Question Answering System

Krishna Rukmini Puthucode
Georgia State University
Atlanta, Georgia
Email: kputhucode1@student.gsu.edu

Ruthuparna Naikar
Georgia State University
Atlanta, Georgia
Email: rnaikar1@student.gsu.edu

Kausik Amancherla
Georgia State University
Atlanta, Georgia
Email: kamancherla@student.gsu.edu

***Abstract*—Question answering (QA) is a computer science discipline within the fields of information retrieval and natural language processing (NLP), which is concerned with building systems that automatically answer questions posed by humans in a natural language.With advances in deep learning, neural network variants are becoming the dominant architecture for many Natural Language Processing tasks. In this project, we apply several deep learning and Natural Language Processing techniques to question answering. Our system takes natural language questions expressed in English along with the context and attempts to provide short answers.**

***Index Terms*—Question-Answering, BERT, Natural Language Processing, Pre-training BERT**

I. Introduction

Search engines have always been effective when dealing with large amounts of data, and they are generally good at retrieving documents that contain the user query.The next logical step will be to try to automate the part of the process where the user searches for a response in the retrieved text or document.Natural Language Processing along with Deep Learning techniques is making possible what was once considered to be a fantastical capacity for computers to answer any and all human questions. Under Natural Language Processing (NLP), Question Answering or QA is a discipline that enables users to retrieve answers from machines for questions posed in natural language.

This problem comes under the Open-domain Question Answering (ODQA) which is a form of language task that asks a model to provide natural-language answers to factoid questions. Since the true response is objective, evaluating model output is easy.he term ”open-domain” refers to the lack of meaning for any factual question that is asked randomly. In the example above, the model just takes the question as an input, but there is no article on ”why Einstein didn’t win a Nobel Prize for his theory of relativity,” where the expression ”the law of the photoelectric effect” is possibly discussed.

Our project tries to use various approaches to solve this using by using various pre-trained models along with training existing BERT models on the data set of our choice.This gave us a chance to create a baseline model to evaluate the performance of the BERT model and the architecture design of our project.

In the rest of this introduction we provide a brief discussion of the dimensions of question answering into 3 different types, followed by a sketch of the history of natural language question answering , an overview of current approaches We hope that this introduction will provide a useful general perspective on question answering research which complements the detailed technical contributions of the other papers under Literature survey.

In general there are three types of QA system which have vary on how the information is being retrieved from a given data-set

- **Information Retrieval Factoid Question Answering**:finding short text fragments on the Web or in another series of documents to address a user’s query
- **Knowledge-based question answering**: Answering a natural language issue by converting it to a formal structured database query
- **Various information sources**: Searching through different data-sets for keywords and querying the related documents.

There has been a dramatic surge in interest in natural language question answering since the introduction of the Question Answering track in the Text Retrieval Conferences, beginning with TREC-8 in 1999.However the attempt make a computer system answer user queries dates way back.Some of those methods included

- List-Structured database system
- Graphic database system
- Text-based system
- Logical Inference system
- Using New Deep Learning Techniques

The new deep learning techniques have shown promising results in picking accurate answers for a given passage and question.These models are generally classified into two parts

- **Retriever Model**: This model uses the classic non-learning-based TF-IDF features or dense embedding vectors of text produced by neural networks
- **Readers Model**: This model learns to solve the passage by extracting answer’s from a given context document.This is generally done by Bi-Directional LSTM or BERT Universe

II. Literature Survey

From the paper ”BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” by Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova [6] we have tried to understand how it works behind the scenes. As the author of the paper state ”BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be finetuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task specific architecture modifications.” [6] This was important as Pre training it with the data we need ,plays a huge role in the results we obtain in the project.So it is a important stating point.

The next paper that we referred was the the ”SQuAD: 100,000+ Questions for Machine Comprehension of Text” by Pranav Rajpurkar and Jian Zhang and Konstantin Lopyrev and Percy Liang. This was our primary dataset to train our BERT model as a dataset crowdworkering on a set of Wikipedia articles, where the answer to each question is a segment of text from the corresponding reading passage.

The paper ”Open domain question answering using Wikipedia-based knowledgemodel ” by Pum-Mo Ryu,Hyunki [5] Kim talks about building multiple answer matching modules based on different types of semi-structured knowledge sources of Wikipedia, including article content, info boxes, article structure, category structure, and definitions.This gave us an aim to find the documents which have the highest probability of having the answer to our question. Here, we have employed a multistep process.This inspired us to approach implemented here is Keyword Extraction.Where the Keyword extraction is the automated process of extracting the words and phrases that are most relevant to a document.

