International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/CSEIT2172139

Deep Learning Based Question Answering Search Engine

Mrunal Malekar
Department of Electronics and Telecommunications, Vishwakarma Institute of Technology, Pune, Maharashtra, India

Article Info
Volume 7, Issue 3
Page Number: 25-32

Publication Issue :
May-June-2021

Article History
Accepted : 01 May 2021
Published : 05 May 2021

ABSTRACT

Domain based Question Answering is concerned with building systems which provide answers to natural language questions that are asked specific to a domain. It comes under Information Retrieval and Natural language processing. Using Information Retrieval, one can search for the relevant documents which may contain the answer but it won't give the exact answer for the question asked. In the presented work, a question answering search engine has been developed which first finds out the relevant documents from a huge textual document data of a construction company and then goes a step beyond to extract answer from the extracted document. The robust question answering system developed uses Elastic Search for Information Retrieval [paragraphs extraction] and Deep Learning for answering the question from the short extracted paragraph. It leverages BERT Deep Learning Model to understand the layers and representations between the question and answer. The research work also focuses on how to improve the search accuracy of the Information Retrieval based Elastic Search engine which returns the relevant documents which may contain the answer.

Keywords : Bert, Transformers, Elasticsearch

I. INTRODUCTION

Information Retrieval and Question Answering is one the most growing research areas in Computer Science discipline. Question Answering systems are a step ahead of the Information Retrieval systems which mainly help in extracting the relevant information from data and aims at identifying the document which might contain answer to the search query. Question Answering system consists of three major components namely: Query/Question Processing, Information Retreival, Answer Extraction. QA Technology presented in this article aims at finding the correct answer to the asked question using BERT Transfer learning. Elastic Search helps in information retrieval and extracts top ten ranked paragraphs which may contain answer. These paragraphs are retrieved from the huge textual data and are retrieved using TF-IDF Scoring retrieval method. The work focuses on increasing the search accuracy or relevancy of the extracted documents along with aiming to increase accuracy and F1 score of the BERT deep learning Model which helps in finding the answer from the paragraph.

NLP has been used in recent days to improve retrieval of information. A lot of research has been happening on how noun phrases, giving weight to query, lemmatization and stemming can help in improving information retrieval accuracy[2].

In [1] , Explicit Semantic analysis (ESA) has been used to create a weighted query vector. ESA uses machine learning to create a interpreter which is semantic in nature that it maps text into concepts as per its input relevance. Using cosine similarity , relevant documents have been extracted for the weighted question query. In [3] multi stream question answering has been used which selects the final answer from a ranked set of answers. It checks for question and its compatibility with the answer, the redundancy of answers. In its primary step many question answer systems extract candidate answer in parallel for the asked question. Then in the secondary step, a classifier is responsible for evaluating all the candidate answers and then giving each of them a category whether correct or incorrect and a confidence value[0 to 1]. In the end, The answer with highest confidence value is chosen as the answer. [4]

The accuracy and success of a Question Answering system relies heavily on the accuracy of its information retrieval system. In our work we have leveraged the power of Elastic Search as a information retrieval engine and researched on how can the search relevancy be improved. Elastic Search uses BM25 to score each document and then return top ranked documents as per user input query. The research focuses on improving scores of the returned top ranked documents by optimizing the hyperparameters used by BM25 algorithm. It aims at using phrase level boosting. chunking, acronyms expansion to increase the relevancy of the documents that are returned upon search. There can also be cases of questions where very few of the words in the question form a match with the words in document. This can result in inaccurate documents being returned for a input query. To resolve this, work has been done to incorporate synonyms of words in to picture. By replacing the words in the question with the synonyms, the new formed input query gives better results. By the use of Word Embeddings models namely FastText and WordVec, synonyms are extracted for a given word.

Example- What is the density of GBFS cement?

Paragraph- Flyash, or Ground Granulated Blast Furnace Slag (GBFS) has density of 1321kg/m^3, 25-50 percent by weight of cementitious material may be used in concrete as part replacement of Ordinary Portland cement, and in such case, the Ordinary Portland cement content shall not be less than 100 kg/m^ of concrete.

