Deep Learning Approaches on Multiple Choice Reading Comprehensions

Samuel Oshay, Leonardo Murri, Peyman Norouzi, Dewang Sultania
University of Pennsylvania
April 29, 2019

Recent breakthroughs in NLP transformers such as BERT and GPT-2 have showcased remarkable results on a variety of NLP tasks such as classification, question answering and text generation. In this paper we will present, compare and contrast BERT-based models against deep and classical machine learning models. We will evaluate these models on the very challenging RACE data set, which consists of nearly 100,000 reading comprehension questions generated by teaching professionals for middle school and high school students in China. The data set was designed to evaluate an individual's comprehension and inference abilities beyond trivial information extraction.

I. INTRODUCTION

A. RACE Dataset

All models presented below were evaluated on the RACE data set [1]. Released in December 2017 by researchers at Carnegie-Mellon University, RACE has become an industry-standard NLP data set used as a drop-in for reading comprehension tasks. The data set consists of approximately 97,000 four-option multiple choice questions accompanied by approximately 28,000 reading passages. Given a passage, a question, and four multiple choice options, the task is then to identify the best answer for the question with respect to the passage. As enumerated by the data set authors, RACE hosts a few question varieties:

1. *Word Matching.* These questions, which comprise about 16% of the data set, asks the reader to find an exact word-match in the article. According to the original authors, these questions are "self-evident."

2. *Paraphrasing.* These questions, which comprise about 19% of the data set, have correct answers that paraphrase a single sentence in the passage.

3. *Single-sentence reasoning.* Unlike the question types above, these questions require inference, a historically difficult task for machine learning models. Comprising 33.4% (the plurality) of the data set, these questions can be answered by making a logical leap using information contained in exactly one sentence of the passage.

4. *Multi-sentence reasoning.* Comprising 25.8% of the data set, these questions can be answered by making a logical leap using information contained in *multiple* sentences of the passage. Thus, to answer these questions, a model must understand the relationship between sentences, as opposed to identifying the correct sentence to draw information from.

5. *Ambiguous.* For a small fraction of the data set, about 5.8%, the question cannot be concretely answered using the passage, as the question is ambiguous or insufficient information exists for the question.

The current state-of-the-art accuracy on RACE is 74.1%, achieved by Shanghai Jiao Tong University researchers through the novel Dual Co-Matching Network Model, which we implement and examine below. Comparatively, top-line human level performance has been estimated at 94.5% accuracy, while actual human performance (via Amazon Mechanical Turk) has been measured at 73.1%.

B. Sample Passage and Question

Below is a sample passage and question from RACE:

> ***Passage:*** *Thomas Edison lit up the world with his invention of the electric light. Without him, the world might still be in the dark. However, the electric light was not his only invention. He also invented the motion picture camera and 1200 other things. About every two weeks he created something new.*
>
> *Thomas Edison was born in 1847. He attended school for only three months. His mother taught him at home, but Thomas was mostly self-educated. He started experimenting at a young age.*
>
> *When he was 12 years old, he got his first job. He became a newsboy on a train. He did experiments on the train in his spare time. Unluckily, his first work experience did not end well. They him when he accidentally set fire to the floor of the train. Then Edison worked for five years as a telegraph operator, but he continued to spend much of his time in experimenting his first patent in 1868 for a vote recorder run by electricity.*
>
> *Thomas Edison was totally deaf in one ear and hard of hearing in the other, but he thought of his deafness as a blessing in many ways. It kept conversations short, so that he could have more time for work. He always worked 16 out of every 24hours. Sometimes*

his wife had to remind him to sleep and eat. Thomas Edison died at the age of 84. He left a great many inventions that greatly improved the quality of life all over the world.

Question *What does the passage mainly talk about?*

1. *The function of the electric light*
2. *Edison and his experiments*
3. *The importance of inventions*
4. ***The whole life of Edison.***

As discussed above, this passage likely qualifies as a multi-sentence reasoning question, as one must take in information about the whole passage in order to answer it. No single sentence in the passage notes that, for example, "This passage is about the whole life of Thomas Edison." Instead, the reader must intuit that fact by reading. It is our (untested) belief that this category of questions is the most difficult, with the exception of the ambiguous/insufficient information class of questions.

C. Data Hypotheses and Models

We developed the following data hypotheses for RACE:

1. Synonym Understanding

A good model should be able to pick the synonym option given a passage with an explicit answer. For example consider the passage "So his teachers didn't like him, and nor did his classmates play with him. Peter often slept in class because his heart was not in school. He almost gave himself up. One day.", and the question "Peter always failed in exams because he?", the correct answer based on the passage is gave himself up, now if we change gave himself up to "stopped trying" in the options, a good model should be able to capture that.

