Natural Language Question Answering

Akhila Jetty
ajetty@umass.edu

Neeharika Karanam
nkaranam@umass.edu

Nikitha Masanagi
nmasanagi@umass.edu

Roshitha Bezawada
rbezawada@umass.edu

1 Problem Statement:

Question Answering is one of the most challenging tasks in the natural language processing domain [1]. The most important components in the question answering are the capability to understand the question and the context in which the question has been generated. It has been considered quite challenging as it is dynamic in nature [1] has led to the application of the data-driven methods of question answering. The main idea is to allow the data to be given more importance than the methods in the question answering as there are many text repositories which are available[2].

There are a lot of problems in the domains of Natural Language Processing and Artificial Intelligence which are aligned as the question answering problems [5]. For instance, the text summarization can be reframed as a question answering task where the user is asking the model for "What is the summary of the article?", which will provide the summary of the entire article. Our aim is to design a system which can be basically used to help solve the problem. In this project we are addressing the Question Answering problem using the SQuAD dataset [21], which is a huge collection of the Wikipedia articles with associated questions with the goal to identify the span of the characters which will contain the answer for the question. The project mainly focuses on the SQuAD dataset that contains of more than 100,000 Question-Answer pairs on 536 articles. In this dataset each of the question has at least one answer which is associated with the question.

We have implemented five different models and compared them, and they are as follows:
1. As our baseline we have used a basic recurrent encoder with a naïve attention mechanism.
2. Used the BiDAF (Bi-Directional Attention Flow) network also as a part of our baseline model. [20]
3. A pre-trained BERT (Bidirectional Encoder Representations from Transformers) language model. [4]
4. A pre-trained DistilBERT [19] is used which is a distilled version of BERT.
5. A pre-trained ELECTRA (Efficiently Learning an Encoder that Classifies Token Re-placements Accurately) [3], which is a BERT model that is trained with more sample efficient pre-training task.

The different models share a custom output module which is built specifically for the Question Answering tasks which will output the various character-level spans of the answer in the particular context. This is done for each and every question in the mini batch.

The recurrent-based models used like the Baseline and the BiDAF have been trained from scratch, whereas the Transformer-based models like the BERT, DistilBERT and ELECTRA have been pre-trained on some other tasks other than the Question-Answering are fine-tuned on the SQuAD dataset. In our fine-tuning approach we have used a single small learning rate for our pre-trained model (backbone) and as well as the output module.

We have observed from our experiments that we have got the best values for our recurrent-based module BiDAF. It has achieved a 68% of F1 score and 55% of EM on the validation split on the entire dataset. In the case of the Transformer-based models we have got the best values for ELECTRA which has achieved an 84% of F1 score and 71% of EM.

2 What you proposed vs. what you accomplished

Here is a list of things that we have proposed to do in the project proposal. There are a few extra things we managed to do and there are a few things which we couldn't accomplish in the given time frame. They are as follows:

1.~~Usage of the SQuAD dataset.~~
2.~~Implementation of the baseline architecture using naive self-attention mechanism followed by building BiDAF architecture.~~
3.~~Usage of BERT based transformer models as encoders.~~
4.~~Implementation of the character-level embeddings in the BiDAF model.~~
5.~~We have tried to use the SQuAD v2.0 dataset to finetune the transformer models. But we have observed that the SQuAD v2.0 dataset doesn't contain answers to all the questions unlike SQuAD v1.1. Therefore, the performance was better with SQuAD v1.1 and we stuck to that.~~
6.~~We have fine-tuned the pre-trained BERT, DistilBERT and ELECTRA models using SQuAD v1.1 dataset.~~
8. We have implemented the prompt-tuning on the BERT model for a few epochs and it was taking a lot of time therefore, we had to terminate it because of the time constraint.

3 Background/Related Work:

Question Answering (QA) has been a difficult task in natural language processing and understanding [1]. The fundamental component of QA requires the understanding the meaning of the question as well as the context of

the question, before generating the answer. Because of the dynamic nature of natural languages, the task of QA has been quite challenging. Many papers were published which classify and report the state of the art of question answering systems[15][16].

The initial question answering systems were rule-based approaches, that is, logical representations of decision trees. These systems utilized rules which are handcrafted and heuristic, relying on lexical and semantic hints on context to determine the correct answer for a given question.[3] But there was a major drawback with these systems. The heuristic rules are needed to be manually crafted and for this an in-depth knowledge of language is required. With further advancements, there are other approaches developed such as neural networks, probabilistic models and algebraic models.[1] Later, due to the increase in data in the form of text repositories and web data, statistical approaches gained momentum.

With the introduction of machine learning to the QA domain algorithms that can learn to understand linguistic features without explicitly being told to. Using statistical methods cemented the path for this approach that system uses to analyse an annotated corpus or training set and then build a knowledge base[9]. Models like Support vector machine (SVM) classifiers, Bayesian classifiers, Maximum entropy models are some techniques that have been used for question classification purpose[8].

The systems are induced a self-learning capability, where there a capable of building the knowledge base from the training data which is provided to the system, and then use the features to answer the given question[17]. This in fact brought a level of independence to the systems that were not quite there in rule-based or statistical approaches. The major advantage machine learning brings to the table is its ability to learn. This makes the system highly scalable as long as there is enough training data. Many advancements have been made in the field of deep learning, which mainly differs from machine learning because of its ability to learn the underlying features in data using the neural networks. The neural network architectures map the textual context into logical representations that are then used for answer prediction.

Many natural language processing (NLP) models have been used for this task.[10]Many new concepts were also proposed to increase the efficiency of the systems. One of them is POS tagging[11] which is a famous NLP process referring to categorizing the words in a text (corpus) in correspondence with a particular part of speech, depending on the definition of the word and its context. Another interesting concept called tf-idf concept[5] was introduced in the field of natural language processing which is used in this project as well. Tf gives us the information on how often a term or a word appears in a document, whereas the idf gives us the information about the relative rarity of a term in the collection of documents. One of the major model proposed uses POS tagging[11] and tf-idf concept [5] to match the question with with questions already present on Yahoo Answers. Although, the major flaw arose when the query might not be present on the internet.

