The Question and Answers system with search engine based on fine-tuning Bert and Doc2vec

Hanxiao Chen
ECE department – MADS
University of Victoria

Abstract

In this paper, we established a question answering system based on the pre-trained Bert. The fine-tuned model could get the most appropriate question-context pair from the database and extract the correct answer from the context. The pre-trained data source is Squad V2.0. This paper mainly focuses on the finetuning of the Bert model. By setting appropriate hyperparameters and parameters, the optimized Bert model can become a robust question answering system. Despite the limitations of the running environment and database coverage, we were able to build a QA system that could answer questions through its knowledge base. One of the major innovations in this paper was the design of a search engine. Through the doc2vector model and inverted index methods, we could match the input questions to the existing questions in the database. It could help to answer any related questions with possible answers rather than provide 'no results' like other Bert models under SQuAD v2.0.

1 Introduction

Machine reading comprehension and question and answering is a hot topic in the field of natural language processing (NLP). It aims to make machines read and understand like humans, while there is no restriction on the background knowledge for machines to read and understand the context. Thereby it helps to decrease the cost of accessing information. Besides, machine reading comprehension and Q&A mainly involve deep learning, NLP and information retrieval. It has high research value and various landing scenes. It enables computers to help humans find accurate answers quickly in a large number of texts.

1.1 Reading comprehension and question and answering system

Machine Reading Comprehension and Question and Answer Task (Q&A) is a task where the trained Q&A system can use the given questions and one or more contexts to find the answers. It includes the retrieval dialogue system, the generative dialogue system, and the task dialogue system. This paper is focusing on the generative dialogue systems based on NLP. The main steps include constructing a deep semantic model by a machine learning algorithm, extracting features by combining statistical laws from sentence analysis, and training the model to learn the conversation rules from a large number of existing dialogues, and finally predicting the results with the trained model. Generally speaking, there are three kinds of questions: simple (factoid) questions that can be answered with simple facts, and the answer is usually just a name entity; semi-complex questions, which are slightly complicated narrative questions with longer answers; complex questions, or "opinion" questions, which are the most complicate ones and they ask the opinions without standard answers. The Q&A tasks involved in this paper are all based on the first type, and the generated answers should be perfectly matched to the one in the text.

One of the traditional approaches to solving the Q&A tasks is the feature-based logistic regression (typically used as a baseline). With the popularity of deep learning, more and more deep learning models have obtained SOTA (state of the art) results on such problems, including the DrQA model proposed by Danqi Chen from Stanford that integrates a large-scale open-source database, the neural recurrent sequence labelling model from Baidu, etc... However, after Google put forward the Bert (Bidirectional Encoder Representations from Transformers) model, only some simple

fine-tunings are needed to obtain SOTA results in the English dataset SQuAD, and it can surpass human performance. The fine-tuned Bert model has been used in this article.

1.2 Related papers

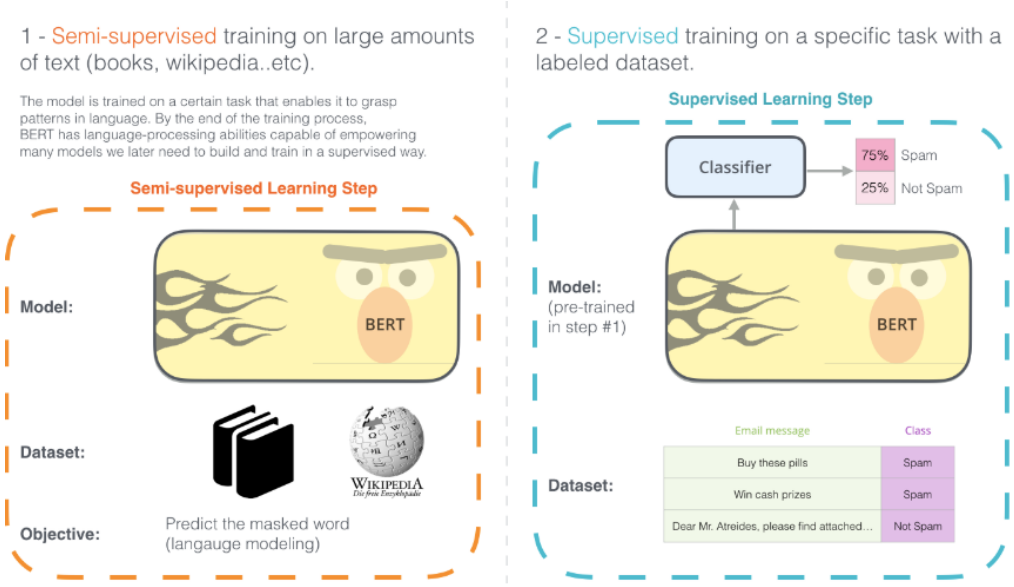
The Bert is built based on the Transformer model, a model developed by Vaswani in 2017[1] in their famous paper "Attention is all you needed". The transformer has abandoned the traditional CNN and RNN structures, and the network is entirely composed of an attention mechanism. To be more precise, Transformer consists of and only consists of self-attention and Feed Forward Neural Network. The Bert model [2] proposed by Google in 2018 is a kind of giant Transformer model. Bert adopts a two-stage model. The first stage is the pre-training of the NLP model; secondly, the fine-tuning mode is used to solve the downstream tasks. The main feature is that it adopts a bidirectional language model similar to ELMO in the pre-training stage.