2. Subject-Action Pairing

A good model should be able to identify subject action relationships. For example consider the following passage, "Alice likes sunny weather just like today. She wants to know what the weather will be like tomorrow. She's going to have a picnic.This is what the reporter is saying." and the corresponding question "What is she going to do ?", based on the passage the correct answer is She's going to have a picnic. Now if we change "She's going to have a picnic" to "She's going to wash clothes" in the passage, our model should change it's prediction to She's going to wash clothes.

3. Subject-Action-Time Pairing

Building off the importance of the subject-action relationship, time is a key variable in most RACE passages, which oftentimes describe an environment in change. Thus, a good model should understand the concept of time and how it relates to the subjects and actions it identifies. For example, if the passage reads, "Yesterday, the Mets lost to the Dodgers, but they will have a chance to prove themselves against the Rockies next week," the model should be able to identify the Dodgers if asked "Which team did the Mets already play?"

4. Proper Noun Identification

A good model should be able to perform correct identification of Proper Nouns. For example for the passage "In January 1880, two of Tesla's uncles put together enough money to help him leave Gospi for Prague where he was to study. Unfortunately, he arrived too late to enroll at Charles-Ferdinand University..." and the question "What city did Tesla move to in 1880?", if we change Prague to Czech, the model should change it's answer to Czech.

5. Concept Similarity

Much of historical projects in reading comprehension have focused on leveraging sentence similarities to identify probable answers. Importantly, we can decompose sentence similarity into syntax similarity and concept similarity. For our purposes, any good model should be able to capture concept similarity over syntax similarity. For example, consider the following sentence: "The Civil War was the deadliest war in American history for Americans. However, more casualties resulted from World War II across the globe." Consider if the questions asks: "What was the deadliest war in history?" A model focused on sentence similarity would elect the Civil War, as that war falls in a similar sentence syntactically. However, a model that captures concept similarity would elect World War II, as it notes that more casualties (deaths) occurred across the globe.

6. Question Answerability

As noted above, a portion of the data set is unanswerable. Thus, a model should be able to recognize whether or not the required information for answering a question is provided in the passage. If insufficient information is given, the model should be able to choose an answer corresponding to not having enough information. If a model fails to do this, we can assume it is either 1) guessing or 2) drawing from outside knowledge. The first option is undesirable, as it takes no understanding to guess whatsoever. The second option is further undesirable, as only

information given in the passage (and no prior knowledge) is to be used to answer the questions.

D. Data Pre-Processing

The first step in learning a natural language data set is to embed the input into a numerical representation. For our advanced models, we use BERT, which we discuss in depth further below. For our non-BERT models, we used the GloVe pre-trained model, developed by Stanford's NLP Research Group [2]. Short for Global Vectors, GloVe is a widely-adopted embedding model that uses word usage correlations to project language onto a meaningful semantic space. GloVe embeddings have a series of desirable properties. For example, synonyms are closely projected, and longer sequences are embedded in a maximally-distinctive way in order to capture nuance and detail.

Another consideration of our pre-processing pipeline was padding. All observations in our data set have variable length. Thus, in order to batch our data set, we must pad and truncate our data in order to fix the width for some models. During our initial data exploration, we saw that the vast majority of observation length was held in the passage, which averaged about 322 words. Questions, on the other hand, was short, at on average about ten words; options were even shorter at approximately five words each. In general, it was our philosophy to truncate articles in deference to time and computing limitations. However, as every word in the question or candidate answer is of immense value, it was our philosophy to pad at all times and never truncate a question or option. Of course, the chosen lengths for each model functioned as a hyper-parameter that was tuned individually. Thus, we will detail our choices for passage, question, and option lengths below.

II. RELATED WORK

A wide variety of architectures have been used for reading comprehension question in the past. In this paper we will compare common architectures (logistic regression and CNN) and compare them to BERT.

A. GloVe

GLoVE is an unsupervised learning algorithm for obtaining vector representations for words. This is achieved by mapping words into a meaningful space where the distance between words is related to semantic similarity. It was develop by Stanford as an alternative to Word2Vec and quickly became very popular. The model, being implemented as a logbilinear regression model for unsupervised learning of word representations combines the features of two model families, namely the global matrix factorization and local context window methods.