Several Deep Neural networks models have been used to solve NLP tasks. These models have majorly used Recurrent Neural networks like LSTMS,s and GRU,s for text classification, summarization.[11]Recurrent Neural Networks have shown significant improvement in the field of natural language processing. These networks differ from the traditional neural networks because of the relationships maintained by the hidden layer with the previous values[18]. This recurrent property gives RNN the potential to model long span dependencies[19]. One of the major breakthrough is Memory networks model (Weston et al.) [12] which proposed the use of memory in the system in order to effectively answer the questions.

In our proposed work, transformers are being used to make better predictions and metrics. These are very strong and powerful neural network architectures which solely rely on the attention mechanism which is trainable and also helps in identifying the complex dependencies between the various elements for each of the given input sequence. A few of them include LUKE [13] which will use the deep contextualized Entity Representations with the help of entity-aware Self-attention, XLNet [14]which will help us integrate the ideas from the Transformer-XL, the state-of-the-art autoregressive model and then comes the most popular transformer called the BERT.

We will be using SQuAD dataset for this project. In contrast to the prior datasets, SQuAD does not provide a list of answer choices for every question. Rather, the system must select the answer from all possible spans of the passage, thus needing to cope with a fairly large number of candidates. To assess the difficulty of SQuAD dataset[21], a logistic regression model is implemented with a range of features. Significant improvements were observed and cited in the paper cited[20] suggesting that there is plenty of room for advancement in modeling and learning on the SQuAD dataset.SQuAD is much larger than all datasets except the semi-synthetic cloze-style datasets, and it is similar to TREC-QA in the open-endedness of the answers.

With the wide success of pre-trained large language models, a range of techniques has arisen to adapt these general-purpose models to downstream tasks. One of them is prompt tuning[22], a simple yet effective mechanism for learning "soft prompts" to condition frozen language models to perform specific downstream tasks.

4 Datasets:

We will be using the Stanford Question Answering Dataset (SQuAD) which is a reading comprehension dataset, consisting of various questions which are posed by the crowdworkers on a few sets of articles which

are gathered from Wikipedia. The creators of the dataset have sampled 536 from the top 1,0000 articles on Wikipedia. Amongst these sampled articles 23,215 individual paragraphs have been extracted. The dataset is split in the form of training, development and testing in the ratio of 80%, 10% and 10% respectively. In this every question will have an answer in the form of a segment of text or in the form of span from the reading passage which is corresponding to it. SQuAD dataset is quite big, and it is very challenging therefore it requires a lot of reasoning when compared to the other existing datasets consisting of reading comprehension. One of the most famous dataset is the cloze dataset in which the model is asked to predict a missing word from the passage. These datasets are a bit too similar, but SQuAD mainly focuses on the task of question answering and tests the ability of the model to read a particular passage of text and answer various questions about it. One difference between SQuAD questions and cloze-style queries is that answers to cloze queries are single words or entities, while answers in SQuAD often include non-entities and can be much longer phrases. Another difference is that SQuAD focuses on questions whose answers are entailed by the passage, whereas the answers to cloze-style queries are merely suggested by the passage.

One of the main reasons that SQuAD is so good is because the dataset is very big unlike the other datasets, the SQuAD dataset in quite challenging and the SQuAD requires a lot of intensive reasoning making it the better in the aspects of evaluating the model and understanding its capabilities. SQuAD has a few key properties which have been very well researched by the creators of the dataset:

- Various categories of answers – There are various categories in which the answers are partitioned like “date”, “person”, “location”, “other numeric”, “adjective phrase”, “verb phrase”, “clause”, “other entity” and “other”.
- Intensive reasoning required – The questions were sampled by the creators from the development set, and they were manually labeled questions into various categories of reasoning required to help answer them.
- Syntactic divergence – The creators have measured the syntactic divergence between the question and the answer to measure the difficulty of a particular question. This is a metric that will help us in evaluating the number of edits which are needed to transform a particular question into the form of an answer.

The SQuAD dataset has 87,599 train samples and 10,570 validation samples. The SQuAD dataset has the following fields:

- Id: string
- Title: string
- Context: string
- Question: string
- Answers: Dictionary

Example of the train data is as below:
{ "title": "University_of_Notre_Dame",
"paragraphs": [{ "context": "Architecturally, the school has a Catholic character. Atop the Main Building's gold dome is a golden statue of the Virgin Mary. Immediately in front of the Main Building and facing it, is a copper statue of Christ with arms upraised with the legend \"Venite Ad Me Omnes \". Next to the Main Building is the Basilica of the Sacred Heart. Immediately behind the basilica is the Grotto, a Marian place of prayer and reflection. It is a replica of the grotto at Lourdes, France where the Virgin Mary reputedly appeared to Saint Bernadette Soubirous in 1858. At the end of the main drive (and in a direct line that connects through 3 statues and the Gold Dome), is a simple, modern stone statue of Mary.",
[{ "answers":
[{ "answer_start": 515,
"text": "Saint Bernadette Soubirous" }],
"question": "To whom did the Virgin Mary allegedly appear in 1858 in Lourdes France?",
"id": "5733be284776f41900661182" }]
}

4.1 Data Preprocessing:

The given dataset is in the JSON format. The raw format of the dataset is loaded and is parsed into a tabular data structure. This will be used for subsequent pre-processing tasks. Since each context has multiple question/answer pairs and each question can have multiple answers, we allocated one row for each context/question/answer, thus replicating the same context and question multiple times, in order to consider it as a separate example. The whole dataset is split into training and validation dataset in the ratio of 80:20. The way the validation set is built guarantees that the entire set of rows referencing to the same context is kept into the same split. This is done to avoid unjustifiable boosting of results.

Tokenization is performed on the processed dataset and happens on the fly at the batch level, thus enabling us to perform dynamic padding and also helps us in avoiding pre-tokenization overhead. Different models use different tokenization pipelines.

In our model we used two micro-categories. One is recurrent-based models tokenizer where the words are split by whitespaces and punctuations. All the accents are removed and a lowercasing function is applied on all the tokens. One thing to remember is that the questions are padded but not truncated where as contexts are truncated and padded. The maximum number of tokens for truncating contexts was fixed to 300.

