Question Answering Model

Praveen Kumar Sridhar

Northeastern University, USA
sridhar.p@northeastern.edu

May 03, 2022

1 Abstract

Our goal is to develop a question-and-answer model, that given a reading comprehension prose and a set of questions, must answer these questions using the knowledge gained from the reading comprehension prose. The data set we use is the Stanford Question Answering Dataset (SQuAD), specifically SQuAD1.1. It is a reading comprehension dataset composed of 100,000+ questions posed by crowdworkers on a collection of Wikipedia articles, with each question's answer being a text segment or span, from the corresponding reading passage. We look at the data to see what kinds of reasoning are needed to answer the questions. We want to tackle this problem in two ways: by developing an information retrieval model and experimenting the BERT model on the dataset. We also intend to compare the results obtained from different approaches.

2 Introduction

We are using the SQuAD1.1 data set which consists of 100,000+ question-answer pairs on 500+ articles.
Attribute Information:
1. id: An index
2. title: The title of the prose or the context paragraph
3. context: The prose for the reading comprehension
4. question: The question we need to find the answer for from the context paragraph
5. answers: A dictionary that contains the answer text and the answer start index
6. is_impossible: A flag to indicate if the question can be answered.

3 Data Analysis:

As a part of EDA, the initial observations showed that for majority of answers, the length less than 10 words(fig.1).Also, there are no missing values in the dataset and all the questions are answerable.

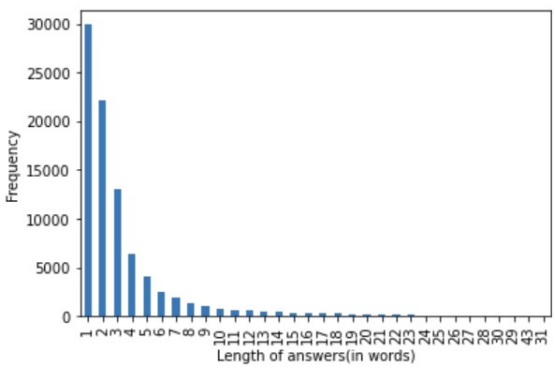
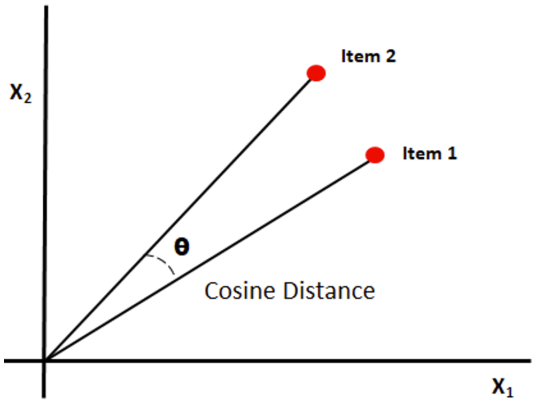


Figure 1: Frequency plot of answers length(in words)

4 Modeling:

4.1 Information Retrieval approach

Information retrieval (IR) is the process of finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).
In the context of our project, it means finding answer. With this approach, we are able to find the sentence from the paragraph which contains the answer phrase for a particular question.
We have used a Vector Space Models which converts words into vectors. It can capture context and represent it using the vectors, which means preserving the semantic and syntactic relationship between words.
The Vector Space Model works on the concept of similarity. It makes an assumption that relevance between a document and the query is directly related to the similarity between their vector representations.Which means document(D1) with a higher similarity score with the query(Q) will be more relevant than document(D2) with a lower similarity score.
In context of our project a 'document'(D) refers to a sentence present in the given paragraph(P).
Pre-processing of Data
As a part of pre-processing, we have combined the query(Q) and the given paragraph(P) to get the final text corpus which will be processed further. We have also replaced the less frequent words with an unknown token (i.e.<UNK>) to prevent the model from facing ambiguous situation while encountering any new data. We have used Gensim's simple_preprocess() method to convert a document into a list of tokens (unicode strings)
IR Embeddings
i. **Custom Trained Embeddings:**
1. Word2Vec model with skip-gram mode enabled trained on the SQuAD dataset.
2. fastText with skip-gram mode enabled again trained on the SQuAD dataset.
ii. **Pre-trained Embeddings:**
1. GloVe(Global Vectors for Word Representation)
2. Word2Vec embedding trained on google news articles
Mathematical Explanation of the Model
Cosine similarity: To compare the two documents, cosine similarity metric is used. It is used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space.The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance because of the size they could still have a smaller angle between them. Smaller the angle, higher the similarity. After calculating the similarity scores for each sentence/document(D) and query pair, we choose the sentence from context as our most probable answer which has the highest similarity score with the question.



