A PERFORMANCE AND EFFICIENCY ANALYSIS OF BIDIRECTIONAL TRANSFORMER DISTILLATION ON QUESTION ANSWERING

MUKUND VENKATESWARAN, EDWARD COHEN, GRIFFIN FITZSIMMONS, KUNAL NAIK, EZAAN MANGALJI, ANNA OROSZ

ABSTRACT. In this report, we take a closer look into bidirectional transformer models and how processes like distillation and more thorough pretraining affect their performance in question answering tasks. We used the SQuAD 2.0 dataset to train models to answer questions based on a given input context and question. Our focus was on Bidirectional Encoder Representations from Transformers (BERT) which can perform advanced language processing. We analyze the efficiency and performance of BERT, distilBERT, RoBERTa, and distilRoBERTa on the SQuAD 2.0 task, and provide error analyses for each model. We find that robust pretraining approaches implemented in RoBERTa and distilRoBERTa improve performance on the task greatly, and that we see a $\bar{4}5\%$ reduction in training and inference time when using the distil models. Our best performing model, RoBERTa, was able to yield a final F-1 score of **77.309** and Exact Match score of **74.160**.

1. INTRODUCTION

Transformer models have become state-of-the-art in a large variety of natural language tasks including question answering. In this report, we build various sized Transformer-based neural language models for the context-question answering task and provide a thorough efficiency and performance analysis to draw comparisons between the models on this particular task.

Our goal is to build models that are able to take as input a paragraph of text along with a question answered by the context paragraph and produce answers using the given text. This task, context-based question answering, is a very rudimentary constituent of the larger ideas of "machine comprehension," or "natural language understanding" (NLU). These terms are very broad and represent very difficult open problems, but notable progress has been made in this component (see Section 2: Literature Review). To understand what is meant by context-based question answering, consider the following example context paragraph along with a question relating to it:

```
CONTEXT : "The conquest of Cyprus by the Anglo-Norman forces of the Third
Crusade opened a new chapter in the history of the island, which would be
under Western European domination for the following 380 years. Although
not part of a planned operation, the conquest had much more permanent
results than initially expected."

QUESTION : "How long did Western Europe control Cyprus?"
```

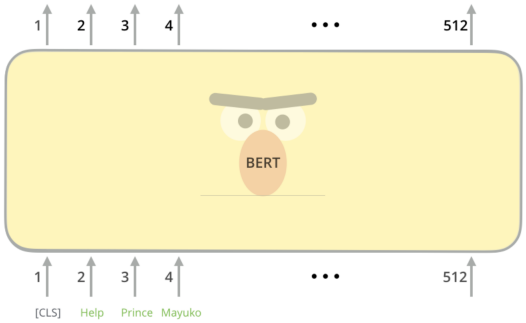
Based on the context paragraph in this case, we would expect the answer "380 years" to the given question.

In this paper, we attempt to train language models to extract the correct answers from the given context and analyze their efficiencies in doing so. Additionally, we train our models to recognize when a given answer is unanswerable based solely on the context. A significant portion (50,000 out of the 150,000 in the training set) of the question-answer pairs that constitute our data are unanswerable. Given the above context, such a question might be "Who dominated Western Europe for 380 years?"

In our solution, we utilize attention-based methods, namely bidirectional Transformer models, which encode the context paragraph in both directions and attend over it when reading in the input query. As data, we make use of the SQuAD question answering dataset from Stanford consisting of context-question pairs from Wikipedia articles in order to train our models to accomplish the task of context-based question answering. Our analysis consists of comparing the efficiency of different variations of the bidirectional Transformer model.

BiLSTM models, which were frequently used as state-of-the-art models on this particular task a few years ago, have been recently outperformed by Transformers. In using either type of neural language model for the task, attention is

quite an important factor as it allows the model to focus on different parts of the context based on the question fed to it. Below, we provide an illustration of the BERT model by Jay Allamar, which produces token-level contextual embeddings based on the context-question pairs that are input to it [1].