The other tokenizer used in our case when DistilBERT is used, is the transformer-based model tokenizer which split words using the wordPiece Algorithm[7']. All the

unicodes and foreign characters are normalized. Even in this case, all the accents are removed and lower case function is applied to all the tokens and also merging questions and contexts using special tokens such as [CLS] and [SEP]. Then the combined sentence of questions along with special tokens and contexts is truncated to maximum number of tokens(512) and padded to the right.

To perform the tokenization, we built a class called The SquadDataManager which acts as both a data collator (i.e. it brings together multiple examples in the dataset with the help of PyTorch's DataLoaders) and a tokenizer. This class also acts as a pre-processor, by removing the rows that contain wrong answers(answers that do not start and end at word boundaries) and also removing that contain answers that would be lost due to tokenization(particularly during truncation). It also groups the answer to the same question and context pair into a single row.

5 Baselines:

5.1 RNN Encoder:

We first tried building the baseline model with single recurrent encoder. We performed a standard word-level embedding on both questions and contexts. Initially, the embedded inputs are projected where they are downsampled into a fixed hidden size dimension, to reduce computation. Both the questions and contexts are fed to the model but as separate inputs. Then, all the hidden states of a single question are averaged together(over the embedding dimension) so as to obtain a single vector which should contain the encoded version of the semantic information of a question at the sentence level. The aggregated question vector is the multiplied to each context token latent representation using element-wise multiplication. This enabled us to perform some kind of query-aware context encoding. These query-aware context vectors are passed onto another recurrent module which are used as inputs for the end token classifier, while they are directly used as input for the start token classifier.

5.2 BiDAF Model:

The second model we used is BiDAF. BiDAF is a hierarchical multi-stage architecture model introduced in 2016 by University of Washington. In order to answer a question, the model checks the accompanying text that has the needed information by which the question will be answered. We can name the accompanying text as Context.We can say that BiDAF is a

- Closed domain model : This Baseline BiDAF model doesn't rely on pre-existing knowledge. It requires a context to answer a query.
- Extractive model : The model returns a substring of context relevant to the query.
- popular deep learning question and answer model.

Steps in BiDAF model :

1. Tokenization: The query(T) and the context(J) contents are tokenized and broken down into their constituent parts.

2. Embedding: In BiDAF, Embedding in performed in different levels on granularity:

   (a) Word Level Embedding: This is the first embedding layer where we substitute the resultant words from the above step using pre-trained glove embeddings into vectors containing numbers then passed on to the character level embedding layer. Glove means Global vectors for word representation.Glove is an unsupervised learning algorithm where word-word cooccurrence statistics from the dataset is used to get linear substructures in the word vector space. Mathematical operations are performed on these vectors which capture both syntax and semantics of the words. We can encounter some words not found in the huge Glove corpus called the out of vocabulary words.Glove assigns some random values to such inputs.

   (b) Character Level Embedding: The out of vocabulary words are handled by using Convolutional neural networks which work like a feature extractor for each word and numeric vector representation for each word is found by observing character level constitution. The output is similar to the output in the above step. The outputs obtained from Word level embedding and character level embedding are then vertically concatenated and sent to a highway network.

   (c) Highway network: A Highway network is a series of feed-forward layers with a gating mechanism. The relative contribution from words resulting from word level embedding and character level embedding are adjusted as while for out of context words, the Glove gives a random output and the character level embedding layer gives a better output. We are increasing the relative importance of the out of context words from CNN.

   (d) Contextual Embedding: The output from the highway network is passed to the final embedding layer called the Contextual embedding layer implemented with a bidirectional LSTM composed of forward and backward LSTM sequences. When we take Homophonic words, they should not be treated the same as the words that might spell the same but do not have the same meaning when used in a context. Contextual information of the words is not taken into account assigning the same numeric vector representation to actually different words, confusing the model.A word should also be understood in a contextual way deriving meaning from its surroundings. The output representation from bidirectional LSTM incor-

porates the contextual meaning of a word.

3. The attention Layer: Attention was introduced in 2016 and got popular very fast. A similarity matrix is generated by application of a comparison function to each column in context H and query matrix U. A cell in the similarity matrix represents the similarity between a particular context word and a query word. Context to query and query to context attention is computed for both the directions from the similarity matrix. The output of the layer are the query-aware vector representations of the context words . The context matrix, the context to query attention matrix and the query to context attention matrix for merged into a single matrix by a concatenation function. Given below is the diagram which shows the BiDAF Architecture.

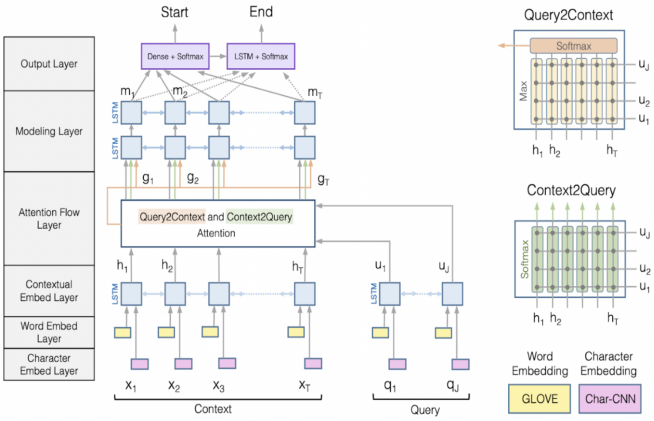
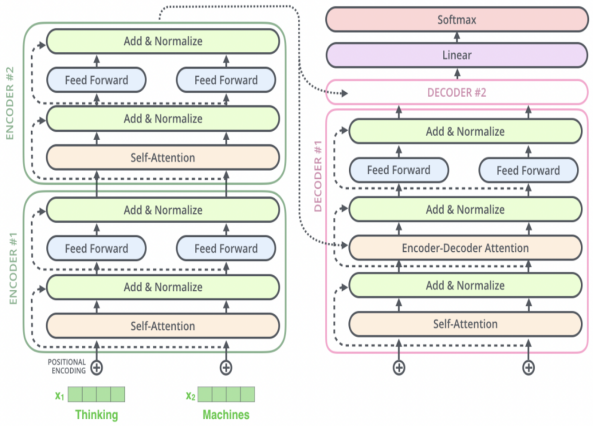


Figure 2: BiDAF architecture

The self-attention layer receives embeddings that are concatenated with positional encoding vectors in order to account for the order of input words. In contrast, the feed-forward layer is agnostic to dependencies between words, which means that operations can be executed in parallel in the feed-forward layer, leading to the impressive performance speedups associated with the Transformer. BERT model is pre-trained on 2 tasks. One of the tasks is Masked Language Model, where a percentage of the input tokens are masked and then predicted. The other task on which the BERT is pre-trained is Next Sentence Prediction, in order to learn the relationship between two sentences, which is not directly modeled by language modeling. Our experiments are all performed using the base, uncased implementation of BERT, which has 12 Transformer layers and 768 as its hidden size.