GBFS and Flyash are synonyms in the domain data. And when Flyash is replaced with GBFS , the probability of finding the right document which contains the answer increases. To let noise not affect the results accuracy, work has been done on how to reduce the weight of a term [ki] in the question query , if it is contributing to irrelevant noise. Our objective has been to look for ways to increase accuracy of information retrieval system and improve the ranking of relevant documents as returned by Elastic Search.

The remainder of this paper is organized as follows: The next section talks about the Methodology incorporated in this work. Section 3 presents the underlying Results and Analysis. Finally, section 4 gives some conclusions.

II. METHODOLOGY

Question Answering system consists of three major components namely: Query/Question Processing, Information Retreival, Answer Extraction.

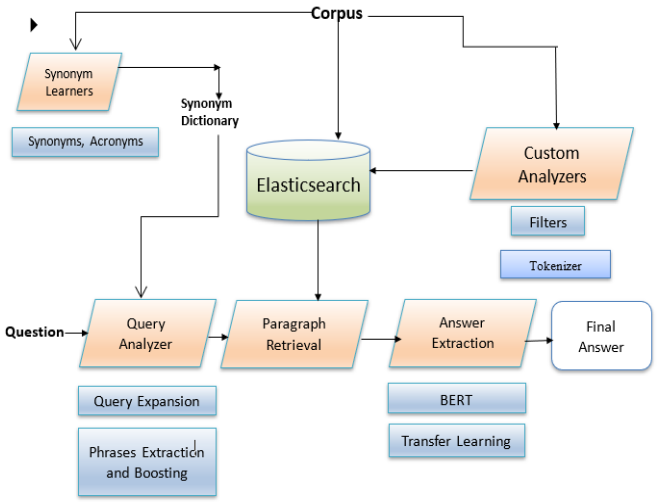

Figure 1 : Block Diagram of Question Answering Engine Architecture

2. 1 Question Processing

Question is entered which forms the query. Before passing it to the Elastic Search which acts as the Information Retrieval Engine, it undergoes some analysis. In order to achieve high accuracy for the information retrieval process it is highly important to do preprocessing on the query which can give better results

A) Query Expansion using Synonym Dictionary: In Query expansion (QE) certain terms are added to the question input query to minimize the query document mismatch and improve the retrieval performance. This work presents use of graph synonym filter of elastic search for query expansion using synonymns.

A synonym dictionary is maintained. Then using graph synonym filter, query expansion takes places. For eg ' What is density of Flyash? '

Flyash has synonym - GBFS[Granulated blast furnace] . So if question asked is 'What is dry density of GBFS?' Then the system should accordingly be able to expand the query[add up synonyms in the query] , perform boosting in a way that the terms added upon expansion argiven lesser boost and then accordingly fetch answer.

Instead of synonym dictionary, word embeddings can also be used. Word embeddings were trained for the document data using Fast Text and Word2Vec. These help in finding similar words. Synonyms extraction methods are presented as below:

1) Training Word Embedings to extract synonyms: Word2Vec and FastText models were used to train word embeddings for a corpus which was the textual data of a construction company. This helped in finding similar words . A word embeddding is a representation that is learned from training the models on huge corpus and it can be used to find words which have similar meanings and similar representation.

There are many parameters which have to be taken care of while training Word2Vec and FastText models.

a) Size: This represents the number of dimensions of the word embeddings. It is the length of the vector which represents every token(word). The default value is 100.
b) Window:This is the maximum distance between a given target word and the words around the target word. The default value is 5.
c) Min_count: The minimum count of words that are considered when training the model. So the words which have an occurrence count lesser than this count would be ignored. The default value is 5.

2. 2. Retrieving Candidate Answer Paragraphs

This is the major step for a question answering system which involves finding the correct paragraph from the textual information which will contain the answer to the input question query asked. Elastic Search was used to help extract relevant paragraphs. Initially Tesseract OCR was used to extract information from pdf documents. The paragraphs

were normalized to a length and indexed into elastic search after application of customized analyzer which had the standard tokenizer and stopwords, stemmer, lowercase and graph synonym filter. They were bulk indexed as JSON serialized format. This made up the information retrieval part where for a input question query , top 10 most relevant paragraphs were retrieved by Elastic Search.