We chose this topic for our final project because we were excited about the possibility of tackling an open conference problem and understanding one of the challenges on the cutting edge of NLP research. Moreover, we think that it will be interesting to analyze the models from a practicality standpoint. Do we always need the full power of a large Transformer for this task, or might we be able to compromise for efficiency gains elsewhere? How does distillation affect perfomance on the task? Can we better pretrain smaller models in lieu of having larger alternatives? We especially like this topic because the problem statement is simple to understand and explain to someone with no knowledge of the field. Attempting an effective solution, however, requires the implementation of high-level neural models which represent the culmination of the NLP knowledge learned in this course.

2. Literature Review

We are adapting the shared task that uses the Stanford Question and Answering Database 2.0 to answer reading comprehension questions and abstain when a question is unanswerable. The shared task homepage can be found at https://rajpurkar.github.io/SQuAD-explorer/ and the overview paper is cited [7].

2.1. **Question Answering on SQuAD 2.0: BERT is all you need [9].** This report focuses on fine-tuning the BERT model to the SQuAD 2.0 task and experiments with changing the output layer of the BERT model and weighing the loss function. In particular, the authors find improvements to the question answering task with an added hidden layer before the shallow output layer of the model and weighing the loss of start tokens of the answers higher than the end index. We extend their methods to the distilBERT, RoBERTa, and distilRoBERTa models in order to compare their relative performances and efficiencies to eachother and to BERT base on the SQuAD task in terms of basic evaluation metrics as well as training and inference times.

2.2. **Teaching Machines to Read and Comprehend [4].** This paper introduces several attention-based BiLSTM based methods for the machine comprehension task. The authors found that the use of the attention mechanism allows for the model to integrate information from the context over long distances, which recurrent models have generally struggled with. Attention also allows interpretation of the models "attention" over the context for different queries, as we can see how this distribution is spread over the various input tokens. We note the importance of attention between the input query and context when producing answers for questions in the dataset. We use bidirectional Transformer models to produce contextual embeddings computing by attending over both the input context and query.

2.3. **Neural Module Networks for Reasoning Over Text [3].** This paper provides two extensions to Neural Module Networks (NMNs) that improve the interpretability, and accuracy of NMNs outputs. Their method is: (1) Define a set of differentiable modules (e.g. find, find-num, filter, time-diff), (2) use contextual token representations from either a bidirectional GRU, or pre-trained BERT embeddings (results using both are included in the paper), (3) create a question-parser using an encoder-decoder with soft attention to map the question into an executable program, at each time step the question attention is available to the module and (4) learning: for any given program z, compute the likelihood of the gold answer, y*, select the answer to the question by maximizing the marginal likelihood of the program and the gold answer: p(y*—z)p(z—q), where q is the question. When the authors used pre-trained BERT embeddings they achieved state of the art performance on a subset of the DROP dataset, using 21,800 open-domain questions. Additionally by using an unsupervised auxiliary objective function improved their performance.

2.4. **The Effect of Embeddings on SQuAD v2.0 [6].** In this paper, the authors explored the benefits of using a fine-tuned BERT model for word-embeddings underlying different, more complex question-and-answer architectures. They also were able to use QA architectures such as BiDAF and experimented with replacing BERT embeddings with other embeddings. They were able to find that the only embedding type that outperformed BERT was GPT-2. They concluded that pre-trained and improved contextualized word-embeddings provide improvements to the score on the SQuAD v2.0 metric. We use contextualized embeddings output from the various Transformer models (BERT, RoBERTa, etc.) for the SQuAD 2.0 task.

3. Experimental Design

3.1. **Data.** Our data comes from the Stanford Question Answering Dataset 2.0 (SQuAD 2.0). Since this is a shared task we are already provided with a public test and dev dataset and there is a hidden test dataset which we can test out model against through an upload portal. However for our model we use the provided dev set as our testing set. The train data itself is composed of 130,319 question and answers that span across 442 articles.