3: Transformers architecture

6 Approach:

6.1 Proposed Architecture:

- BERT(Bidirectional Encoder Representations from Transformers):

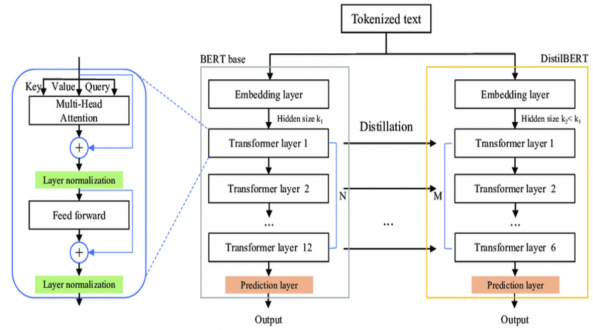
  The key innovation of the BERT model is applying bidirectional training of transformers to language modeling. BERT produces a state-of-the-art language model. At the highest level, its architecture is a trained stack of Transformer Encoders. Although we are using bi-directional LSTM in BIDAF, that captures forward and backward context but that is outperformed by the transformers in BERT. BERT gives better performance when compared to BIDAF because it includes transformers that use self attention, whereas in BIDAF, we are using only attention, that is how much attention we have to pay to other words in the sequence. Self attention layer in BERT helps the encoder look at other words in the input sentence as it encodes a specific word. The entire input sequence is read at once, so it would be more correct to define it non-directional. BERT has two layers. a self-attention layer and a feed-forward neural network layer.

- DISTILBERT

  As a part of an advancement, we will be updating the model to use DistilBERT which is a small, fast, cheap and light transformer model trained by distilling BERT Base. Compared to the BERT base uncased model, DistilBERT has 40% less parameters and runs 60% faster while preserving 95% of the BERT's performance. Knowledge Distillation could be seen as a transfer learning technique, which is a form of compression from a huge high precision model to a smaller one without losing too much in generalization.

  In DistilBERT the pooler and the token-type embeddings were removed. Reinforcement learning helps us to reduce the dimensions of a huge model and also reduce the training time by transferring knowledge between two models. Our smaller, faster and lighter model is cheaper to pre-train and we demonstrate its capabilities for on-device computations in a proof-of-concept experiment and a comparative on-device study. It uses a single linear layer, with no activation

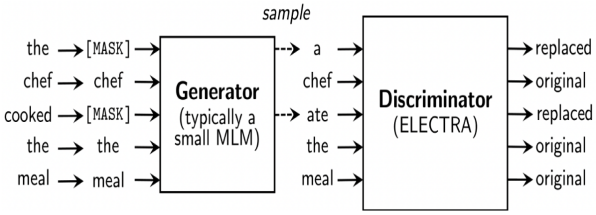
function, to reduce the 768 output dimension that comes from the DistilBERT backbone to 2.



DistilBERT Architecture

- ELECTRA

  ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) has the same architecture as BERT, but uses a new pre-training approach which aims to outperform the MLM strategy by replacing with a new strategy called as RTD (Replaced Token Detection) where the generator replaces tokens in the sequence trained as a masked language model and the discriminator (the ELECTRA contribution) attempts to identify which tokens are replaced by the generator in the sequence. At the end of training the generator is not useful anymore, so that only the discriminator needs to be saved. This new pre-training task allows the model to compute the loss over each and every input token (not only on the masked ones, as done in BERT), which is what sets apart ELECTRA in terms of convergence speed and performance on downstream problems.



ELECTRA Architecture

6.2 Prompt Tuning:

Large pre-trained language models, which are continuing to grow in size, achieve state-of-art results on many natural language processing (NLP) benchmarks. However, as models become larger, storing and serving a tuned copy of the model for each downstream task became impractical in the past few years. Researchers proposed a new method to solve the problem which is called as prompt tuning. Prompt tuning is more efficient and effective method for conditioning frozen models using tunable soft prompts. It is the approach of adding extra information for the model to condition on during its generation of Y. Normally, prompting is done by prepending a series of tokens P, to the input X, such that the model maximizes the likelihood of the correct Y. There are many other advantages of prompt tuning. It makes the model resilient to domain shift.

7 Experiments:

7.1 Baseline implementation:

The baseline model we have implemented have a total of 245202 parameters embedded in the model.

```
baseline_model = model.QABaselineModel(embedding_layer, device=DEVICE)
print(f"The baseline model has {baseline_model.count_parameters()} parameters")

The baseline model has 245202 parameters
```

We have chosen the hyperparameters by trial and error method, considering the fact that the baseline model is pretty light weight, take less time to run and we can also afford using higher batch sizes. We also set the number of training epochs to a higher value, to understand how well the model is learning and to also observe if the model is overfitting. To be clear, the validation set is not used to tune hyperparameters, but just as a reference to report results over unseen data. We have used Adam optimizer with learning rate of 0.0001.