B. BERT

BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms, an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERTs goal is to generate a language model, only the encoder mechanism is necessary. The BERT paper[3] presents two model sizes for BERT:

1. BERT Base: Comparable in size to the OpenAI Transformer in order to compare performance

2. BERT LARGE A ridiculously huge model which achieved the state of the art results reported in the paper.

Both BERT model sizes have a large number of encoder layers (which the paper calls Transformer Blocks) twelve for the Base version, and twenty four for the Large version. These also have larger feedforward-networks (768 and 1024 hidden units respectively), and more attention heads (12 and 16 respectively) than the default configuration in the reference implementation of the Transformer in the initial paper (6 encoder layers, 512 hidden units, and 8 attention heads).

C. Convolutional Neural Networks

Convolutional neural networks (CNN) utilize layers with convolving filters that are applied to local features.Originally invented for computer vision, CNN models have subsequently been shown to be effective for NLP and have achieved excellent results in semantic parsing, search query retrieval, sentence modeling and other traditional NLP tasks.

III. MODELS

A. Logistic Regression

For a non-deep baseline, we passed our GloVe embeddings into a simple logistic regression model. While the industry-standard GloVe embedding for a single word is 100-dimensional, we opted to expand this embedding to 300 dimensions, as passages in RACE are 1) longer than most texts in the NLP ecosystem and 2) are full of rich detail and contrasting contexts, that we hoped to capture in order to answer the multi-sentence reasoning questions. After obtaining our GloVe embeddings for each word, we use a sequence-wise average pooling to obtain

a single 300-length vector representing the article, a single 300-length vector representing the question, and four 300-length vectors representing the four multiple-choice options. In this instance, average pooling was a way to avoid having to truncate or pad our inputs, as all variable lengths inputs were reduced to 300-dimension in the average pooling process. For each option, we then concatenated the option to the passage and the question, passing this 900-length vector through a logistic regression model to obtain a single logit. We repeated this process four times: once for each option. We then concatenated our four logits and applied log softmax to output the probabilities that each of the four outputs was correct. Training for this model was done for ten epochs, using the Adam optimizer with a learning rate of 0.0002 and a batch size of 32.

(We also tested using concatenation of article, question, and option to obtain a single 300-length vector on which to do logistic regression. However, this model was unable to improve upon randomly guessing.)

B. Feed-Forward Net

For our first deep baseline, we implemented a simple feed-forward net. Unlike above, feed-forward nets require fixed-length input. We first truncated and padded our passes to 400 words, our questions to 30 words, and our options to 16 words each. We then passed the flattened word-length-by-300 GloVe embeddings through separate linear models for the passage, the question, and each of the four options. Reduced "hidden states" for the article, the question, and each of the four options were then concatenated and passed through a final linear model to produce four probabilities as outputs. The details of our model are viewable below:

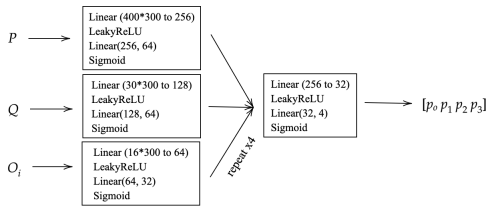

FIG. 1. Feed-Forward Net Using Separate Modeling

C. Gated Recurrent Unit (GRU)

In order to test the power of recurrence on this data set, we implemented a simple gated recurrent unit. As we did with the feed-foward net, we used separate GRU modeling for the passage, the question, and the option in each case. Further, we elected to use the bi-directional power of GRUs, as most passages have to do with change in time and place. Reading a passage forward and back, we believe, contextualizes time changes. After passing our GloVe embeddings through three separate GRUs, we flattensx these hidden states and passed them through a final linear model. The details of our model are below:

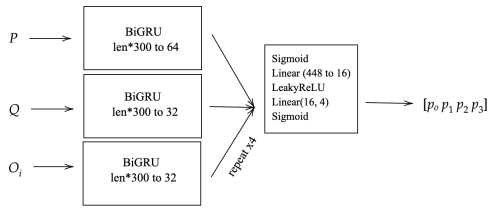

FIG. 2. GRU Using Separate Modeling

D. Convolutional Neural Networks (CNNs)

As a final deep baseline, we developed a convolutional neural network on top of the GloVe embeddings mentioned above. In constrast to the logistic regression model, CNNs require a fixed-length input. In deference to this requirement, we truncated and padded our passages, question, and answers to 500 words. We then passed the passage, the quesiton, and each candidate answer through separate CNN models. After extensive experimentation, we concluded that two layers per model were most effective. To keep the size of the model minimal, we used a standard kernel size of three for each layer. The outputs from each CNN were then flatten and passed through a linear model. The dimensions and details of our model are viewable in Figure 3.