(a) Cosine Similarity Representation in Vector Space

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

(b) Cosine similarity formula

Figure 2: Cosine Similarity

Drawbacks of using IR approach
i) IR approach helped us in finding the sentence/document(D) which contains the probable answer but not the exact answer phrase itself.

ii)The IR approach does not capture the idea of Contextual Embeddings: representations of words in context. The methods like word2vec or GloVe learned a single vector embedding for each unique word 'w' in the vocabulary. However with contextual embeddings, each word 'w' will be represented by a different vector each time it appears in a different context.

4.2 Transformer models, and the need for it?

Given that the problem statement is to get the answer phrase from the context(paragraph) that best answer the question, the input to a model should be the context and question, ideally separated by a special token. Let the special token be [SEP].The model is ideally going to do the same thing we tried to do in the information retrieval approach, match the question with the answer in the context, but instead of returning the entire answers sentences, the model we are trying to train should return the most probable start and end indices, (i.e. the model's output will be the start and end indices). This indicates that this model must have a very good long term memory because the input will be the context (a large enough paragraph with numerous sentences) followed by the question. This input is huge enough that the memory offered by LSTMs is insufficient, so we need something better to tackle this. Attention is the solution; in fact, attention is all you need!

Self-attention

Self-attention creates a score matrix to determine how much words corresponds to each other. Attention takes in three inputs a Query(Q), Keys(K), Values(V). What ends up happening is that queries are mapped to keys, then they are scaled down by dividing them with square root of the dimensions of the keys to avoid exploding gradients, then softmax is applied to get probabilities. Then finally, the probabilities are multiplied with the value vector to get the final attention matrix. The following is what attention does:

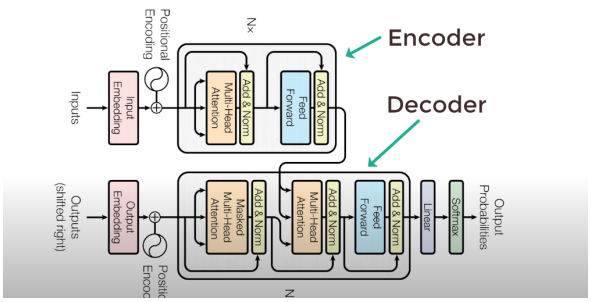
$$Attention(Q, K, V) = softmax(\frac{Q, K^T}{\sqrt{d_k}})V$$

Multi-head attention

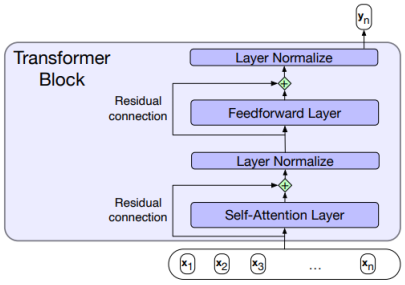
But self-attention vectors tend to have a tendency to weight attention to itself, too high. To handle this, the Attention module splits its Query, Key, and Value parameters N-ways and passes each split independently through a separate Head. All of these similar Attention calculations are then combined together to produce a final Attention score.

Transformers

Transformers use attention to do tasks like machine translations. The architecture of a transformer model is in fig.3a. Let us now examine transformers through the lens of machine translation. The encoder in machine translation learns the language syntax, grammar, and everything else required for the machine to comprehend the language, and then creates encoded input for the decoder. As the name implies, the decoder decodes the encoded input to produce output in a different language. In a nutshell, the encoder excels in understanding a language and hence providing a good encoding for a given input sentence, whereas the decoder excels at learning to generate texts. To answer questions, we will not generate any text; instead, we will predict the most likely start and end indices. So the encoder would be more than adequate for this.