Keyword Extraction generally involves

- Candidate Keywords/Keyphrases:A list of candidate keywords or keyphrases from a document. Using Scikit-Learn’s CountVectorizer we generate keywords/key phrases.
- Embeddings: Converting both the document as well as the candidate keywords/keyphrases to numerical data sentence-transformers package creates high-quality embeddings that work quite well for sentence- and document-level embeddings We are using Distilbert as it has shown great performance
- Cosine Similarity: Find the candidates that are most similar to the document

III. Implementation

The lack of sufficient training data is one of the most significant challenges in NLP. While there is a vast amount of text data available, we must divide it into the many different fields in order to construct task-specific datasets. And we just end up with a few thousand or a few hundred thousand human-labeled training instances when we do this.To help bridge this gap in data, researchers have developed various techniques for training general purpose language representation models using the enormous piles of unannotated text on the web (this is known as pre-training). These general purpose pre-trained models can then be fine-tuned on smaller task-specific datasets, e.g., when working with problems like question answering and sentiment analysis.This approach results in great accuracy improvements compared to training on the smaller task-specific datasets from scratch. BERT is a recent addition to these techniques for NLP pre-training; it caused a stir in the deep learning community because it presented state-of-the-art results in a wide variety of NLP tasks, like question answering.

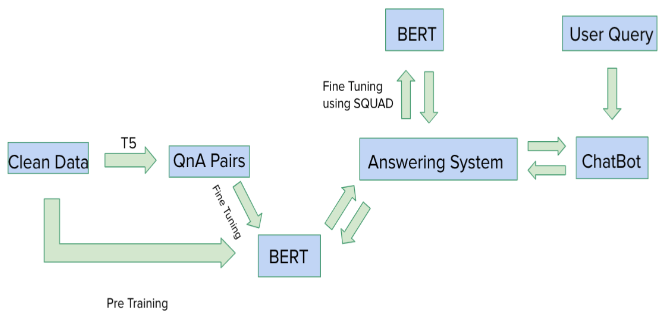


Fig. 1. Our System Architecture

BERT employs a novel technique known as Masked LM (MLM), in which it masks terms in a sentence at random and then attempts to predict them. In order to predict the masked expression, the model looks in both directions and uses the entire context of the sentence, including both the left and right surroundings. Unlike previous language models, it considers both the previous and subsequent tokens.

For this project, we wanted to pre-train a BERT and not use a pre-trained one for two reasons. Primarily because we aimed to customize our own model to meet our dataset needs and also explore the research in this particular domain. Pre-training a BERT model from scratch has not been an much explored domain and hence we opted to pre-train the model from scratch with the Wikipedia data which is our corpus.

1) Finding the dataset: The data is 300 articles of Wikipedia. We have preprocessed the textual data.

2) Train a tokenizer: We choose to train a byte-level Byte-pair encoding tokenizer (the same as GPT-2), with the same special tokens as RoBERTa. Let’s arbitrarily pick its size to be 32000.We have trained a byte-level BPE (rather than let’s say, a WordPiece tokenizer like BERT) because it will start building its vocabulary from an alphabet of single bytes, so all words will be decomposable into tokens.

3) Train a language model from scratch: We will now train our language model using transformers.As the model is BERT-like, we’ll train it on a task of Masked language modeling, i.e. the predict how to fill arbitrary tokens that we randomly mask in the dataset. This is taken care of by the example script

4) Check that the LM actually trained: Aside from looking at the training and eval losses going down, the easiest way to check whether our language model is learning anything interesting is via the FillMaskPipeline. Pipelines are simple wrappers around tokenizers and models, and the ’fill-mask’ one will let you input a sequence containing a masked token (here, ¡mask¿) and return a list of the most probable filled sequences, with their probabilities.

5) Fine-tuning with Cloud TPUs: We now can fine-tune our new language model on a downstream task of Part-of-speech tagging.As mentioned before, The word endings typically condition the grammatical part of speech. We can use the run$_{n}er.pyscriptfromtransformers.$

We have fine-tuned our model with SQuAD dataset.SQuAD 1.1 The Stanford Question Answering Dataset (SQuAD) is a popular question answering benchmark dataset. BERT (at the time of the release) obtains state-of-the-art results on SQuAD with almost no task-specific network architecture modifications or data augmentation. However, it does require semi-complex data pre-processing and post-processing to deal with (a) the variable-length nature of SQuAD context paragraphs, and (b) the character-level answer annotations which are used for SQuAD training

Fig 2 is the loss function as our BERT model is getting pre-trained.We can see that as it is iterating through every step,the loss function.It starts 6.1 at 1000 steps to 1.56 at 1000000 steps.This is a clear indication that our model is learning the dataset.