We will be using the provided dev dataset of 11,873 questions and answers as our testing dataset to use for evaluation of our models. This set spans 35 different articles.

In our error analysis for each of the models, we further break down questions into the types of questions, and analyze how the length of the question / answer affects the model performance. Statistics for these measures on the testing set are provided below.

Type	Percentage
What	63.1%
How	11.3%
Who	9.7%
When	8.0%
Which	4.7%
Where	4.3%
Why	1.6%

FIGURE 1. Percentage of questions by question type in our test set. Note that the percentages add to more than 100 because some questions contained two or more keywords.

Type	Average length (characters)
Question	59.5
Gold answer	9.9

FIGURE 2. Average character length of test set questions and gold answers.

Data is structured as a .json object with article title, paragraphs corresponding to those articles, questions corresponding to those paragraphs, and answers corresponding to those questions. The JSON data is formatted as shown below:

```
title: ARTICLE_TITLE,
paragraphs: [{
  qas: [{
    question: QUESTION_TEXT,
    id: HASH_ID,
    answers: [POSSIBLE_ANSWERS],
    is_impossible: BOOLEAN
  }],
  context: PARAGRAPH_CONTENTS
}]
```

3.2. **Evaluation metric.** The evaluation metrics that we will be primarily concerned with are the F1 score and the Exact Match (EM). The F1 scoring that we will use is calculated normally:

$$\text{recall} = \frac{TP}{TP + FN} \qquad \text{precision} = \frac{TP}{TP + FP}$$

$$\text{F-1} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

In these equations, TP is true positives, TN is true negatives, FP is false positives and FN is false negatives. EM will be a binary output of whether the system's prediction matches ground truth answer exactly.

The evaluation statistics are broken into two sets of scores and then combined using a weighted average: one for questions that are answerable with a certain probability (HasAns below), and questions that are not answerable with a certain probability (NoAns below). For each of these sets two scores are given: one for the percent of questions that were answered exactly correct (after some minor normalization like casing and removing whitespace: exact below), and another that is an F1 score based on common tokens between answers (F1 below).

These two sets are combined with a weighted average based on the number of answerable (HasAns total below) and unanswerable questions (NoAns total below) as a fraction of the total questions (total below). This provides an exact score and F1 score for the entire dataset ("exact", "F1" below).

We also plan on measuring the training and inference time for each of our attempted models. Training time directly comes from the tensorboard plots produced over the training period. Inference time is a measure of how long the model took to produce predictions on the testing set.

3.3. **Simple Baseline.** Our simple baseline simply outputs the most common one or two gram from the given context paragraph as the answer to the question regardless of the given question.

Note that our model removes stopwords in the case of one-grams but not two-grams as stopwords may be a part of a two-gram entity (whereas stopwords are very rarely answers themselves in the case of one-grams).

The evaluation metrics for this simple baseline model are listed below. Note metrics are given as percentage value.

	Exact match	F1 Score	Has Answer EM	Has Answer F1	No Answer Accuracy
Simple baseline	0.707	3.356	1.417	7.122	0.0

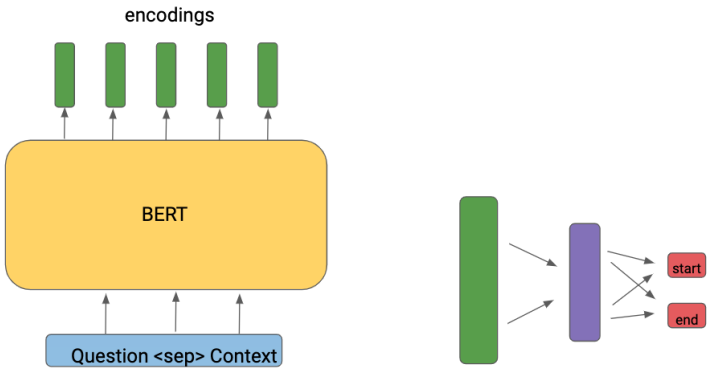
FIGURE 3. Simple baseline performance

This simple baseline performs quite poorly. For reference, this model took 28 seconds to produce predictions on the testing set. We make strong improvements to it with the neural language models below.