Elasticsearch is an open source, document-based search tool which provides fast search capabilities.

Elastic Search has API's which were used. We used create, search, update, etc API's

It is document oriented, which means it stores entire object or documents. It will store and index the content of each document in order to make it searchable. Elastic Search helps in indexing, searching, sorting and filtering docuemnts. Elasticsearch uses JSON as the serialisation format for the documents. The paragraphs were normalised and indexed into Elastic Search as documents. Elastic Search uses RESTFUL API for searching, indexing, sorting, filtering documents. Its operations can be accessed over HTTP using the RestFul API and this makes it easier to be integrated with any application. Normalising the length of paragraphs- Normalisation of the paragraphs was done and then they were bulk indexed into Elastic Search using BULK API . We had used generators for bulk indexing.

Use of custom based analyzers: Analyzer is a combination of filter and tokenizers. Standard tokenizer was used. Lowercase, English stemmer and stopwords filter. Graph synonym filter was also used for incorporating synonyms


```
  "hits" : {
    "total" : {
      "value" : 1392,
      "relation" : "eq"
    },
    "max_score" : 20.423903,
    "hits" : [
      {
        "_index" : "lnt_qa_final8",
        "_type" : "_doc",
        "_id" : "IRC/12/4",
        "_score" : 20.423903,
        "_source" : {
          "text" : """a low in-situ strength and an open textured surface. Trial mixes of
              dry lean concrete shall be
prepared with water content of 5.0, 5.5, 6.0, 6.5 and 7.0 percent of the total weight of
  material.
Optimum moisture and density shall be established by preparing cubes with varying moistur
contents, and moisture-density curve shall be drawn. Special vibratory hammer shall be us
for compacting the specimens. While laying sub-base in main carriageway; the DLC may"""
        }
      },
      {
        "_index" : "lnt_qa_final8",
        "_type" : "_doc",
        "_id" : "Morth/439/5",
        "_score" : 16.97214,
        "_source" : {
          "text" : """9$03.5.4 Dry Lean Concrete Sub-base
903,5.1.1 Sampling and Testing of Cubes
Samples of dry lean concrete for making cubes shall be taken from the uncompacted materic
from different locations immediately before compaction at the rate of 3 samples for each
1000 sq.m or part thereof laid each day. The sampling of mix shall be done from the pavir
  site.
429
Section 900 Quality Control for Road We:
Test cubes of 150 mm size shall be made immediately from each mix sample.
```


Figure 2 : Top n returned results by Elastic Search

Elastic Search uses BM25 Scoring Algorithm to score each document during the extraction process for a input query.

Score(ith Document) =

$$\sum_{i}^{n} IDF(q_i) \frac{f(q_i, D) * (k1+1)}{f(q_i, D) + k1 * (1 - b + b * \frac{fieldLen}{avgFieldLen})}$$

This formula is used to calculate score for every document whenever a search query is entered. The documents with top 10 confidence score values are returned.

q_i is the i^{th} query term: For example, if I search for "Mrunal, " there's only 1 query term, so q_0 is "Mrunal".

IDF(g_i) is the inverse document frequency of the i^{th} query term. The IDF calculates how often a term occurs in all of the documents and "penalizes" terms that are common. The formula BM25 uses for IDF is:

$$ln\left(1 + \frac{(docCount - f(q_i) + 0.5)}{f(q_i) + 0.5}\right)$$

here docCount is the total number of documents that have a value for the field in the shard and $f(q_i)$ is the number of documents which contain the i^{th} query term.

fieldLen/avgFieldLen : In the BM25 algorithm it is seen that the length of the field is divided by the average field length in the denominator **fieldLen/avgFieldLen**. This tells us as to how long a given document is relative to the average document legth. If a given document is longer than the average length then the denominator will get bigger and in turn decrease the score and if it is shorter than the average length then the denominator will get smaller and increase the score. So more terms in the document which are the ones not matching the query then automatically the score for the document will be on the lower side.