IV. Key Results

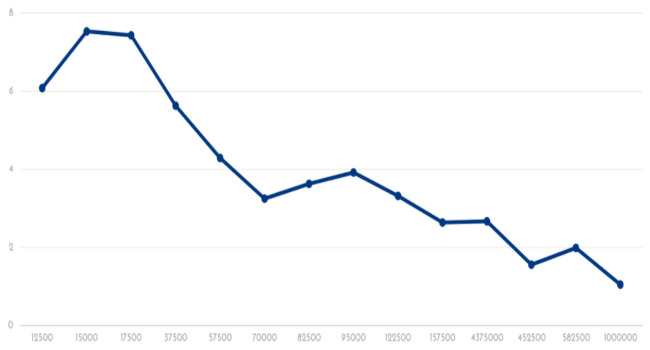
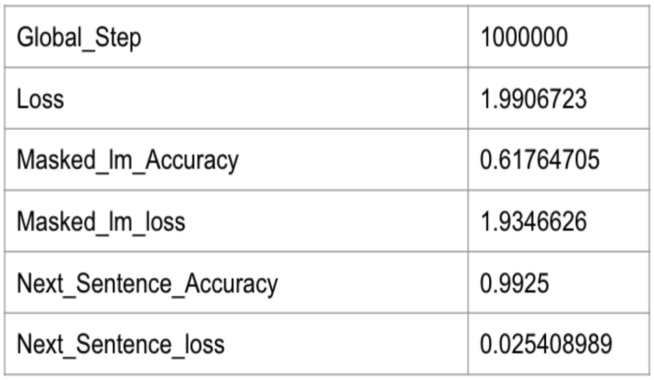


Fig. 2. BERT loss function as it is pretrained



Global_Step	1000000
Loss	1.9906723
Masked_lm_Accuracy	0.61764705
Masked_lm_loss	1.9346626
Next_Sentence_Accuracy	0.9925
Next_Sentence_loss	0.025408989

Fig. 3. BERT Pre-Training Evaluation Results

Fig 3 is the evaluation report of our BERT model after 1000000 iterations of training. The final loss function ends up being at 1.99 and the masked accuracy comes to 61.7.But what was encouraging was that the next sentence accuracy was at 99.25 percentage

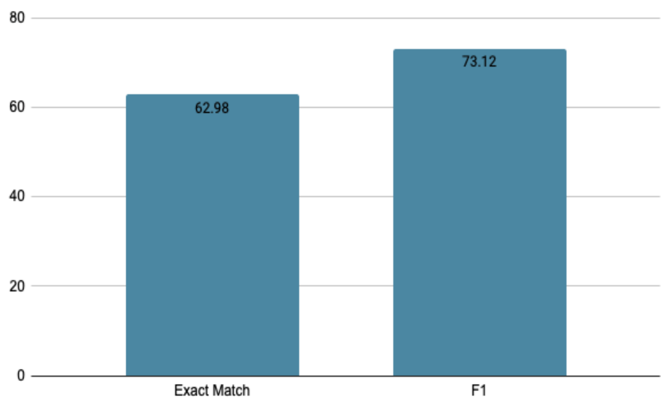


Fig. 4. BERT Fine-Tuning Evaluation Results

After Fine-Tuning our BERT model,the evaluation report from Fig 4 give a F1 score of 73.12 percentage and an Exact match score of 62. It didn't beat the top notch models shows that our methodology is in the right direction.

V. What Worked

The pre-training of BERT with our Wikipedia dataset. We achieved a loss of 1.99 which is a good indication of how the implementation has been successful. Also the other evaluation metrices masked lm accuracy and masked lm loss have also been good.

VI. Challenges

Failure is an unavoidable part of any project process: it's the degree of failure that makes the difference. If a task fails, there are ways to reallocate resources and get back on track. But a systemic collapse will derail the whole project.Natural Language Processing is relatively a new and unexplored field of Data Science . Although the objectives weren't fully met, in this section we try to elaborate on what lessons we learnt.

A. *Question Answer Pair Generation*

The first aim of our project was to generate question answers pairs from the given corpus of data. We built an transformer model T5.This model is a sequence-to-sequence question generator which takes an answer and context as an input, and generates a question as an output. It is based on a pretrained t5-base model.The model is trained to generate reading comprehension-style questions with answers extracted from a text. The model performs best with full sentence answers, but can also be used with single word or short phrase answers.We used the corpus of 30000 Wikipedia articles to generate question-answers pairs with an aim to fine-tune the BERT model. We were successful to generate the question-answers pairs from Wikipedia corpus using T5.

We implemented the above said methodology and we were able to generate question answers pairs but the shortcoming we faced here was the datasize. We aimed to use this question-answers pairs to fine-tune the pre-trained BERT model. The primary requirement was the json format for the question-answer pairs and converting such large amount of question into json was time-consuming. We ran into a lot of errors doing the conversions because of large number of question-answer pairs generated. Also, the generated pairs had to be mapped with an unique ids for fine-tuning the BERT. ID creation was a humongous task in terms of processing. Hence we could not use the generated pairs for fine-tuning our model.