7.2 BiDAF implementation:

BiDAF network is a hierarchical multi-stage architecture which is used to model the representations of the context paragraph at different levels of granularity. The attention mechanism we implemented follows the following improvements when compared to the previously popular attention paradigms and also the naive attention mechanism implemented in the baseline model. First, the attention layer is not used to summarize the context paragraph into a fixed-size vector. Instead, the attention is computed for every time step, and the attended vector at each time step, along with the representations from previous layers, is allowed to flow through to the subsequent modeling layer. This reduces the information loss caused by early summarization. Apart from this, we also used a memory-less attention mechanism. That is, while we iteratively compute attention through time, the attention at each time step is a function of only the query and the context paragraph at the current time step and does not directly depend on the attention at the previous time step. The hyperparameter to fine-tune the model are directly taken from the BiDAF paper where the number of epochs are 18, batch size is 60, the optimizer we will be using is Ada Delta optimizer and the learning rate is 0.5. But these models are not able to maintain long-term dependencies. To solve this issue, we started using transformers as encoders.

7.3 Transformers implementation:

Since we know that deeper architectures such as transformers can give out much better predictions, as a part of our project enhancement, we used various transformer networks(especially BERT related) to train our SQuAD Dataset. Given below are the models we have used to build the Question-Answering systems.

1.BERT (Bidirectional Encoder Representations from Transformers)
2.DistilBERT (a distilled version of BERT)
3.ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately)

Since pre-training such models is very expensive, we relied on pre-trained versions of them (where the pre-training tasks are different than question answering), publicly available through the HuggingFace Transformers library. Pre-trained models are wrapped into an ad-hoc PyTorch module, which attaches to them the output layer to be used for question answering. The main peculiarity of these transformer architectures allow the model to learn the context of a word based on all of its surroundings, at the same time, without having to perform a sequential scan of the input, as done in standard RNNs.

BERT: We used a pre-trained model of BERT which is trained on two main tasks. One of the task is Masked Language Modeling(MLM) which is performed by masking 15% of the words in each sequence by replacing them with a [MASK] token, so that the model is forced to predict the original value of the masked words, using the other available ones. Other task is the next sentence prediction(NSP) which is achieved by giving a pair of sentences as the input to the model. 50% of the cases are two subsequent sentences or a random picked pair otherwise, so that the model is asked to identify if the second sentence follows the first one. The first task, MLM, is used to gain a detailed understanding of the processed language, while the second task, NSP, is used to let the model have a broader comprehension of semantic connections.BERT is trained using both tasks together, with the goal of minimizing the combined loss. As a part of this task, we trained the BERT model for 3 epochs with a batch size of 32 and a learning rate of 0.00005. We have implemented prompt-tuning into the BERT and trained it for few epochs due to time constraint.

2. DistilBERT: Since, we know DistilBERT is better than BERT in terms of runtime and also it has about half the parameters of BERT. While most prior work investigated the use of distillation for building task-specific models, we leverage knowledge distillation during the pretraining phase. The hyperparameters used to fine-tune the model are same as the ones we used for the BERT model to fine-tune.

3. Electra: ELECTRA is a new pretraining approach which trains two transformer models: the generator and the discriminator. The generator's role is to replace tokens in a sequence, and is therefore trained as a masked language model. The discriminator, which is the model we're interested in, tries to identify which tokens were replaced by the generator in the sequence. We have fine-tuned the electra transformer for 3 epochs with Adam optimizer and a learning rate of 0.00005.

7.4 Evaluation:

The performance of the models is measured by using two metrics.Given a paragraph and a question regarding it, the model provides us with an answer. The answer given by the model is a text selected from the paragraph itself. These scores are computed on individual question-answer pairs.

Exact match: The metric measures the percentage of predictions that match any one of the ground truth values in an exact way. This metric is simple but is fairly strict as well. For each pair of the question and answer, if the characters of prediction output from the model matches exactly with the characters of the true answers , the EM score would be zero. Let us take an example to describe the strictness of this metric. Example: Let us suppose we got an answer 'cat' for some question asked to the model. And the true answer is 'Black cat'. Then the EM score for this example would be zero. We can also say that exact match is a binary measure (true/false).

F1 score: In general F1 score is the harmonic mean of precision and recall. We always desire true positive and true negative in our model but we might also get false positive and false negative.Accuracy tells us the percentage of correctly classified data instances over total amount of data instances. Accuracy is generally used a lot but it is not a good measure when we have a non-balanced dataset.Precision also known as positive predictive value gives us the measure of the only relevant data instances. Precision the number of true positives divided by the number of true positives plus the number of false positives. Recall also known as sensitivity or true positive rate is a measure of the ability of a model to find all the relevant cases within a data instances.It is the number of true positives divided by the number of true positives plus the number of false negatives. In a good model , we want both values of precision and recall to be one. For this reason we use the F1 metric which takes both recall and precision into account. F1 score would be high when both the precision and recall are high . In this case , precision is the fraction of number of shared words to the total number of words in the model prediction and recall is the fraction of number of shared words to the total number of words in the ground truth. And F1 score is the harmonic mean of both of these values.

8 Results:

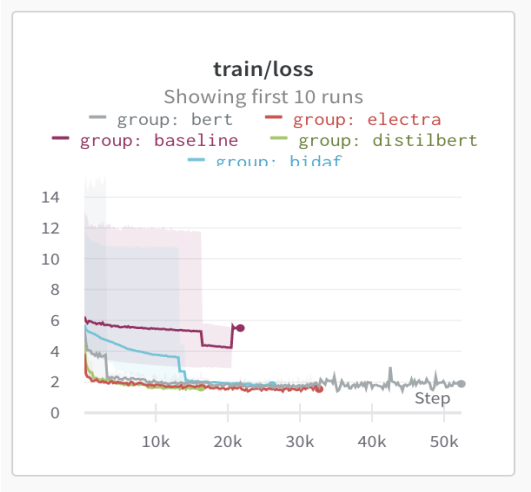
We have observed from our experiments that we have got the best values for our recurrent-based module BiDAF. It has achieved a 68% of F1 score and 55% of EM on the validation split on the entire dataset. In the case of the Transformer-based models we have got the best values for ELECTRA which has achieved an 84% of F1 score and 71% of EM.
Given below are the values of the metrics obtained by different models trained and fine-tuned using SQuAD v1.1.