(a) Transformer Architecture



(b) Look into the Transformer Block

Figure 3: Transformers

4.3 BERT

Previously we discussed, to solve this problem we only need encoders, and that's what BERT is, it is encoder layers stacked on top of each other continuously. Hence the name, BERT which stands for Bidirectional encoders representation from Transformers. The original BERT consisted of the following:

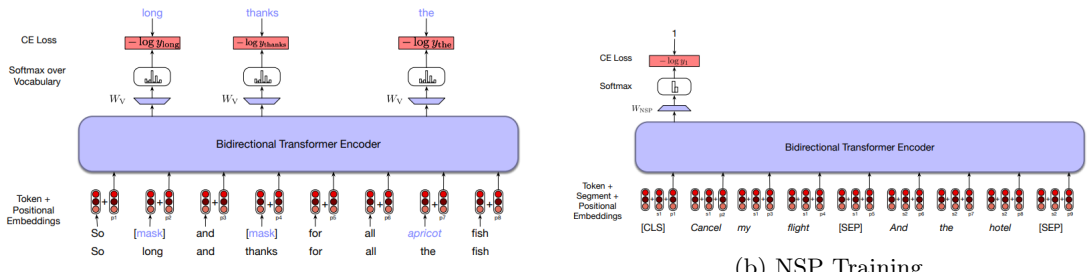
- A subword vocab consisting of 30,000 token generated using wordpiece algorithm,
- Hidden layers of size 768,
- 12 layers of Transformer Blocks(fig.3b), with 12 multi-head attention layers each.

There are two stages to using BERT: pretraining and fine-tuning. Pretraining is the process of learning some kind of representation of meaning for words or phrases by the processing of a vast amount of text. The following stage is to apply these models to various tasks such as named entity tagging or, in our case, question answering. This is known as fine-tuning.
Bert is pretrained using 2 tasks: Masked language modeling (MLM) and Next sentence prediction(NSP)(fig.4). These tasks are done in parallel while training.
Masked Language modeling(MLM) Masked Language Modeling is the original method for training bidirectional encoders. MLM, like language model training, employs unannotated text from a vast corpus of text. The model is shown a series of sentences from the training corpus, with a random sample of tokens from each training sequence chosen for use in the learning task. Once selected, a token can be utilized in one of three ways:

- It is replaced with the unique vocabulary token [MASK].
- It is replaced with another token from the vocabulary, randomly sampled based on token unigram probabilities.
- It is left unchanged.

In BERT, 15% of the input tokens in a training sequence are sampled for learning. Of these, 80% are replaced with [MASK], 10% are replaced with randomly selected tokens, and the remaining 10% are left unchanged.
Using a bidirectional encoder, the MLM training goal is to predict the original inputs for each of the masked tokens (refer fig.4a). The cross-entropy loss resulting from these predictions drives the training process for all of the model's parameters. The self-attention process involves all of the input tokens, but only the sampled tokens are used for learning. To produce a probability distribution over the vocab of the masked tokens, the output vector from the final transformer layer for each of the masked tokens is multiplied by a set of classification weights $W_v \epsilon R^{|V| \times d_h}$ and then through a softmax function to give the required predictions over the vocabulary. We can then use cross-entropy loss for each of the masked item- negative of the log probability for each of the masked words.

$$y_i = softmax(W_v h_i)$$



(a) MLM training

(b) NSP Training

Figure 4: BERT FINE-TUNING

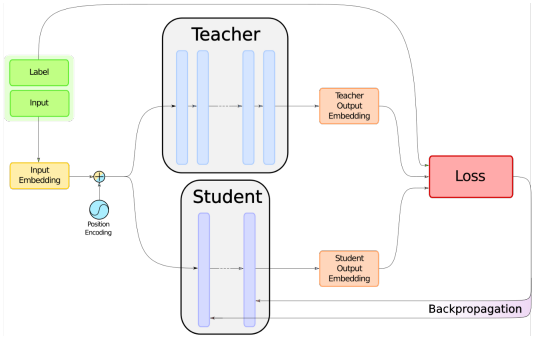
Next Sentence Prediction(NSP)
The focus of MLM is on predicting words from surrounding contexts with the goal of producing effective word-level representations, thereby learning context between words. However, our applications also need us to understand context between sentences. BERT learns this through the NSP task. In BERT, 50% of the training pairs consisted of positive pairs, and in the other 50%

the second sentence of a pair was randomly selected from elsewhere in the corpus. The NSP loss is determined by how efficiently the model distinguishes between true and random pairs (refer fig.4b). Following tokenization with the subword model, the token [CLS] is prepended to the input sentence pair, and the token [SEP] is placed between the sentences.During training, the output vector from the final layer associated with the [CLS] token represents the next sentence prediction. As with the MLM objective, a learned set of classification weights $W_{NSP}\epsilon R^{2\times d_h}$ is used to produce a two-class prediction from the raw [CLS] vector.Cross entropy is used to compute the NSP loss for each sentence pair presented to the model.