b: Then there is a variable **b** which is present in the denominator and that it's multiplied by the ratio of the field length. If b is more, then the effects of the length of the document compared to the average length are more amplified . In Elastic Search default value of b is 0. 75.

k1 and f(qi, D) : There are two more parameters used to calculate score which are **k1 and f(qi, D)** . **f(qi, D)** tells the frequency of the ith query term for a given document D. So if more number of query terms occur in a particular document then its score will be higher. The way to think about **f(qi, D)** is that the more times the query terms will occur in a document, the higher its score will be . **k1** is a variable which helps determine term frequency saturation characteristics. That is, it puts a limit on how much a single query term can have its effect on the score of a given document. The default value of k1 is 0. 7.

Initially we used the default values of b and k1. Variations in b and k1 resulted to variations in the accuracy of search results. Thus we tried out different variations of b and k1 and then plotted graphs of b vs accuracy and k1 vs accuracy to arrive at values of b and k1 which were giving us better accuracy. This was done to increase search relevancy of elastic search.

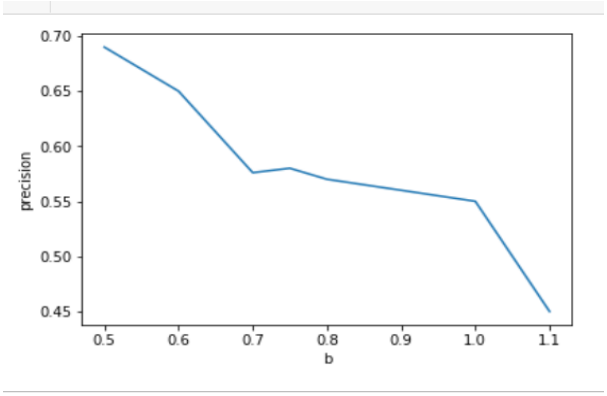

Figure 3 : Plot figure of b vs Precision of Elastic Search Results

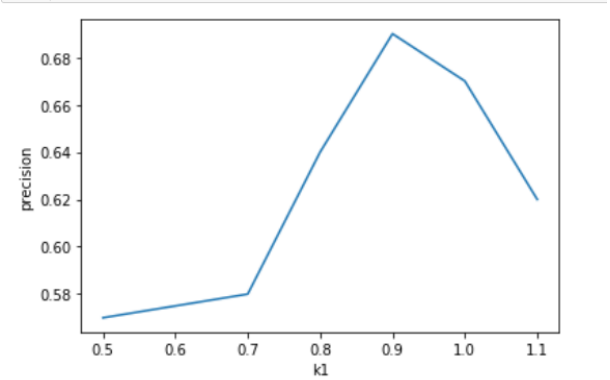

Figure 4: Plot of k1 vs Precision

The precision of first 10 top ranked documents in taken into consideration. The goal is to look at how precise or relevant are the top 10 and top 5 returned results for an individual query i.
Precision-Total number of documents retrieved that are relevant / Total number of documents retrieved.

So, Precision of Search Results (Top 10) for query input is :

$$\sum_{i=1}^{10} \text{Total number of documents retrieved that are relevant / Total number of documents retrieved}$$

Initially with default value of b and k1 the precision was 0. 58 but on changing the value of b and k1 the precision saw a change. The search relevancy increased when we both decreased k1 and b values. We plot a graph to understand the relationship and try to find a optimal range of b and k1 which can increase the search relevancy for the question answering System. On doing that it was observed that a value of 0. 9 for k1 and a value of 0. 5 for b gave accuracy of 69% for the search results .

After having achieved precision of 69% it was essential to increase the precision further to make the Question Answering Retrieval System efficient and robust.

Some methods that were researched and worked on to increase relevancy of the information retrieval system were as below:

a) Chunking - Using Spacy in NLP , performing chunking and extracting noun and verb phrases.

Then accordingly boosting the noun phrases while searching with the question query. Performing this increased the accuracy to 72 % from 69%.

b) Acronym Expansion- We performed acronym expansion by maintaining a dictionary. Acronyms like EI, FI, GSB, DLC, etc were replaced by there full forms in order to increase accuracy.