Given below are the graphs for various metrics we

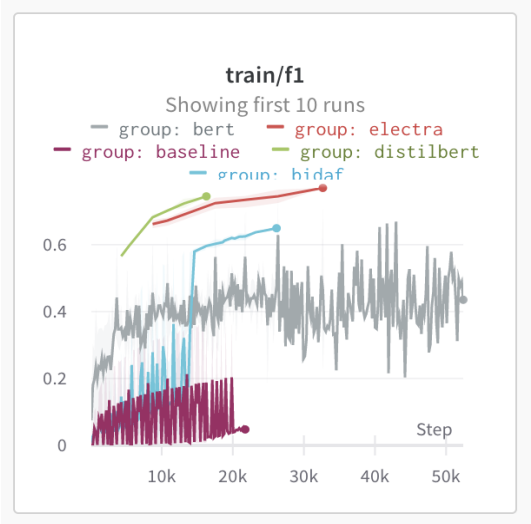
Model	Train-F1	Train-EM
Baseline	32.21	22.27
BiDAF	60.42	43.27
BERT	71.78	56.14
DistilBERT	60.72	55.95
ELECTRA	72.42	58.89

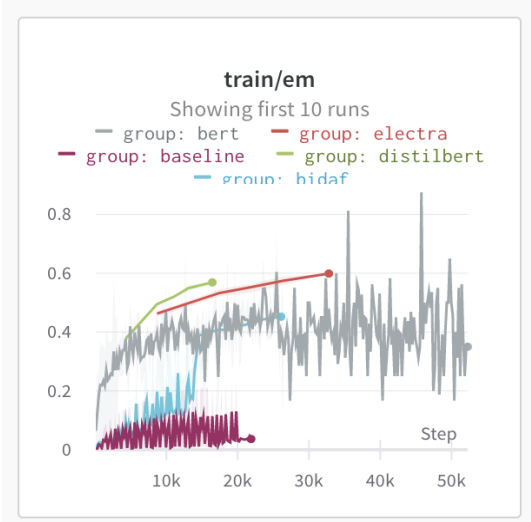
Model	Val-F1	Val-EM	Test-F1	Test-EM
Baseline	32.59	26.47	35.80	27.35
BiDAF	59.40	55.31	65.15	60.09
BERT	80.28	67.80	81.17	74.17
DistilBERT	77.26	66.97	79.44	73.64
ELECTRA	81.45	70.83	85.27	78.60

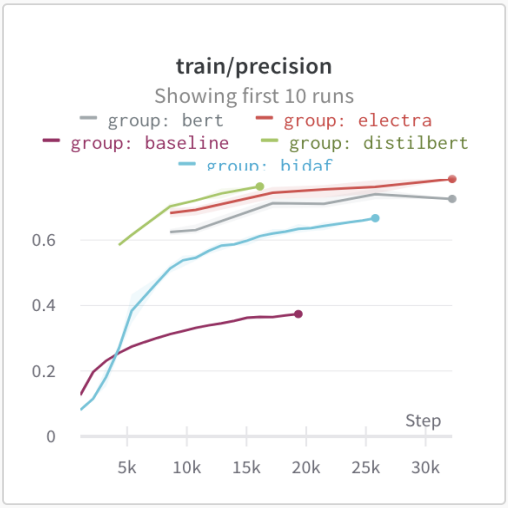
have calculated to evaluate the performance of the model. We have calculated precision, recall, f1 score, accuracy and EM metrics during training as well to see if the model is learning or over-fitting. It is very important to keep track of the metrics during training so that the we can have a knowledge on how the model is getting tuned. We can see that the loss value for all the models is reducing with every epoch.

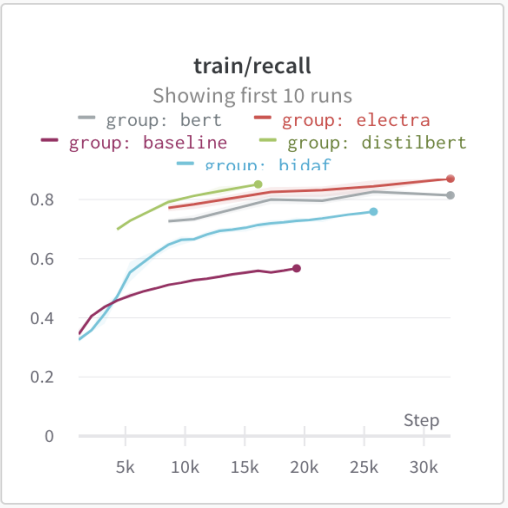


Here are the graphs showing the changes in the metrics after epoch of training for different models.