There is another branch in the Transformer model family. In 2019, Google Brain and CMU came up with a new model, Transformer-XL [3]. In order to improve the long-distance dependency modelling ability of language models, it proposed a piece-level recursion mechanism and designed a better relative position encoding mechanism, which was more effective for encoding long text. On this basis, Google designed a new model, XLNet [4]. It is a queuing language model starting from the unsupervised pre-training method, considered the advantages and disadvantages of the autoregressive language model and the autocoded language model, which is the most outstanding model recently in the downstream tasks of natural language processing.

Since the pre-trained Bert model is more friendly to small projects based on local environments, our paper used the pre-training Bert model and did the finetuning to build a Q&A system. In terms of the application of Bert, we refer to relevant materials. For example, Deveshree et al. (2020) [5] compared the performance of the Bert model with other machine learning models in a SQuAD and concluded that the Bert model still has outstanding performance on small data, but the training time was relative too long. We also referred to the book "Hands-on Question Answering Systems with Bert" by Navin et al. (2021) [6] to understand the application of Bert in Question Answering Systems.

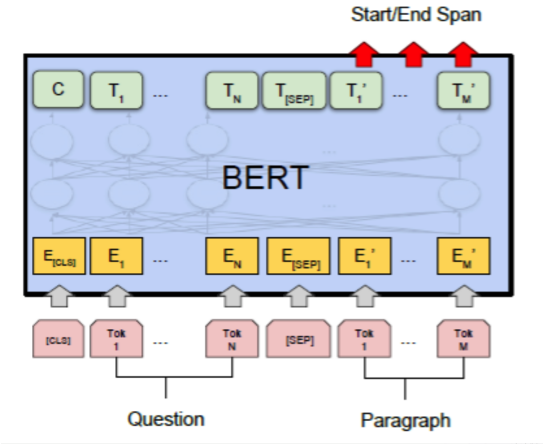
1.3 Problem's design

As mentioned earlier, the purpose of this paper is to design a Q&A system. Our design was to build a bot through the "Discord" platform that can ask and answer questions through a dialogue format. As we explained, the most suitable model will be the Bert, so we would give a detailed description of Bert, and our datasets from here.



As shown in the figure above, BERT consists of two main steps. Firstly it uses Encoder of Transformer to train the bidirectional language model BERT, and then it picks up the task-specific

classifier after BERT [2]. Among them, the pre-training step mainly consists of Masked Language Model (MSM) and Next Sentence Prediction (NSP). In this paper, we focus on the second step, which is finetuning the pre-trained model to train a reading comprehension Q&A system. The main idea of the finetuning parts are shown in the figure below, the input sequence of the model is the embeddings corresponding to sentence pairs, which consist of questions and text containing answers, and they are separated by a special separator "[SEP]". As in other downstream tasks, the first token of the input sequence is the special classification embedding "[CLS]", and the input sequence is the sum of token embeddings, segmentation embeddings, and position embedding.



