the maximum number of input words the model will process). In our experiments, increasing max sequence length and model size corresponded with an improvement in performance, but also an increase in memory consumption. We also note that increasing the model size, at the cost of decreasing max sequence length, leads to better performance. We believe that for the configurations we experimented with, our model is still underfitting the data, because increasing model size generally improves performance. Hence, training the model further should lead to a further improvement in results.

One interesting aspect to explore would be in trying to study the representations being learned for the sentences. It would be interesting to study the representations of two syntactically similar sentences, two semantically similar sentences, and in general, sentences which are both syntactically and semantically similar. Also, in the future, we hope to test larger models such ALBERT-XXL and fine tune its performance, so as to improve our results.

Acknowledgments

We would like to thank Dr. Sheng Li for his continued guidance and inputs through the course of the project. We are also grateful to the Institute of Artificial Intelligence for the lab resources, which were used to train these models with a high computational requirement.

References

[1] Davis, F.B. Psychometrika (1944) 9: 185. https://doi.org/10.1007/BF02288722

[2] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems. 2017.

[3] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[4] Yang, Zhilin, et al. "XLNet: Generalized Autoregressive Pretraining for Language Understanding." arXiv preprint arXiv:1906.08237 (2019).

[5] Lan, Zhenzhong, et al. "Albert: A lite bert for self-supervised learning of language representations." arXiv preprint arXiv:1909.11942 (2019).

[6] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.

[7] Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

[8] Peters, Matthew E., et al. "Deep contextualized word representations." arXiv preprint arXiv:1802.05365 (2018).

[9] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[10] Ando, Rie Kubota, and Tong Zhang. "A framework for learning predictive structures from multiple tasks and unlabeled data." Journal of Machine Learning Research 6.Nov (2005): 1817-1853.

[11] Ruder, Sebastian. "Neural Transfer Learning for Natural Language Processing." Diss. National University of Ireland, Galway, 2019.

[12] Radford, Alec, et al. "Improving language understanding by generative pre-training." URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf (2018).

[13] Sanh, Victor, et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." arXiv preprint arXiv:1910.01108 (2019).

[14] Tang, Raphael, et al. "Distilling Task-Specific Knowledge from BERT into Simple Neural Networks." arXiv preprint arXiv:1903.12136 (2019).

[15] Chatterjee, Debajyoti. "Making Neural Machine Reading Comprehension Faster." arXiv preprint arXiv:1904.00796 (2019).

[16] Rajpurkar, Pranav, Robin Jia, and Percy Liang. "Know What You Don't Know: Unanswerable Questions for SQuAD." arXiv preprint arXiv:1806.03822 (2018).

[17] Lai, Guokun, et al. "Race: Large-scale reading comprehension dataset from examinations." arXiv preprint arXiv:1704.04683 (2017).