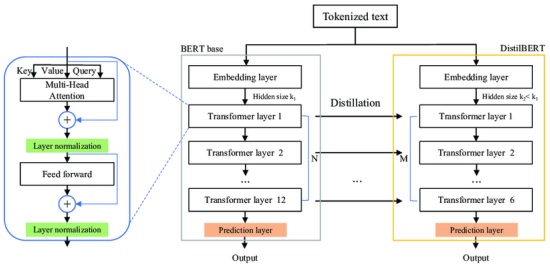
$$y_i = softmax(W_{NSP}h_i)$$

4.4 DistilBERT

This is the model obtained as a result of distilling the BERT model to a much smaller model with similar performance as BERT. The distilling process is done through a Teacher-student network(fig.5a), where the teacher is the BERT model with its weights frozen, and it forces the student (distilbert) to behave the same as the teacher i.e give similar predictions. Distilbert(student) has half the number of layers as BERT (6 layers of transformer blocks) (fig.5b)and it does not make use of the positional encoders.



(a) Teacher student network



(b) Distil Bert architecture

Figure 5: Distil BERT

The student is trained with a distillation loss over the soft target probabilities of the teacher:

$$L_{ce} = \sum_i t_i * log(s_i)$$

where t_i is the probability estimated by the teacher and s_i is the probability estimated by the student. This objective results in a rich training signal by leveraging the full teacher distribution. The final training objective is a linear combination of the distillation loss L_{ce} with the supervised training loss, in our case the masked language modeling loss L_{mlm}. It's a small, fast, cheap and light model with 40% less parameters as bert-base-uncased, while running 60% faster than BERT, while preserving 95% of BERT's performance.

4.5 ALBERT

ALBERT stands for A Lite BERT. The goal of this is to improve the training and results of BERT architecture by using different techniques like parameter sharing, factorization of embedding matrix, Inter sentence Coherence loss.
Factorization of the Embedding matrix:The input layer embeddings and hidden layer embeddings are the same size in the BERT model. The developers of ALBERT, however, separated the two embedding matrices. This is due to the fact that input-level embedding (E) requires just context-independent learning to be refined, whereas hidden level embedding (H) requires context-dependent learning. When compared to BERT, this step results in an 80% reduction in parameters and a minor drop in performance.
Cross-layer parameter sharing:The authors also suggested that parameters be shared between levels of the model to improve efficiency and reduce redundancy.

Inter Sentence Coherence Prediction: ALBERT, like BERT, trained using the Masked Language model. However, rather of applying the NSP (Next Sentence Prediction) loss, ALBERT employed a novel loss known as SOP (Sentence Order Prediction). The downside of NSP is that it tests for coherence as well as the topic to to identify the next sentence. The SOP, on the other hand, just looks for sentence coherence.

4.6 Fine-tuning

Once we have the pretrained model we need to fine-tune it to solve our question answering problem statement. To do that we take the model BERT/DistilBERT/ALBERT and attach a head to it. This head is going to help us output the most probable start and end indices. This head is going to be another layer where we multiply the weights of each token with a start index weight matrix and then apply softmax to get the most probable start index (refer fig.6). To obtain the end index the process is similar, we would multiply with end index weight matrix and then apply softmax again to get the most probable end index.

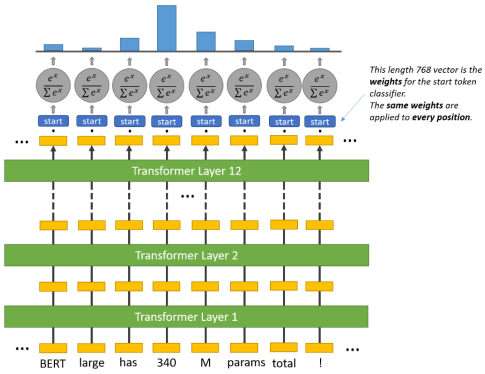


Figure 6: Bert Finetune in detail

4.7 Ensemble

Using the Ensemble Learning method, in which we use a combination of multiple models to boost the overall performance, we got better results. Combining BERT and DistilBERT improved both our Exact match and F1-scores.