The output of BERT is the encoding vector corresponding to each token. Assuming that the dimension of the vector is D, then the whole output sequence is $T^{N \times D}$, where N is the length of the whole sequence. Since the answer consists of consecutive tokens in the text, the process of predicting the answer is essentially the process of determining where the tokens at the beginning and the end of the answer are located. Thus, after the fully connected layer, we obtain $O^{N \times 2} = FC(T^{N \times D})$. Where FC represents the fully connected layer, $O^{N \times 2}$ is the logit value of each token as the beginning and the end of the answer, and after the Softmax layer, the corresponding probability value p_i (below function) is obtained. After post-processing the data, the predicted answer is obtained.

$p_i=\frac{e^{S\times T_i}}{\sum_jS\times T_i}$

$$P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_i}}$$

1.4 Datasets

We used SQuAD v2.0 as the database [3], which is a reading comprehension dataset consisting of mainly Wikipedia-based questions and answers. One of the major changes in SQuAD 2.0 is that it combines both the answerable questions based on SQuAD 1.1 and additional 50,000 non-answerable questions for testing the function of providing no answer.

The structure of a paragraph in the dataset consists of a series of ['title', 'paragraphs', 'qas', 'answers']. In addition, this dataset has the following two features: (1) all questions are factoid questions, and (2) the answer to a question consists of a named entity. The summary of the SQuAD is shown in the following figure.

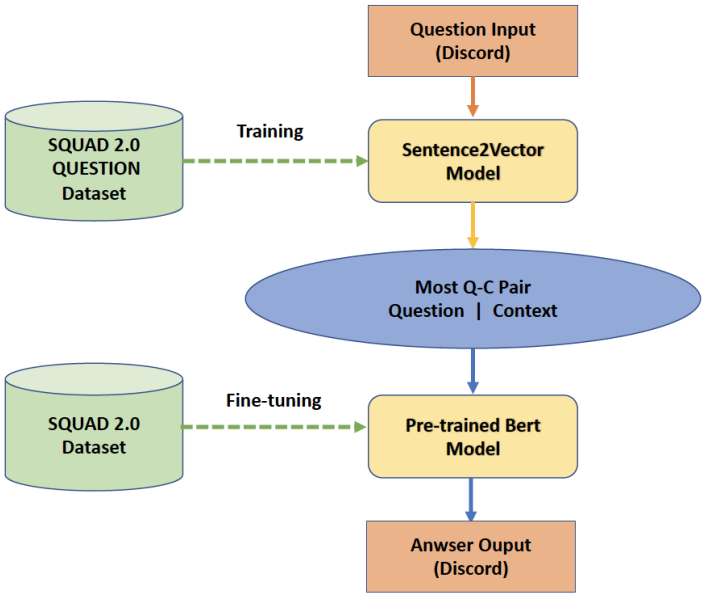
	SQuAD 1.1	SQuAD 2.0
Train		
Total examples	87,599	130,319
Negative examples	0	43,498
Total articles	442	442
Articles with negatives	0	285
Development		
Total examples	10,570	11,873
Negative examples	0	5,945
Total articles	48	35
Articles with negatives	0	35
Test		
Total examples	9,533	8,862
Negative examples	0	4,332
Total articles	46	28
Articles with negatives	0	28

from【8】

There are two JSON files been used in our dataset from the SquAD (train.json and dev.json). We used the train set to finetune the Bert Model and use the development data set to test and evaluate the results.

2 Approaches

This project mainly included two models, they were Question-to-vector and Finetune Bert. Among them, , the Question-to-vector model could turn the input questions into 2d vectors and compare the similariry with the pre-set questions in the SQuAD database with corresponding context and paragraphs. The second model was the finetuning Bert. It was based on the pre-trained Bert base model from Google, and we finetuned with different parameters and hyperparameters.



2.1 Search engine based on the inverted index

The aim of our model is to predict answer from a huge database containing millions of paragaraphs. However, The BERT model only capable of predicting an answer from a given paragraph(we call it context). Therefore there must be a mechanism which tells the BERT model where to look for answer, and this is acheieved by our search engine which is based on inverted index. Since we are dealing with a large database we use leverage BigData framework "PySpark"

- **Steps:**

1) Word tokenize each question-context pair. Create an inverted index for these tokens.

2) Export the inverted index in mysql database.
3) To find percentage of similarity of the given question with existing question-context pair from the database we compute cosine similarity. The greater the cosine similarity the more relevent is the context.
4) Find the most relevent context and pass this to the BERT model to make prediction.