B. Document Retrieval

The matching of a user query against a collection of free-text records is known as document retrieval. These documents may be something with a lot of unstructured text, like newspaper papers, real estate reports, or manual paragraphs. User questions can vary in length from multi-sentence complete explanations of information requirements to a few sentences. We aimed to retrieve the relevant document which has the highest possibility of having the answer to the using question. The strategy we employed is keyword extraction.

Keyword extraction is the automated process of extracting the words and phrases that are most relevant to an input text.BERT is a bi-directional transformer model that helps us to convert phrases and documents into vectors that accurately represent their context. Next, we converted both the document as well as the candidate keywords/key-phrases to numerical data. We used BERT for this purpose as it has shown great results for both similarity- and paraphrasing tasks.We used the sentence-transformers package as it allows us to quickly create high-quality embeddings that work quite well for sentence- and document-level embeddings.We implemented this using Distilbert as it has shown great performance in similarity tasks, which is what we are aiming for with keyword/keyphrase extraction.In the final step, we found the candidates that are most similar to the document. We assumed that the most similar candidates to the document are good keywords/keyphrases for representing the document. To calculate the similarity between candidates and the document, we used the cosine similarity between vectors as it performs quite well in high-dimensionality.We took the top 5 most similar candidates to the input document as the resulting keywords.

The next step in our implementation was processing the user question and map them to the relevant keywords. We again implemented the cosine similarity to get the top N documents.

The next step was to generate context for the given question query. Context Retrieval from the documents is the concept of extracting the passage/sentences which potentially have the answer to the given question.To achieve this we got a list of all sentences in top documents. We used tokenizer to tokenize all our sentences and used lemmas of the words instead of the original words to imprve accuracy. We used the BM25 ranking function to rank all our sentences against the given query.

BM25is a search engine ranking feature that ranks matching documents based on their importance to a given search query.We then extracted the top N results from the step above and build a paragraph out of all those N sentences.

The reason for the above implementation to not show desired outputs was for two major reasons. First is generating few keywords for such a huge corpus of data was inefficient. Using a heave load model such as BERT was also not successful in fetching relevant keywords because wikipedia articles have hundereds of important keywords and ranking them is extremely difficult. The next shortcoming we experienced was to processed and build similarity matrices of such huge data. Although we manage to implemented this,this approached under-performed to fetch appropriate documents

VII. Conclusions and Future Work

We have described our Question-Answering System which inputs user question along with a context to fetch the answer. We have also described how we implemented our customized BERT by pre-training and fine-tuning it from scratch. Although it was a very intensive task, we have managed

As a scope to improve, we plan to work on algorithmic part of our system. The above described system is the initial version and can be improved in many ways. Although this is an inferential study, we plan to make this work for entire Wikipedia corpus. Working a way out the amount of data is something to ponder upon.

The next important goal for us is to overcome all the challenges mentioned and work for a roundabout.Larger benchmark datasets with more sophisticated correct answer verification will boost analysis and model training. Building an sophisticated pipeline to answer all open-domain questions seems a a hurdle to overcome.

References

[1] QA with Wiki: improving information retrieval and machine comprehension by Rohan Sampath,Puyang Ma

[2] Using wikipedia at the trec qa track. In Proceedings of TREC 2004. David Ahn, Valentin Jijkoun, Gilad Mishne, Karin Mller, Maarten de Rijke, and Stefan Schlobach.

[3] Modeling of the question answering task in the YodaQA system.Petr Baudis and Jan ˇ Sediv ˇ y. 2015

[4] Mining knowledge from Wikipedia for the question answering task Davide Buscaldi and Paolo Rosso. 2006

[5] Open domain question answering using Wikipedia-based knowledge model Pum-Mo Ryu,Hyunki Kim

[6] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding byJacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova Google AI Language

[7] Benchmarking question answering systems by Ricardo Usbeck,Michael Röder

[8] End-to-End Open-Domain Question Answering with BERTserini Wei Yang, Yuqing Xie

[9] Question answering system on education acts using NLP techniques Sweta P. Lende; M. M. Raghuwanshi

[10] FERRUCCI, D., AND LALLY, A. UIMA: An architectural approach to unstructured information processing in the corporate research environment

[11] Large-scale simple question answering with memory networks Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015

[12] Semantic parsing on freebase from question-answer pairs. In EMNLP (2013),BERANT, J., CHOU, A., FROSTIG, R., AND LIANG, P.

[13] Overview of a question answering engine. arXiv preprint arXiv GALLAGHER, S., ZADROZNY, W., SHALABY, W., AND AVADHANI, A. Watsonsim

[14] Answer extraction as sequence tagging with tree edit distance. YAO, X., VAN DURME, B.,