4. EXPERIMENTAL RESULTS

4.1. **Published Baseline.** We use a simple extension of the BERT model to the context-question answering task as shown in [9]. After inputting tokens corresponding to the question-context pair separated by a space token into BERT, it outputs 768-sized contextual embeddings for each input token. From this, we train a fully connected network with one size 256 hidden layer to train a size-two output representing logits for the probability that the token is a start or an end token. We take the softmax of these values over all the tokens in order to interpret these as probabilities, and we take the token with the highest probability of being the start token as the start token as the the token with the highest probability of being the end token as the end token. If this is nonsensical, such as outputting part of the question or outputting an end token that comes before the start token, we simply interpret this as no answer. We train the network to output the dummy start token of the sequence as the output for both start and end token in the case of a question with no answer.

We provide an illustration of the implemented system below.



(A) First, we compute encodings for each of the input tokens using the pre-trained BERT model

(B) Next, we pass these encodings through a fully connected network to produce start and end logits for each encoding

FIGURE 4. Diagram of the BERT Q/A model

In order to train the additional parameters added in the fully connected network and fine-tune the BERT model, we calculate loss using the cross-entropy function on the fully connected output based on what the actual starting and ending token were. As suggested by "BERT is All You Need" report, we weigh the loss for the start token twice as much as the end token as this is arguably the more important token for the model to guess correctly[9]. This gradient is passed through both the parameters of the fully connected network and through the BERT Transformer when performing back propagation to train the system. We train using the ADAM optimizer with a small learning rate of .00003 for two epochs. The other models used in our experimentation are trained with the same hyperparameters so that we can draw strong conclusions from our comparisons between them for the SQuAD task.

A table of the number of parameters in each of our implemented systems, along with a loss plot of the models over the training period is left below. Note that since we train the model using batches of size 6 (as to not overload the GPU RAM), so we apply necessary smoothing to interpret the loss plot. The loss plots for all attempted models converge quite similarly.

Model	Number of parameters
BERT	110m + 197120
distilBERT	66m + 197120
RoBERTa	125m + 197120
distilRoBERTa	82m + 197120

FIGURE 5. Number of trained parameters in each of the Transformer architectures. The first value denotes the number of parameters in the Transformer model and 197120 denotes the number of additional parameters we trained in the feed-forward network for fine-tuning.

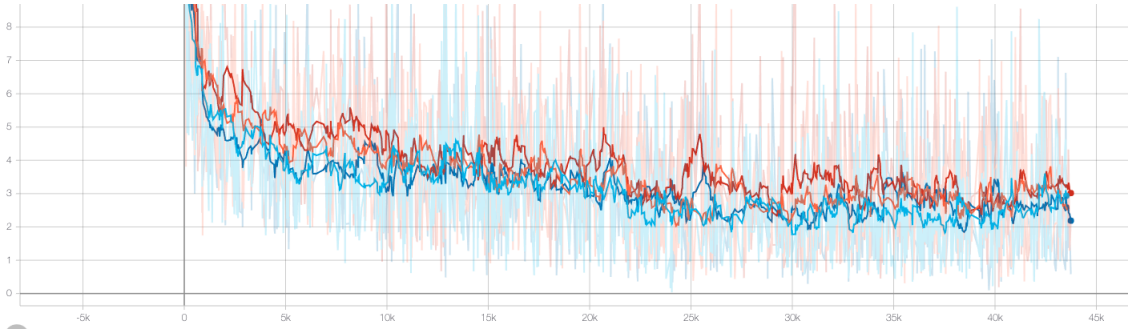


FIGURE 6. Loss plot for our 4 trained models. BERT and RoBERTa are shown in light blue and dark blue respectively. distilBERT and distilRoBERTa are shown in orange and red respectively.