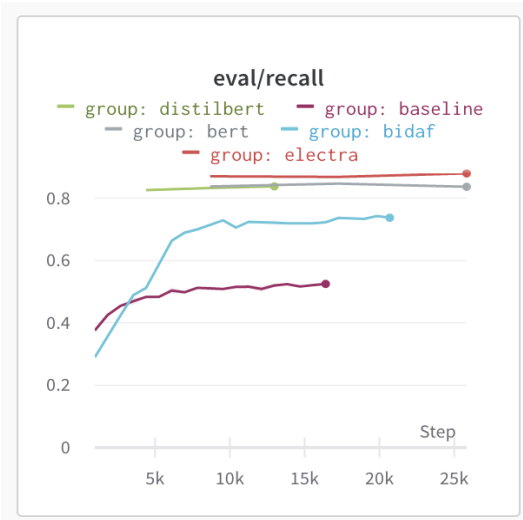


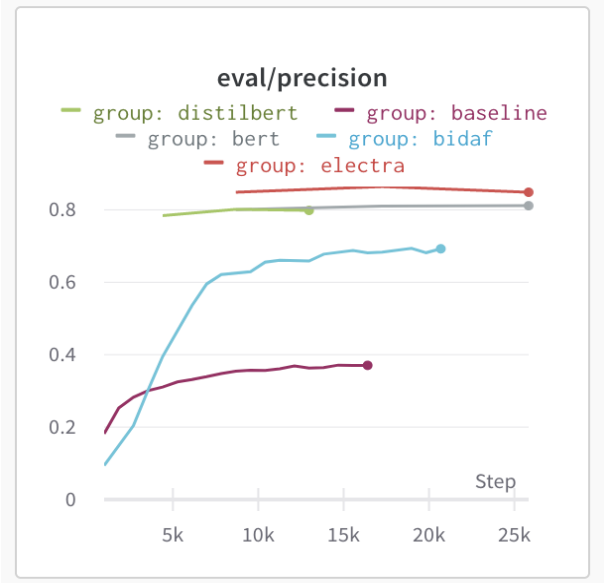


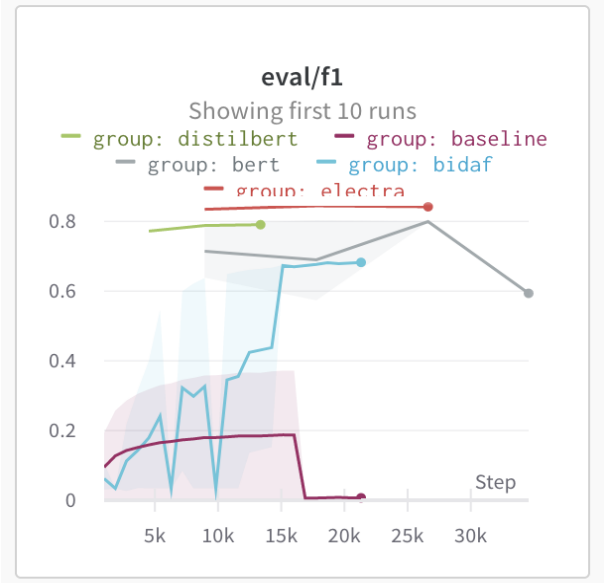


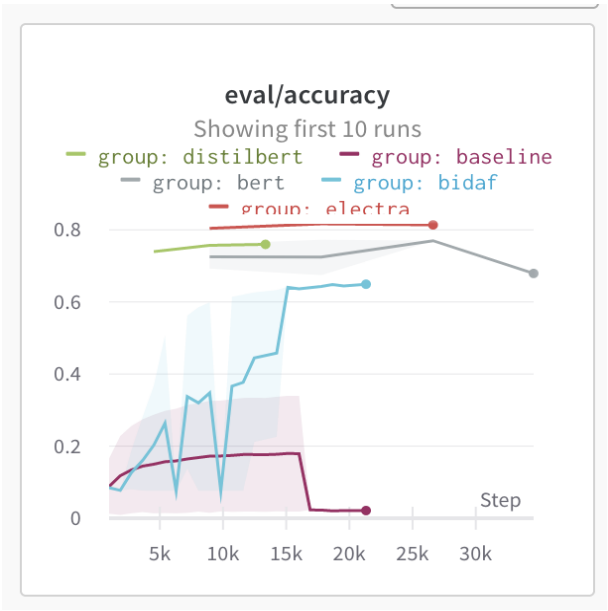


Given below are the graphs showing the changes in the values of all the metrics calculated on the test set for various models.









9 Error Analysis:

To get more insights on how the model works, we decided to concentrate more on the mis-classified examples from the dev SQuAD v1.1. While using BiDAF, the number of errors are around 4k. But with using Transformer Based model like ELECTRA, the number of errors decreased to 2000. Given below is one of the example where the answer shows the model's limitations in producing better outputs when math operations are expressed in natural language.

Context: In July 1888, Brown and Peck negotiated a licensing deal with George Westinghouse for Tesla's polyphase induction motor and transformer designs for $60,000 in cash and stock and a royalty of $2.50 per AC horsepower produced by each motor. Westinghouse also hired Tesla for one year for the large fee of $2,000 ($52,700 in today's dollars) per month to be a consultant at the Westinghouse Electric Manufacturing Company's Pittsburgh labs. Question: How much did Westinghouse pay for Tesla's designs?

Answers: 60000 in cash and stock and royalty of 250 per ac horsepower produced by each motor, 60000 in cash and stock and royalty

Prediction by ELECTRA: 60000

Prediction by BiDAF: 60000

There are few examples for which the predicted answers are contradicting each other. Given below is the example for such anomaly.

Context: Luther secretly returned to Wittenberg on 6 March 1522. He wrote to the Elector: "During my absence, Satan has entered my sheepfold, and committed ravages which I cannot repair by writing, but only by my personal presence and living word." For eight days in Lent, beginning on Invocavit Sunday, 9 March, Luther preached eight sermons, which became known as the "Invocavit Sermons". In these sermons, he hammered home the primacy of core Christian

values such as love, patience, charity, and freedom, and reminded the citizens to trust God's word rather than violence to bring about necessary change.
Question: How did Luther want people to bring about change?
Answers: trust gods word, love patience charity and freedom.
Prediction by ELECTRA: violence
Prediction by BiDAF: reminded citizens to trust gods word rather than violence

10 Contributions of group members:

Since this is a new field for everyone working on this project, we have to read many papers related to the project.The group has decided to work collectively on each of the subtasks.
1. Akhila Jetty - Worked on Data collection and data preprocessing. She helped in fine-tuning and training of Baseline and BiDAF model.
2. Neeharika Karanam - She implemented naive baseline and BiDAF model. She also guided everyone by referring to many papers and gathering information related to the models we have to build.
3. Nikitha Masanagi - She implemented BERT and DistilBERT models and trained them. She also implemented prompt tuning in BERT.
4. Roshitha Bezawada - She implemented Electra and tokenization function for transformers. She also did most of the paperwork and collecting information related to transformers and their implementation.
Everyone of us equally tried to understand the concept of prompt tuning and it's uses. We felt it would be great if we implement prompt tuning in transformers. We were not able to fine-tune it to a good extent but we have seen some improvements in the epochs we ran. Due to time-constraint, we were unable to report the results by prompt tuning.