2. 3 . Answer Extraction

Once the candidate paragraphs are extracted they are passed through the developed QA system which finds the right answer from the paragraph in order to answer the question. We have used Bidirectional BERT Model for this.

BERT Model is finetuned and trained on SQUAD Question answering dataset. For a given question and a paragraph of text which contains the answer, BERT extracts the span of text which is the correct answer. Once the paragraph and question is given as input to BERT model, the question and paragraph are packed together as a single packed unit (question with A segmentation embeddings and paragraph with B segmentation embeddings). The two pieces of text are separated by the special [SEP] token. BERT also uses “Segment Embeddings” to differentiate the question from the paragraph text. These are the two embeddings (for segments “A” and “B”) that BERT learned, and which it adds to the token embeddings before feeding them into the input layer.

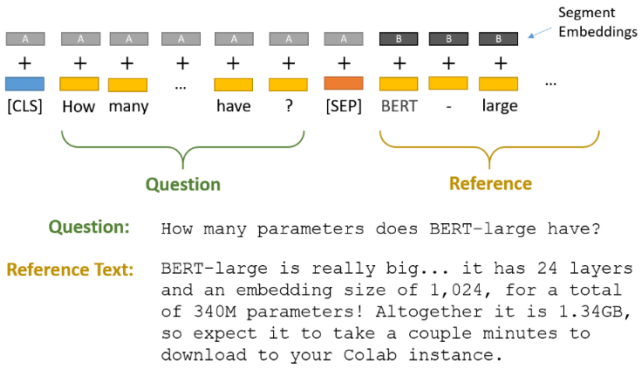

Figure 5 : Use of segment embeddings to differentiate question from text.

Finding answer: BERT predicts the span of text which contains the answer by predicting start token that marks the start of answer, and the end token which marks the end of answer. For every token present in the text, we feed the final embedding into start token classifier.

The start token classifer has single set of its weights which it goes on to apply to every word. After that dot product between output embeddings and start weights is taken , and it is applied to the softmax activation to produce a probability distribution over all of the words. Whichever word has the highest probability of being the start token is the one that we

pick. We repeat this process for the end token–we have a separate weight vector this. It then picks the tokens with highest start and end scores. This way the answer is predicted. If the question has no answer, we will simply predict both the start and end positions as 0.

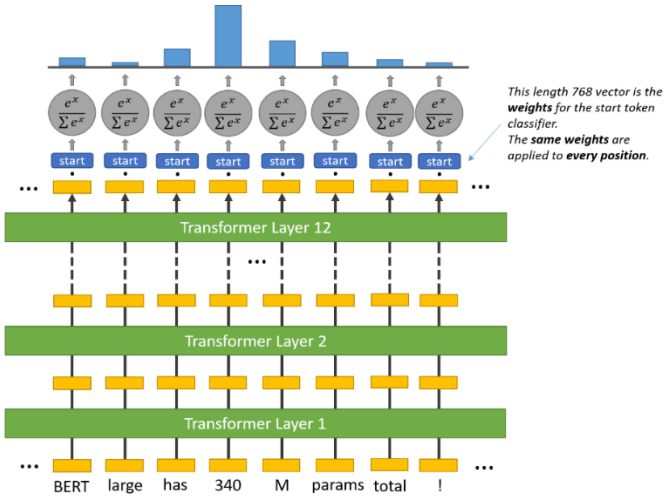

Figure 6 : Transformer layer Architecture used in BERT

III. EXPERIMENTAL ANALYSIS AND RESULTS

Formation of Dataset- BERT was trained on SQUAD dataset.

SQUAD - Stanford Question Answering Dataset (SQuAD) is dataset, which consists of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a span of text, from a paragraph , or the question could be unanswerable. We obtained the SQUAD 2. 0 dataset and then trained BERT mini model on it.

Bert Mini has 4 layers and embedding size of 256. The finetuned BERT was then used to obtain prediction for the domain data which consisted of the construction company textual data and a list of questions. The paragraphs obtained from Elastic Search along with the input query questions were sent to BERT mini trained model and then predictions for answer were obtained. BERT mini gave F1 score of 65. 8% when tested for questions set.