E. Dual Co-Matching Networks

The current state-of-the-art model on RACE is the Dual Co-Matching Network, first published by researchers at Shanghai Jiao Tong University in January of 2019. The DCMN passes the passage, the question, and each candidate answer through BERT separately, receiving hidden states from each word in the input sequence. From these hidden states, the DCMN uses a bi-directional attention mechanism to match the question to the passage and each candidate answer to the passage. These hidden states are then reduced by a sequence-wise max pool and fed through a linear layer to obtain the probability of correctness for each class. The general model is as in the figure below.

The details of the matching unit are core to the value of this model and are best explained by the original paper [5]. For our extra-credit novel implementation, we implemented the general architecture of the model using the BERT base model. As we will discuss below, the dev and test performance of this model was limited due to various

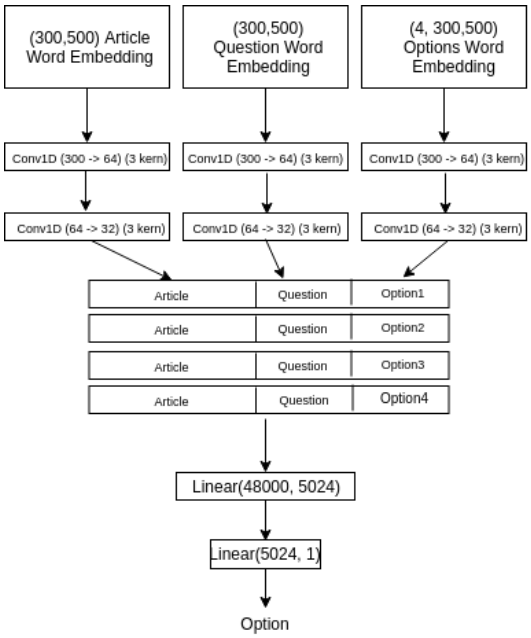

FIG. 3. CNN on Separate Inputs

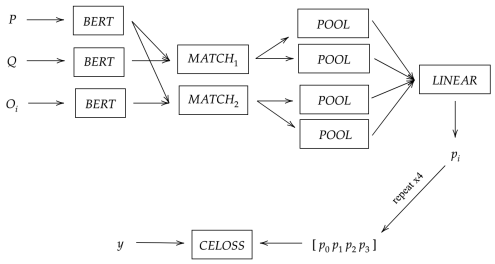

FIG. 4. The Dual Co-Matching Network Architecture

limitations, although the novel model was successful in beating all non-deep and deep baselines by considerable margins.

F. BERT

BERT originally expects a single text sentence or a pair of text sentences as it's input. For RACE, we constructed an input for each option. In that pair of text sentences BERT expects we pass the comprehension as the first input and the question concatenated with the option as the second one. We then apply a linear and a softmax layer on the final hidden state for the class token for the four input sequences of each example. By doing this we try to maximize the log probability of the correct example. We trained BERT without freezing the model weigths for two epochs, which was sufficient to obtain great results as detailed below. To build on top of the BERT sentence embedding we tried using different em-

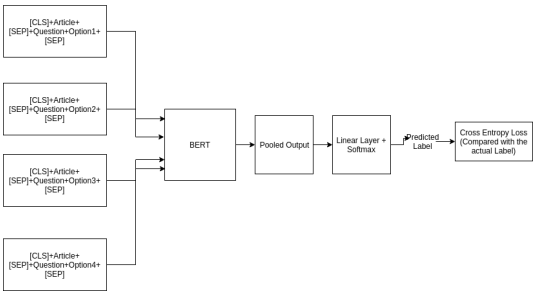

FIG. 5. BERT Training

bedding elements to make our predictions. The standard way to use BERT for this kind of classification is to train a regression on top of the pooled layer of the input. To try too boost accuracy we tried to use the embeddings of BERT large (without re-trained the model) and then ran a feed forward network on top of those embeddings. Moreover, instead of using just the pooled layer we concatenated the word embeddings from BERT last hidden state layer from the question and answer tokens.