4.2. **BERT base.** The first Transformer model we used in this task was the BERT base model [9]. BERT base is the original Bidirectional Encoder Representations from Tranformers model as presented by Google AI Language. We train the model as specified above for two epochs. The model took just under 14 hours to train, which was the longest out of the advanced models that we attempted. This is the direct implementation of the published baseline model. We use this particular Transformer model for the SQuAD task as the contextual embeddings produced masked language model pre-training procedure are likely a good starting point for our Q/A task. We find that this model (along with the other Transformer models) is able to converge quite quickly after only 2 epochs.

	Exact match	F-1 Score	Has Answer EM	Has Answer F1	No Answer Accuracy
BERT base	68.045	72.533	58.620	67.608	77.443

FIGURE 7. BERT base model

The performance of the published baseline is better than our implemented model likely due to the longer training time, though our results are somewhat comparable.

4.2.1. *Error analysis.* For incorrect exact matches 27% of the time was because we did not answer the question at all and 35% of the time was because we answered but there was not an answer. In 3% of our wrong answers, after removing punctuation and discounting empty string instances, the predicted answer was actually a substring of the correct answer. The converse occurred 6% of the time, even more often.

Additionally, there was some difficulty for the model when answering with approximate measurements as there seems to be an inconsistency in the expected answers in that case. Here are some examples of the aforementioned error type:

```
Question: At what age do immune responses typically begin to decline?

Gold Answer: "around 50 years of age"

Predicted Answer: "50 years of age"


Question: How long is the section that turns north?

Gold Answer: "86 km"

Predicted Answer: "nearly 86 km"
```

These types of error indicate that the predicted error was very close and in many cases would be considered correct under a different evaluation metric besides ”exact match.” Fact that the second type of error occurs more frequently than the first (both here and in all models below) indicates that our implementations might benefit from an addition

post-processing step in which certain modifiers (i.e. “nearly” above) are removed from the predictions.

Below is a table showing the question type of the questions the published baseline got wrong. Based on the question type percentages in Figure 1 the published baseline performed worse than expected on “How” questions, and “Why” questions, but performed notably better on “Who” and “When” questions and close to expected on the rest.

Type	**Percentage**
What	62.3%
How	13.4%
Who	8.8%
When	6.3%
Which	4.9%
Where	4.6%
Why	2.1%

FIGURE 8. Percentage of incorrect predictions by question type.

Type	**Average length (characters)**
Incorrect questions	60.4
Incorrect gold answers	15.0
Incorrect predicted answers	27.0
Predicted answers	13.5

FIGURE 9. Average character length of questions and gold answers that the model got incorrect. Also, average character length of the model’s over predictions and incorrect predictions.

From the table above we see that the BERT baseline generally predicted long sequences. On average the baseline predictions were 36% longer than the gold answers. Especially when it predicted incorrectly its answer was very long, almost double the length of what the answer actually was (comparing Incorrect predicted answers to Incorrect gold answers in Figure 9. There was not a significant difference in the question lengths that the baseline got wrong and the average question lengths in the test set (from Figure 2).

4.3. **distilBERT.** The distilBERT model is a distilled version of BERT trained by using BERT as a teacher model. In particular, the training process is modeled as supervised learning where the distilBERT model is trained to replicate the output of the BERT model on a given input sequence using a cross-entropy loss to minimize the difference between the output. As BERT is trained with the Masked Language Model objective, it is trained to predict masked tokens in the input sequence. When training this distilBERT model, we can simply input the same masked sequence to the BERT model and train the distributions of output tokens to be the same.

This model is able to compress BERT up to 40% while retaining 97% of language understanding capabilities [8]. We train the distilBERT model using the same methodology outlined above in order to test it’s language modeling capabilities against BERT for the Q/A task in particular. Below is a table summarizing the results.