BERT-large is really big. It has 24-layers and an embedding size of 1, 024, for a total of 340M parameters! If the system has higher configuration [GPU] then BERT large can be used.

BERT large gave F1-score of 91. 4 %.

Hyperparameters obtained on fine tuning on SQUAD dataset with BERT LARGE –

Train batch size=24

Learning Rate=3e-5

Number of train epochs=2. 0

Maximum sequence legth=384

Doc Stride=128

The accuracy of the search engine is combined accuracy of Elastic Search and BERT accuracy. Methods discussed in section 2. 2 were employed to increase accuracy of Elastic Search too. Final accuracy obtained for Elastic Search was 72%. Initially it was just 58%. As discussed above several methods were employed to increase precision of Elastic Search Results too.

Development of Search User Interface for Question Answering System : This user interface had used FLASK as backend framework and HTML and CSS as frontend.

It took a user query and displayed the top 5 results for that question. It also displayed pdf name, page no , paragraph no where the answer lies for that question.

Improving response time – When question query was entered, initially it took a lot of time to render results. So some methods were employed to increase response time.

Batch prediction instead of multiple predictions per question.

Below are the accuracy results for BERT Models according to the layers. The model was trained on SQUAD Dataset and then tested for predictions on Domain Dataset.

Table 1. Accuracy results for BERT Model

Model	Layers	F1 Score
Bert Tiny	2	64.2%
Bert Mini	4	65.8%
Bert	8	73.5%
Medium Bert Large	24	91.4%

Table 2. values of b, k and precision

b	K1	Elastic Search Precision
0.75	0.7	58%
0.7	0.6	63%
0.5	0.9	69%

IV. CONCLUSION

This work aims in developing intelligent question answering engine for industry stakeholders. It would reduce the manual effort which is needed to scan thousands of documents in order to look up for paragraphs, documents and answers. This work discusses methods by which the relevancy of the search results for information retrieval can be increased. In its future scope it would have voice enabled bot with it which would make it into a conversational question answering bot. This system would be highly helpful for domain like healthcare, finance and other firms too.

V. REFERENCES

[1]. Said Alami Aroussi, Nfaoui El Habib, Omar El Beqqali, “Improving question answering systems by using the explicit semantic analysis method, 11th International Conference on Intelligent Systems: Theories and Applications (SITA)”,2016.

[2]. Evgeniy Gabrilovich and Shaul Markovitch, “Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis” : Proceedings of The 20th International Joint Conference on Artificial Intelligence (IJCAI), Hyderabad, India, January 2007

[3]. Tellez-Valero, Alberto., Montes-y-Gomez, Manuel., Villasenor-Pineda, Luis. and Padilla Anselmo Penas , “Learning to select the correct answer in multi-stream question answering. Journal of Information Processing and Management, Elsevier”,2010.

[4]. Roussinov, D., Chau, M., Filatova, E., & Robles-Flores, “Building on redundancy: Factoid question answering, robust retrieval and the other. In Proceedings of the thirteenth text retrieval conference” ,2005, pp. 1518.

[5]. Anietie Andy ,Mugizi ,Robert Rwebangira ,Mohamed Chouikha, “Exploiting Synonyms to Improve Question and Answering Systems” ,International Journal of Computer Applications (0975 - 8887) Volume 108 - No 18, December 2014

[6]. Xiao Huang, Jingyuan Zhang, Dingcheng Li, Ping Li. “Knowledge Graph Embedding Based Question Answering”, Twelfth ACM International Conference on Web Search and Data Mining (WSDM ’19), February 11-15, 2019, Melbourne, VIC, Australia. ACM.

[7]. Zhang, Wen., Yoshida,Taketoshi., and Tang Xijin, “TFIDF, LSI and Multi-word in Information Retrieval and Text Categorization”, IEEE, International Conference on Systems, Man and Cybernetics,2008, 1-4244-2384-2/08.

Cite this article as :

Mrunal Malekar, "Deep Learning Based Question Answering Search Engine", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 7 Issue 3, pp. 25-32, May-June 2021. Available at

doi : https://doi.org/10.32628/CSEIT2172139

Journal URL : https://ijsrcseit.com/CSEIT2172139