11 Conclusion:

In this work, we addressed the problem of question answering on the SQuAD v1.1 dataset, by using both recurrent-based models and Transformer-based ones. As expected, our Baseline is outperformed by every other architecture, and models based on BERT are able to almost reach human level performance, when evaluated on the F1 score metric (for reference, that is around 90%). We mostly concentrated on implementing the baseline recurrent-based models(one of them is BiDAF network) and we used the same input for all the transfomer based models as well. We gave the same output interface to each and every model, so as to gain modularity and fairnerss in the evaluation metrics of the backbone. Many inferences can be taken from the results to understand the implemented models. We have also implemented character embeddings in BiDAF model. Prompt tuning did show good improvement in the results observed after every epoch. Apparantly, we have observed that working on SQuAD v1.1 is better than SQuAD 2.0. One of the interesting fact is that the conducted experiments show that both recurrent-based and Transformer-based models perform similar errors, such as those on word boundaries and question comprehension. The total number of errors, in the SQuAD v1.1 dev set, that are common in every model is exactly 940. From the given results of the metrics, we can say that transformers are much useful in predicting the output which is more similar to the human generated output.

11.1 Tools:

We will start off using model using BIDAF architecture on SQuAD dataset which is trained specifically for QA Task in PyTorch framework. All of the training and validation processes were executed on Google Colaboratory. Hugging Face, a library to build, train and deploy state of the art NLP models is used to load pretrained transformer models. We have also used Wandb to log all the training and evaluation metrics, along with model checkpoints and results.

12 References:

[1] L. Kodra and E. Kajo, "Question Answering Systems: A Review on Present Developments, Challenges and Trends", International Journal of Advanced Computer Science and Applications, vol. 8, no. 9, 2017 [Online]. Available:https://thesai.org/Downloads/Volume8No9/Paper_31- Question_Answering_Systems_A_Review_on_Present_Development s.pdf. [Accessed: 22- May- 2018].
[2] E. Brill, J. Lin, M. Banko, S. Dumais and A. Ng, "Data-Intensive Question Answering", Trec.nist.gov, 2018. [Online]. Available: https://trec.nist.gov/pubs/trec10/papers/Trec2001Notebook.AskMSRF inal.pdf. [Accessed: 22- May- 2018].
[3] H. Madabushi and M. Lee, "High Accuracy Rule-based Question Classification using Question Syntax and Semantics", Aclweb.org, 2018. [Online]. Available: http://www.aclweb.org/anthology/C16- 1116. [Accessed: 23- May- 2018].
[4] E. Riloff and M. Thelen, "A Rule-based Question Answering System for Reading Comprehension Tests", 2018. [Online]. Available: https://pdfs.semanticscholar.org/4454/06b0d88ae965fa587cf5c167374 ff1bbc09a.pdf. [Accessed: 23- May- 2018].
[5] Trait Larson, Johnson (Heng) Gong, Josh Daniel "Providing a Simple Question Answering System by Mapping Questions to Questions.", Technical report, Department of Computer Science, Stanford University, 2006
[6] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. "DistilBERT, a distilled version of BERT: smaller, faster, cheap and lighter". In: CoRR abs/1910.01108 (2019). arXiv: 1910 . 01108. URL: http://arxiv.org/abs/1910.01108
[7] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev,

and Percy Liang. “SQuAD: 100,000+ Questions for Machine Comprehension of Text”. In: (2016). URL: https://nlp.stanford.edu/pubs/rajpurkar2016squad.pdf
[8]. K. Dwivedia and V. Singh, “Research and reviews in question answering system,” in Proceedings of International Conference on Computational Intelligence: Modeling Techniques and Applications,2013, pp. 417 – 424
[9] Senevirathne, K. U., N. S. Attanayake, A. W. M. H. Dhananjanie, W. A. S. U. Weragoda, A. Nugaliyadde, and S. Thelijjagoda. ”Conditional Random Fields based named entity recognition for sinhala.” In 2015 IEEE 10th International Conference on Industrial and Information Systems (ICIIS), pp. 302-307. IEEE, 2015
[10] Leon Derczynski, Alan Ritter, Sam Clark, Kalina Bontcheva,(2013) “Twitter Part-of-Speech Tagging for All: Overcoming Sparse and Noisy Data “ Proceedings of Recent Advances in Natural Language Processing, pages 198–206
[11] J. Ramos(2003) “Using TF-IDF to Determine Word Relevance in Document Queries”Technical report, Department of Computer Science, Rutgers University, 2003
[12] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut. 2017. Text Summarization Techniques: A Brief Survey. ArXiv e-prints (2017). arXiv:1707.02268
[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention is all you need. In Advances in Neural Information Processing Systems”. In: (2017). url: http://papers.nips.cc/paper/7181-attention-is-all-you- need.pdf.
[14] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention. 2020. arXiv: 2010.01057 [cs.CL].
[15]Dwivedia, S., Vaishali, S.: Research and reviews in question answering system. International Conference on Computational Intelligence: Modeling Techniques and Applications (2013)
[16] Shekarpour, S., Endris, K. M., Kumar, A. J., Lukovnikov, D., Singh, K., Thakkar, H., Lange, Ch.: Question Answering on Linked Data: Challenges and Future Directions. World Wide Web Conference (2016).
[17]D. Cohn, Z. Ghahramani and M. Jordan, “Active Learning with Statistical Models”, Journal of Artificial Intelligence Research, vol.4, 1996. [Online]. Available: https://jair.org/index.php/jair/article/view/10158. [Accessed: 22- May- 2018].
[18]T. Mikolov and G. Zweig, “Context Dependent Recurrent Neural Network Language Model”, 2012. [Online]. Available: https://www.microsoft.com/en-us/research/wp- content/uploads/2012/07/rnn_ctxt_TR.sav_.pdf. [Accessed: 24- May- 2018].
[19]Boukoros, Spyros, Anupiya Nugaliyadde, Angelos Marnerides, Costas Vassilakis, Polychronis Koutsakis, and Kok Wai Wong. ”Modeling server workloads for campus email traffic using recurrent neural networks.” In International Conference on Neural Information. Processing, pp. 57-66. Springer, Cham, 2017.
[20]Bidirectional Attention Flow for Machine Comprehension Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, Hannaneh Hajishirzi
[21]Mike Schuster Yonghui Wu et al. “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation”. In: CoRR abs/1609.08144 (2016). arXiv: 1609.08144. url: http://arxiv.org/abs/1609.08144.
[22]”The Power of Scale for Parameter-Efficient Prompt Tuning” Brian Lester Rami Al-Rfou Noah Constant