	Exact match	F-1 Score	Has Answer EM	Has Answer F1	No Answer Accuracy
distilBERT base	61.324	65.027	52.379	59.797	70.244

FIGURE 10. distilBERT base model

We find that the distilBERT model performance is somewhat on-par with the BERT base model, but the sharp decrease in parameters comes at a cost of performance for the model.

4.3.1. *Error analysis.* For errors on exact matches 34% of the time distilBERT did not answer the question at all and 39% of the time the model answered but there was not an answer.
In 2% of our wrong answer, after removing punctuation and discounting empty string instances, the predicted answer was actually a substring of the correct answer. The converse occurred 4% of the time, even more often.

Below is a table showing the question type of the questions the published baseline got wrong. Based on the question type percentages in Figure 1 distilBERT performed worse than expected on "How" questions, and "Why" questions, especially bad on "Why" but performed notably better on "Who" and "When" questions and close to expected on the rest.

Type	Percentage
What	62.5%
How	13.0%
Who	8.9%
When	6.7%
Which	4.9%
Where	4.5%
Why	2.0%

FIGURE 11. Percentage of incorrect predictions by question type.

Type	Average length (characters)
Incorrect questions	60.5
Incorrect gold answers	14.2
Incorrect predicted answers	19.1
Predicted answers	11.6

FIGURE 12. Average character length of questions and gold answers that the model got incorrect. Also, average character length of the model's over predictions and incorrect predictions.

Here is an example of distilBERT performing poorly on a "Why" question that is unanswerable:

```
Question: Why was no damage caused by breathing pure O in space applications?

Gold answer: ""

Predicted answer: low total pressures used
```

4.4. **RoBERTa base.** RoBERTa is a Transformer model with slightly more parameters than BERT [5]. Essentially, the authors found that the pre-training approach used in the BERT paper yields an undertrained model, so they extend the masked language model pre-training procedure to more robustly train the language model. We fine-tune this model using the same methodology outline above.

	Exact match	F-1 Score	Has Answer EM	Has Answer F1	No Answer Accuracy
RoBERTa base	74.160	77.309	67.696	74.005	80.606

FIGURE 13. RoBERTa base model

We find that this model is able to outperform the published baseline after fewer epochs of training! This is evidence that the pre-training approach utilized by RoBERTa does in fact help to improve performance on the context-question answering task.

4.4.1. *Error analysis.* On incorrect exact matches 36% of the time RoBERTA did not answer and 38% of the time RoBERTA did answer but it was an unanswerable question.

In 5% of our wrong answers, after removing punctuation and discounting empty string instances, the predicted answer was actually a substring of the correct answer. The converse occurred 10% of the time, even more often.

Below is a table showing the question type of the questions the published baseline got wrong. Based on the question type percentages in Figure 1 RoBERTa performed worse than expected on "How" questions, and "Why" questions, but performed notably better on "When" questions and close to expected on the rest.

Type	Percentage
What	62.2%
How	12.9%
Who	9.0%
When	6.5%
Which	4.9%
Where	4.7%
Why	2.4%

FIGURE 14. Percentage of incorrect predictions by question type.

Type	Average length (characters)
Incorrect questions	60.6
Incorrect gold answers	14.5
Incorrect predicted answers	19.0
Predicted answers	11.1

FIGURE 15. Average character length of questions and gold answers that the model got incorrect. Also, average character length of the model's over predictions and incorrect predictions.

In Figure 15 we see again that a model performed poorly on questions with long answers and generated longer than necessary output. However, with RoBERTa the difference in the length of predicted answers and gold answers is decreasing, but on questions that RoBERTa got correct and incorrect. Also we now see a pattern that the question length is not, or is minimally correlated with whether models got the question right or wrong. This is an example of RoBERTa producing longer output than necessary:

```
Question: What were the origins of the Raoullii family?

Gold answer: descended from an Italo-Norman named Raoul

RoBERTa answer: Norman mercenary origin during the period of the Comnenian
Restoration, when Byzantine emperors were seeking out western European
warriors. The Raoullii were descended from an Italo-Norman named Raoul.
```

4.5. **distilRoBERTa.** distilRoBERTa is a compressed version of the RoBERTa Transformer much like distilBERT is a compressed version of BERT. We train this model using the same methodology outline above.