IV. RESULTS AND DISCUSSION

Below we report the dev set and test set accuracies for each of our six implemented models. We will first discuss the performance of each of these models. To better understand the performance of each model class (non-deep baseline, deep baseline, and advanced deep), we will then take the best model from each class and examine the performance of the model with respect to our data hypotheses. These, as we discuss later, will be important in how we will approach further improvements of this work in the Future Work section. By being able to understand each model failure point, we can then adjust our architecture inductive biases to improve the quality of our predictions.

Below is the performance summary of our six models on the RACE data set:

Model List			
Model	Final Train Accuracy	Final Dev Accuracy	Final Test Accuracy
Bert Base	59.80%	57.35%	56.70%
DCMN	59.10%	43.71%	43.08%
CNN	39.33%	35.52%	34.09%
GRU	36.94%	35.18%	34.16%
Feed Forward Network	41.24%	34.24%	34.79%
Logistic Regression	32.41%	29.10%	28.22%

TABLE I. Model Performance on RACE

As the table demonstrates, every deep model performed better than our non-deep baseline, while BERT-

based models performed far better than all our baselines. Notably, our novel implementation of the advanced Dual Co-Matching Network merited significantly better performance than all our baselines. In our non-deep baseline class, the best model was only non-deep baseline, logistic regression. For our deep baseline class, all models were of equivalent performance. Even so, we select our CNN for further inquiry, as convolution is conducive to an interesting discussion about our data hypotheses. For our advanced deep models, our BERT model was most successful by a wide margin. Table II below is a summary of how each model performed against each data hypothesis.

Model List			
Data Hypothesis	Bert Base	CNN	Logistic Regression
Synonym Understanding	✓	✓	✗
Subject-Action Pairing	✓	✗	✗
Subject-Action-Time Pairing	✓	✗	✗
Proper Noun Identification	✓	✓	✗
Sentence Similarity	✗	✓	✗
Question Answerability	✗	✗	✗

TABLE II. Performance of various models on the introduced data hypothesis

Let us now explore the results of our models in greater detail.

A. Logistic Regression

As our baseline non-deep learning model we implemented logistic regression on top of average GloVe word embeddings. As expected, logistic regression did not pass any of our data hypotheses. The failure point of this linear regression model can be most likely be attributed to the average word embedding input. Each word in the article, question and options was converted to a 300 dimensional vector. First, this type of embedding does not account for any type of relationship based on the specific passage context. Next, for each component we found the euclidean mean of all the word vectors. GloVe's linear substructure representation in theory does allow for knowledge abstraction representation through geometrical operations, however given that our passages were on average longer than 500 words it is fair to assume that the great majority of information present in each word would be lost when the average is computed. The dev test accuracy achieved with logistic regression was 29.10% and the test accuracy was 28.22%. The learning curves are shown below. We can say that this type of model does a really bad job at any kind of reading comprehension, however this model's accuracy also proves that word embeddings are able to capture some information since it comfortably beats a random guessing accuracy. Moreover, this model provides evidence that our inductive bias of meaning clustering holds. When designing our models we had a prior that in an embedding space the passage and option cluster would be more correlated given the correct option. Given how logistic regression tries to separate data across teh 300 dimensional GloVe embedding and given that we achieved passing results it is safe to say that this prior can be built upon with more complex architectures.

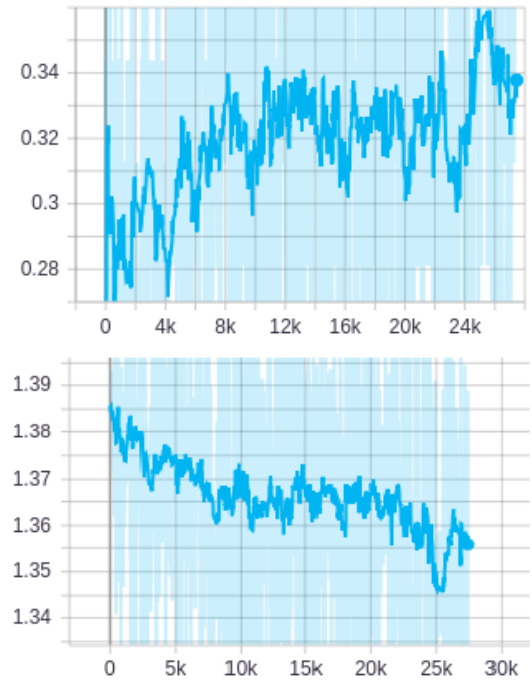

FIG. 6. Logistic Regression Learning Curves (Top: Accuracy, Bottom: Loss)

B. Feed Forward Network

The feed-forward network performed relatively well, falling in line with other deep models. The major failure point of this model is conceivably the single-sentence and multi-sentence reasoning questions, as the correct sentences to reason from are not likely to be in the same positions with passages. Thus, the correct weights that would up-weight an important sentence in one passage are unlikely to carry over to the next passage. The training accuracy and loss curves for this model appear in Figure 6 below.