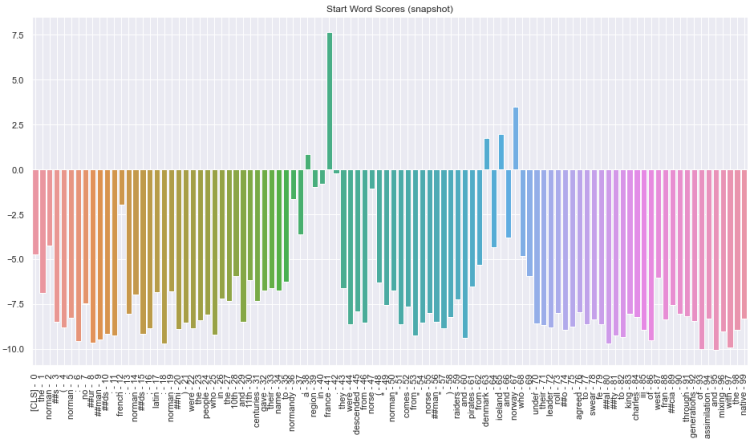
5 Results

As displayed in the (fig.7a) the question is "In what country is Normandy located?" and the answer our model predicted is "France" which is correct as expected.
Also, as explained in the previous sections, it can be observed in (fig.7b) that the highest probability for answer_start_index is corresponding to the word "france" and similar plot is observed for answer_end_index and hence the answer we get here is "france".
The table below shows that we began with Information Retrieval using word2vec on context combined with questions with low frequency words replaced, resulting in a 72.5 %.

Model	**Train Accuracy**	**Validation Accuracy**
Information retrieval with Word2Vec on context + questions with low frequency words replaced	71.2%	72.5%
Information retrieval with Word2Vec on context + google news	NA	72.2%
Information retrieval with fastText on context + questions with low frequency words replaced	71.0%	71.%
Information retrieval with fastText on context + questions with low frequency words not replaced	70.9%	70.8%

```
Context The Normans (Norman: Nourmands; French: Normand
s; Latin: Normanni) were the people who in the 10th and
11th centuries gave their name to Normandy, a region in
France. They were descended from Norse ("Norman" comes f
rom "Norseman") raiders and pirates from Denmark, Icelan
d and Norway who, under their leader Rollo, agreed to sw
ear fealty to King Charles III of West Francia. Through
generations of assimilation and mixing with the native F
rankish and Roman-Gaulish populations, their descendants
would gradually merge with the Carolingian-based culture
s of West Francia. The distinct cultural and ethnic iden
tity of the Normans emerged initially in the first half
of the 10th century, and it continued to evolve over the
succeeding centuries.
question In what country is Normandy located?
answer france.
```

(a) Sample Output Question and Answer



(b) Start indices score visualization

Figure 7: Results and visualization

However, we can see a significant improvement when implementing BERT and BERT like models. The results for which are in the following table:

Model	**Exact match score**	**F1 score**
BERT	62.53%	74.15%
DistilBERT	63.23%	75.02%
Ensemble	66.44%	77.68%
Albert	***70.9%***	***80.95%***

After using the models listed in above table, we found that the ALBERT performs the best and results in an F1 score of 80.95%. Hence, we are moving forward with this model.

6 Conclusion

Meaning of results:
The end goal of our project was to find the answer to a given question from the paragraph. We used different models and word embedding techniques to discern which pair gives us the best accuracy possible. The results show that the model 'Ensemble' gives us the best accuracy.
What could be improved about the project in future work?
In future, our work can be extended by utilizing normal tokenizer instead of fast tokenizer as we did in our current model. We can run the model for more epochs so that we get a good performance and better fit for the model. Alternatively, we can also do an ensemble of various BERT models where we could average on the scores to get better results. In addition to these things, we can also experiment with other BERT models such as Roberta, ELECTRA etc.

7 References

1. https://arxiv.org/pdf/1606.05250.pdf
2. https://arxiv.org/pdf/1806.03822.pdf
3. https://rajpurkar.github.io/SQuAD-explorer
4. https://huggingface.co/blog/bert-101
5. https://huggingface.co/datasets/squad
6. https://radimrehurek.com/gensim/models/word2vec.html
7. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/default/15848021.pdf
8. https://youtu.be/xI0HHN5XKDo
9. https://youtu.be/l8ZYCvgGu0o
10. https://huggingface.co/docs/transformers/index
11. https://arxiv.org/abs/1910.01108

8 Appendix

The code along with plots and descriptions can be found at the github repository mentioned below:
https://github.com/PraveenKumarSridhar/Question-Answering-model