2.2 Bert model

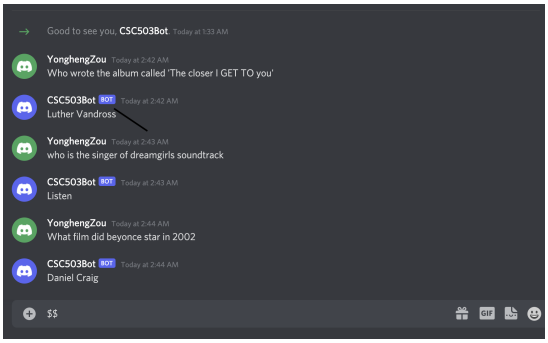
As explained in the Introduction part, the Bert model is a suitable tool to solve the tasks of matching sentence or paragraph and finding the answer from the text. Since Bert adds the Next Sentence Prediction task in the pre-training phase and the Self Attention mechanism comes with Attention effects for words in Sentence A and any words in Sentence B, Bert works well for scenarios with a high demand for deep semantic features.

- **Steps**

1) Data preprocessing: Firstly, we tokenize the questions and context, i.e., converted the sentence into a character level sequence, and then connected the question sequence and context and distinguished it by "[SEP]". The total length of the sequence after padding was a hyperparameter was set to 400 in our project.
2) Model Training: The model consists of two different parts. A Search Engine based on inverted index which finds the most similar question context pair. This context is used by BERT to predict the answer. The BERT model we downloaded and used is Bert-Base, Uncased: 12-layer. 768-hidden,12-heads,110M parameters. Once we have the pretrained model we then attempt to Fine tune it on our dataset. However, due to the large dataset size used for both training and evaluating and also the limitation of GPU on Google Colaboratory, we could not test multiple settings. We tried our best with limited resources. We used batch size 5, epoch 3, decay 1e-6, learning rate 5e-5. This is not the optimal parameters as we had to work with restricted hardware resources.

2.3 Discord bot

In order to facilitate a user interface which makes it convenient for the user to ask questions to our model we deploy a chatbot in Discord Server. The questions input by the user are redirected to out QA-model using API and the output of the model is shown to the user as a response from the bot. A sample screenshot of our bot is shown below.



3 Results

Our model is a fact based system. The answer to question must contain at least one named entity. Our model cannot answer opinion based questions. We divided the SQUAD dataset into training set, development set and test set in ratio 70-20-10. Our finetuned model gives 62% accuracy in the testset. We notice that it performs poorly in inference based question when the information is not explicitly mentioned in the text. We also find that our model performs poorly when the question is paraphrased, in words different from the ones present in the paragraphs.

4 Conclusion and future work

We have created a question answer system and deployed it in Discord server. Once a question is enetered, Our model first uses the search engine based on inverted index to narrorow down to the most relevent context(paragraph). Then the BERT model finds the answer to the question within the filtered context. A pretrained BERT model was developed and was finetuned on our dataset. The final result of this custom model is quite satisfactory and get an accuracy of 62%. A couple of things could be done in future to improve this model. Firstly, Due to the large dataset size and the limitation of GPU on Google Colaboratory, we could not test multiple settings. We could upgrade our hardware capability and explore multiple settings to fine-tune the best model. Secondly, instead of using a search engine based on inverted we could use a doc2vector model to find relevent context to a given question.

References

[1] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need [C]//Advances in Neural Information Processing Systems. 2017: 5998-6008.

[2] Devlin J, Chang M W, Lee K, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[3] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860, 2019.

[4] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov and Quoc V. Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding. arXiv preprint arXiv:1906.08237, 2019.

[5] Patel, D., Raval, P., Parikh, R., & Shastri, Y. (2020). Comparative Study of Machine Learning Models and BERT on SQuAD. ArXiv, abs/2005.11313.

[6] Navin Sabharwal, Amit Agrawal, Hands-on Question Answering Systems with BERT, DOI:https://doi-org.ezproxy.library.uvic.ca/10.1007/978-1-4842-6664-9, Pages 97-137,2021.

[7] Jay Alammar ,The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning). Visited at July 15th at https://jalammar.github.io/illustrated-bert/

[8] Rajpurkar, Pranav et al. “Know What You Don't Know: Unanswerable Questions for SQuAD.” ArXiv abs/1806.03822 (2018): n. pag.

[9] Le Q, Mikolov T. Distributed representations of sentences and documents[C]// International conference on machine learning. 2014: 1188-1196