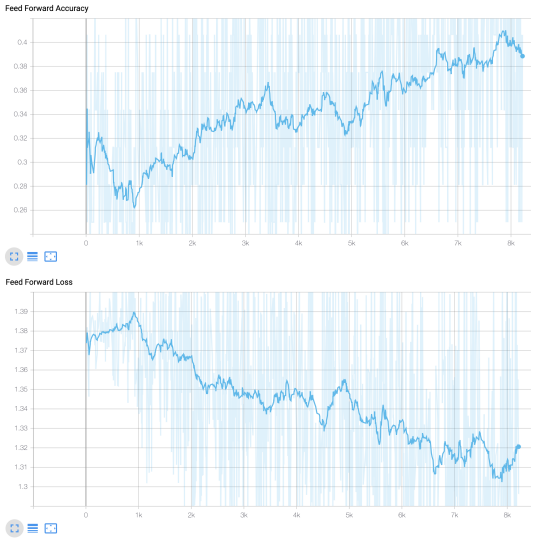

FIG. 7. Feed Forward Learning Curves
(Top: Accuracy, Bottom: Loss)

C. GRU

The GRU also performed well, in-line with the deep baseline models. The failure point for this model is likely in the GRU used to model the passage. Unlike for the question and the option, which has only a few words, passages in our model were variable length, with lengths up to 1000 words. Thus, one might surmise that much of the context of the original passage is lost through recurrence over time. The training accruracy and loss curves of this model appear in Figure 7 below.

FIG. 8. GRU Learning Curves
(Top: Accuracy, Bottom: Loss)

D. CNN

Building on top of the GloVe word embeddings we decided to implement a Convolution Neural Network. As we learned with our logistic regression model, averaging each word loses too much information. By running a n-gram convolution across all the word embeddings we hoped to capture local information in the passages to be then matched with the features extracted from the question + options CNN models. This local invariant inductive bias was meant to provide a strong flow of information from the beginning to the end of our model. In some of the answers a small piece of a sentence contained the answer, by having this locality inductive bias at play with CNN we expected to achieve decent results with respect of most of our data hypothesis. In fact just as we expected, the CNN was able to correctly identify correct answers when the passage explicitly mentioned a synonym of the correct option. Given the high quality of geometrical meaning in the GloVe word embeddings this result was expected. Also, not surprisingly the CNN model was able to capture Proper Noun identification for the same reason. More surprisingly, the CNN passed the sentence similarity data hypothesis. Our initial assumption is that the local feature extraction performed by the CNN would lose details as the number of channels decreased. Thus, we thought that sentences in the passage that were similar to the question wording would have been thought to be the correct answer, no matter what the context was. However, the CNN was actually able to understand that a sentence less similar to the question in terms of Levenshtein distance was actually the correct option. This could be explained by how the CNN extract features. Our original word embeddings were 300 dimensional and after two layer of convolution a 50 dimensional features were extracted. This type of compression most likely was able to suppress syntactical information in favor of meaning abstractions showing a great ability to abstract non-explicity text meaning.

On the other hand, the CNN failed at the rest of the data hypothesis. This was expected given the locality inductive bias underlying convolution. Similarly, extracting local feature as expected did now allow our models to subject action relationships and sentence chronology.

On the bright side, we were able to significantly improve upon the logistic regression results with 35.52% dev accuracy and 34.09% final test accuracy. These results might not seem very promising, however looking at the level of difficulty of the type of questions this is actually shows that our models were able to truly abstract some meaning from the passages.

E. Dual Co-Matching Network

The DCMN was among our most interesting models. The researchers who published the framework for this model claim state-of-the-art accuracy of 74.1%. However,