	Exact match	F-1 Score	Has Answer EM	Has Answer F1	No Answer Accuracy
distilRoBERTa base	68.508	71.489	59.1596	65.130	77.830

FIGURE 16. distilRoBERTa base model performance

Performance of this model is on-par with the BERT base model, while being comprised of far fewer parameters. We expand on these observations in the comparisons section below.

4.5.1. *Error analysis.* 41% of time we got wrong on an exact match was because we did not answer the question at all. 35% of time we got wrong on exact match was because we answered but there was not an answer.
In 4% of our wrong answer, after removing punctuation and discounting empty string instances, the predicted answer was actually a substring of the correct answer. The converse occurred 7% of the time, even more often.

Below is a table showing the question type of the questions the published baseline got wrong. Based on the question type percentages in Figure 1 distilRoBERTa performed worse than expected on "How" questions, and "Why" questions, but performed notably better on "Who" and "When" questions and close to expected on the rest.

Type	Percentage
What	62.7%
How	12.8%
Who	8.8%
When	6.8%
Which	4.8%
Where	4.4%
Why	2.3%

FIGURE 17. Percentage of incorrect predictions by question type.

Type	Average length (characters)
Incorrect questions	60.7
Incorrect gold answers	15.4
Incorrect predicted answers	17.4
Predicted answers	10.4

FIGURE 18. Average character length of questions and gold answers that the model got incorrect. Also, average character length of the model's over predictions and incorrect predictions.

Analyzing the above we see that distilRoBERTa had the smallest difference between predicted sequence length, and gold answer sequence length, however, this did not translate to the most correctness. That title goes to RoBERTa and is further discussed in the next section. All of distilRoBERTa's predicted answers were barely longer than the gold answer, even on incorrect answers. This is most likely due to RoBERTa and distilRoBERTa often getting questions wrong because they did not answer the question which will naturally weight the predicted answer length downwards. The below shows an example of when distilRoBERTa did not answer a question that had a long answer. This exemplifies both why distilRoBERTa had short predicted answers on average and a question missed because it was not answered.

```
Question: From which countries did the Norse originate?

Gold answer: Denmark, Iceland, and Norway

distilRoBERTa answer: ""
```

4.6. **Comparing Models.** We summarize results for all of our models in a table below along with their training and inference times over the testing set.

Model	Exact match	F-1 Score	Has Answer EM	Has Answer F1	No Answer Accuracy
BERT base	68.045	72.533	58.620	67.608	77.443
distilBERT base	61.324	65.027	52.379	59.797	70.244
RoBERTa base	**74.160**	**77.309**	**67.696**	**74.005**	**80.606**
distilRoBERTa base	68.508	71.489	59.1596	65.130	77.830

FIGURE 19. Scores on test set for each model.

Model	Training Time	Inference Time
BERT	13hr 48min 33sec	4min 21.50sec
distilBERT	7hr 03min 58sec	2min 11.60sec
RoBERTa	13hr 12min 33sec	4min 25.19sec
distilRoBERTa	7hr 09min 56sec	2min 20.10sec

FIGURE 20. Training time for two epochs over the SQuAD dataset along with inference time over the entire testing set. Models were trained on AWS ml.p2.xlarge instances and test set predictions were produced on Google Colab GPUs.

There does seem to be the obvious correlation between increased number of parameters in the Transformer model, and performance on this task, however it seems that more thorough pretraining of the Transformer model seems to realize more significant performance improvements. Specifically, we find that RoBERTa beats BERT by over 6% in both F1 score and Exact Match in the questions with answers. Similarly, distilRoBERTa outperforms distilBERT by over 5% in both F1 score and Exact Match in the questions with answers. This is evidence that the robust pretraining approaches in RoBERTa do aid significantly in performance on this task, and specifically in the questions that do have answers!