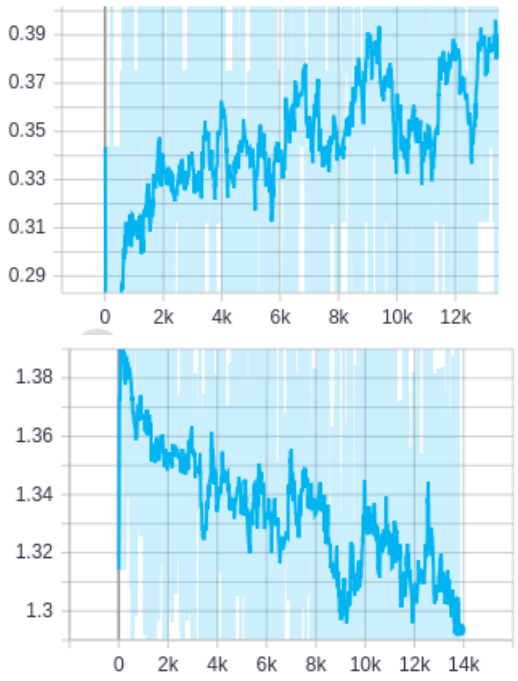

FIG. 9. Convolutional Neural Network Learning Curves (Top: Accuracy, Bottom: Loss)

our implementation fell short of their paper, achieved only 43.08% test accuracy. We hypothesize the following reasons for this odd result:

1. *Memory Issues.* The researchers in question had access to far more time and compute power than we had available on Colab. We faced a trade-off between including a large part of our passage or a larger batch size. When testing a small batch size of four with maximal passage length, we obtained irregular gradients. With smaller passage lengths, our model achieved lower levels of training accuracy. After much tinkering, we were able to balance this trade-off, while staying just under the GPU memory limitations of Colab. The researchers, with access to multiple GPUs, conceivably did not face these limitations.

2. *Compute Time.* The researches in question did not publish how many epochs they trained for. Other models using BERT trained for two to three epochs. Training for three epochs timed in at just under the Colab limit, so further traing was "off-limits." We see in our accuracy and loss curves that performance on both metrics improved greatly at the start of the third epoch. If we had more time, we hypothesize better performance.

3. *Tokenization.* BERT is compatible with a variety of tokenization schemes. We used a fairly vanilla tokenization scheme, but it unclear what the original researchers used and whether or not that this decision would affect the model.

4. *Segmentation and Masking.* In addition to our tokenized ID inputs for BERT, the model requires segment IDs and masks. For our segments IDs, we used all zeros, as each text was passed in separately. For our masks, we passed in all ones, as this is what worked for our BERT-base model below. The researchers make no mention of their schemes for this. Perhaps they used some other innovative masking scheme that we do not know of.

5. *Overall Implementation Ambiguity.* The researches in question published no code, no instructions for tokenization and indexation, no details on segmentation or masking, no hyper-parameters, and skinny details on the core implementation of layers. Thus, it is possible that the model we have implemented shares many different details.

Despite these challenges, we see much to enjoy in this implementation. A curious point about this performance is the disparity between training accuracy and dev and test accuracy. For nearly all models, accuracy on all three sets was generally equivalent. For the DCMN, the training accuracy was far higher. Due to the noise of the dataset, we find it unlikely that we overfit. Moreover, testing on the 2-epoch-trained DCMN and the 1-epoch-trained DCMN showed a similar phenomenon. The training accuracy and loss curves are below in Figure 9.

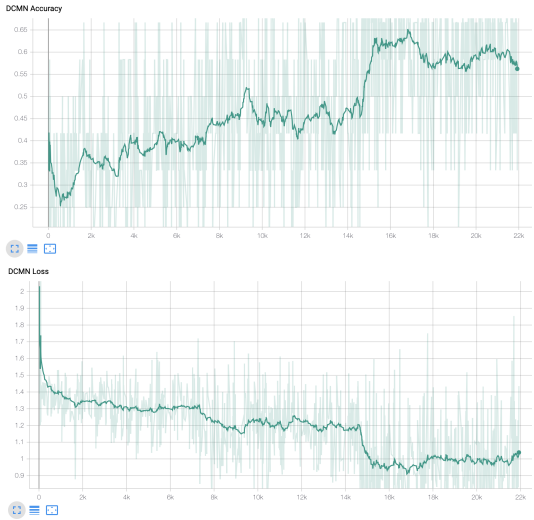

FIG. 10. DCMN Learning Curves (Top: Accuracy, Bottom: Loss)