The RoBERTa model actually performs the best across the board for every measure performance metric, and was especially impressive in how much better it did for correct exact matches. However, this performance does come at the cost of RoBERTa taking the second longest amount of time to train and the longest amount of time to inference, almost double distilRoBERTa training time and inference, which was arguably the second best performing model. RoBERTa also requires more parameters than any other model. Considering these factors it makes sense RoBERTa would perform the best, however, when considering which model to use the time and size matter. For example, if you are simply training a model for a one time use, or have to train a model quickly you should choose a distil model which trains faster and spends much less compute. distilBERT was the worst performing model, however that is not a surprise because it is a smaller and lighter BERT. Also, the differences between RoBERTa and distilRoBERTa and BERT and distilBERT are comparable with respect to all measures of performance. The most interesting takeaway is that distilRoBERTa performed as well or almost as well as BERT in every category, making the case that more robust pre-training approaches aid in performance on the task more than increased parameterizations. We are able to yield similar performance to BERT using a smaller model with better pre-training approaches, and able to yield much better results using a similar sized model with these more robust pretrainining approaches.

We found that the distil models were able to yield $\tilde{4}5\%$ reduction in training time and inference time on this task. If these models were being made available on a large scale to consumers and we are concerned with availability, it may make sense to further fine-tune one of the distil-models as inference time becomes an important factor when making consumer software. On the other hand, if we are more directly concerned with the performance of the model, it makes more sense to thoroughly fine-tune one of the base RoBERTa models, or even go up to large versions of the model!

5. CONCLUSIONS

In this report we analyze the performance and efficiency of various bidirectional Transformer models on the SQuAD 2.0 task. Using a simple fine-tuning approach, we train two layers on top of each of the Transformer models to be able to output start token and end token predictions, which comprise the system's answer to the question.

We find the robust pretraining approaches implemented in the production of RoBERTa, and thus distilRoBERTa, to strongly improve performance specifically on questions with answers when compared to BERT and distilBERT.

Moreover, we find $\tilde{4}5\%$ reduction in training and inference times for the distil models compared to the base models. We conclude not only that distil models are still able to model the task well with decreased parameterization, but that a distilled bidirectional Transformer model with proper pretraining, namely distilRoBERTa, can reach performance standards of BERT in the context-question answering setting.

6. ACKNOWLEDGEMENTS

We would like to thank Professor Callison-Burch and Arun Kirubarajan for mentoring this project as well as HuggingFace for their Transformers library, which was used throughout the implementation described in this report. Also, thank you to the Stanford NLP group for making the SQuAD dataset so accessible.

7. APPENDICES

(1) Alammar, J. (2018, December 3). The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning). Retrieved April 29, 2020, from http://jalammar.github.io/illustrated-bert/

(2) Devlin, Jacob et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." ArXiv abs/1810.04805 (2019): n. pag.

(3) Gupta, N., Lin, K., Roth, D., Singh, S., Gardner, M. (2019). Neural Module Networks for Reasoning over Text. arXiv preprint arXiv:1912.04971.

(4) Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., Blunsom, P. (2015). Teaching machines to read and comprehend. In Advances in neural information processing systems (pp. 1693-1701).

(5) Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

(6) Perez, L. A. "The Effect of Embeddings on SQuAD v2.0." (2019). Semantic Scholar. pdfs.semanticscholar.org/a62f/5234e3ab7848667af2f0d5f5cd58fe3febc0.pdf.

(7) Rajpurkar, P., Jia, R., Liang, P. (2018). Know what you don't know: Unanswerable questions for SQuAD. arXiv preprint arXiv:1806.03822.

(8) Sanh, V., Debut, L., Chaumond, J., Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108.

(9) Schwager, S., Solitaro, John. (2019). Question and Answering on SQuAD 2.0: BERT Is All You Need. web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/default/15812785.pdf.