F. BERT

Given that CNN was failing in capturing subject action relationships and chronology of actions we concluded that we needed a model that has an attention component

in it which can use Bi-directionality to capture complex relationships. We realized we needed a model that capture a prior of context attention, that is our prior is that some words in different sentences refer to the same underlying subject. We were able to level this prior by using BERT which is a Bidirectional Encoder Representation from Transformers which allows the model to learn the context of a word based on all it's surroundings. As expected BERT was able to capture these complex relationships because it was not paying attention only to the words preceding the context but also to the words succeeding it. It improved the performance of our existing models by around 25% bringing it up to 57% for test from the 32% we previously had. We fine tuned BERT by using a max sequence length of 200 and training the last few layers while freezing others. We could not go bigger because of GPU memory constraints. Due to this, BERT was failing miserably in long passages. To get around that we tried to use a text summarizer which uses page rank algorithm to find the most important sentences of a passage instead of simply truncating the extra words, this gave us better results for some of the longer passages but failed in some others because many questions were not necessarily from the parts of the passage that were considered important by the text summarizer. The State of the art BERT base on this dataset uses a max sequence length of 512 to train the model and gets around 65% accuracy. So we believe that if we are able to train our model with a longer sequence length it will be able to at least match the accuracy of the current tuned BERT base.

A second approach of building on top BERT was to use BERT large without a full end-to-end training of the model. For BERT small we downloaded the pre-trained weights and then let it train for 10 more hours without freezing any weights. We could not use this same process for BERT large because of how long it would have taken to train and our lack of GPU clusters. As a compromise, we were able to use the pre-trained BERT large model to embed all out article + question + option elements. Moreover, instead of using just the pooled output layers, we used the concatenation of the last 10 embedded vectors, which represented the answer. The idea here was to let the model focus on the context representation of the words in the answers. In theory, the correct answer should have "more" attention with respect of the question and the passage. With this procedure were were able to train our model for longer (200 epochs) which achieved a much better training accuracy (90%). Unfortunately, this model did not beat the end-to-end trained BERT-Base.

V. FUTURE WORK

Improving upon models mentioned in this paper requires a novel approach that takes solving each of the six mentioned data hypotheses into consideration.

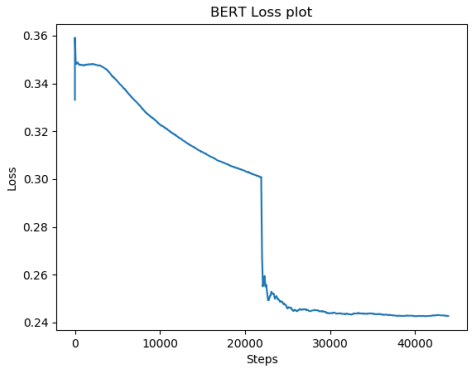

FIG. 11. Bert Base Fine Tuning Loss Learning Curve

Question Answerability was one of the big weaknesses of all of the models talked in this paper. Further work is needed to analyze possible improvements that take an evidence-based approach upon selecting an answer for each passage based question. Combining two separate models, one that predicts answerability and the other that predicts the correct choice for a question, can be one approach that can possibly address the answerability of a question. This approach requires pre-processing of the data so that in addition to the known choice, each question has a regarding 0 (not answerable) and 1 (answerable) tagged to it. If options of a question does not include "Not enough information" as an option, we can use 1 to indicate that the question is indeed answerable. But if there is "Not enough information" as one of the options, we need to manually analyze passage to see if the information given is enough for answering the question. After having proper labeling, we can train a model(lets call it *is_answerable*) on the answerability of questions.
If a question has "Not enough information" as an option, we can call *is_answerable* model to see if we can answer the question based on the passage information. If the output of *is_answerable* is 0 then we can choose "Not enough information" as our choice for the question and if the output is 1, we can go ahead and call our second model that predict the correct answer from the remanding choices.

Sentence/concept similarity was a defect of that our best performing model, Bert. On the other hand, the CNN model with GloVe embedding performed surprisingly well on this task. Further analysis is needed to see how the benefits of these two models can be combined to achieve a better overall result. One approach can use our Bert model for the embedding of the passage, question and all of the choices. Then a CNN model can be used to map the embeddings to the correct choice as explained in the methods section of this paper. This way we are taking advantages of both of the models' strengths and getting rid of each's weakness. Upon

availability of large computational power, further training of the Bert model can moreover fine-tune the weights of the model to the structure of the dataset further improving its performance. Hyperparameter tuning of the Bert model, especially a larger max sequence length input, can immensely contribute to the quality of the embedding of each sentence and thus improving the overall performance.

VI. BIBLIOGRAPHY

1. https://arxiv.org/pdf/1704.04683.pdf
2. https://nlp.stanford.edu/projects/glove/
3. https://arxiv.org/pdf/1810.04805.pdf
4. https://www.aclweb.org/anthology/D14-1181
5. https://arxiv.org/pdf/1901.09381.pdf