Search Engine for QnA using Distributed Inverted Index System

Snehasis Dey
Electronics & Telecommunication Engineering
College of Engineering
Bhubaneswar, India
snehasis9dey@gmail.com

Bhimasen Moharana
Department of Computer Science & Engineering
Lovely Professional University
Punjab, India
bhimasen.moharana@gmail.com

Utpal Chandra De
School of Computer Applications
Kalinga Institute of Industrial Technology
Bhubaneswar, India
deutpal@gmail.com

Tapaswini Samant
School of Electronics Engineering
Kalinga Institute of Industrial Technology
Bhubaneswar, India
tsamantfet@kiit.ac.in

Trupti Mayee Behera
School of Electronics Engineering
Kalinga Institute of Industrial Technology
Bhubaneswar, India
truptifet@kiit.ac.in

Shobhan Banerjee
Department of Engineering Technology
Birla Institute of Technology and Science
Pilani, India
shobhanbanerjee3@gmail.com

2024 3rd International Conference for Innovation in Technology (INOCON) | 979-8-3503-8193-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/INOCON60754.2024.10511792

***Abstract*—The internet along with its global uses has also evolved as a platform where various people can pose their questions on certain sites and get them answered by others from the same domain. Certain sites like Quora, Reddit, Yahoo Answers, etc. are sites enabling people to enhance their range over the net to find solutions to various issues or answers to various questions. But this leads to redundancy in data & hence increased memory consumption where different users can pose the same question in different ways. The same thing stands for the answers posted by different users for the same question in various ways. In this paper, we have worked on a search engine system for Question & Answers where given a question, we can find similar questions with the same semantic meaning using Elastic Search which uses an inverted index system & looks at various aspects of the formulated problem statement.**

Keywords—Search Engine, Stack Exchange, Inverted Index, Elastic Search, Text Similarity

I. INTRODUCTION

Almost every day we seek solutions to various issues or questions over the net in our day-to-day life. Various users may have the same question which they pose in their own unique way. Similarly, various users answer those questions in their own unique way. This creates a lot of redundancy and hence memory occupancy for various questions of the same semantic meaning. Answers are subjective and many users can offer their own opinion or solution in various ways, which cannot be neglected. But the questions with the same semantic meaning need to be identified & regarded as one.

Authors in [1] have developed an automatic QnA generation system for pre & post-operative education of patients dividing the system into three modules – text generation, answer extraction & BART-based question generation. In [2], have studied the application of a similarity algorithm to design an intelligent QnA system using WordNet semantic dictionary. A Question Driven Multiple Attention (QDMA) model is proposed in [3] that reduces the redundant & inaccurate information to focus effectively on the targets for Visual QnA. An interactive & simple question-answering system using NER & BERT models has been proposed by the authors in [4]. The use of framing theory has been proposed in [5] for community QnA to understand the influence of expressions on responses.

An enhanced hybrid approach has been proposed in [6] to build a QnA prototype using three modules – question processing, information retrieval & answer processing. Authors in [7] have proposed a contrastive semantic similarity learning (CSSL) method for multi-hop QnA over event-centric knowledge graphs. In [8], a method to improve the accuracy of extraction of appropriate QnA pairs has been proposed by the authors using GQ with negative sampling. A BERT-based hybrid QnA matching model has been proposed in [9]. Authors in [10] have created an automatic peer-reviewing system using computational linguistics for the evaluation of research manuscripts.

In this paper, we have addressed the task of finding similar questions, given a question. We have used data from Stack Overflow. There are basically two approaches to solve this – the first is using an ML approach that utilizes various techniques like Doc2Vec, BERT models, etc. & the second is using NLP techniques where a data structure called an inverted index can be used. We have implemented the latter using Elastic Search. The ordering of similar questions has been performed through a scoring function implemented using TF-IDF. Finally, a comparison has been made between simple keyword-based search and that integrated with semantic similarity search.

II. PROBLEM FORMULATION

A. Objective

The objective is to find similar questions, given a question. Each question can have many answers from different people. In addition to this, the results need to be ordered by relevance i.e., we want an ordered list of questions. The solution must be fast i.e., < 500 ms, with high precision & recall and low computational cost. We intend to build a model that is quick to deploy. Suppose we have n number of questions (Q), from which we chose the i^{th} question (Q_i). There might be k answers corresponding to this i^{th} question, i.e., $\{A_1^i, A_2^i, \ldots, A_k^i\}$.

B. Dataset

The data used by us has been taken from the Stack Overflow dataset from Kaggle which is derived from a dump of posts in June 2016. Since the data is quite huge (≈ 3.3 GB) in size, we didn't use 'pandas' as it loads all the data into the RAM at once. We used the 'csv' module instead. We have two separate files for Questions & Answers which consist of 1264216 rows of questions and 2014516 rows of answers individually. A question is uniquely recognized by the Question ID and the Owner User ID. We used the titles corresponding to each question as a basis to carry out our work. For the approach taken by us, we don't need to consider the answers file. Also, we just considered 200k data to train our model due to a lack of adequate computation power. The word cloud after the removal of stop words, for these 200k titles, is shown in the figure below, where the size of the words is directly proportional to their frequency in our corpus.

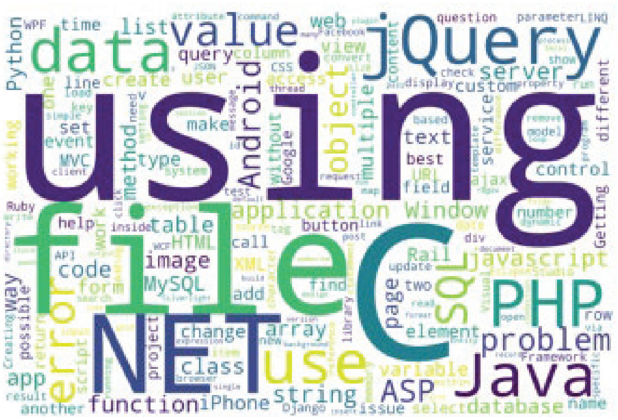


Fig. 1. Word cloud for 200k titles

From the figure above, we can see that using, file, C, NET, jQuery, etc. are some of the frequently used words in the corpus. Hence a general notion can be derived as to what is being searched for by most of the users.

III. High-Level Design

For a given question Q & answer A, the usable available attributes are as follows:

$$Q_i – \{ID, Title, Body, \ldots\}$$

$$A_j – \{ID, Body, ParentID, \ldots\}$$

The simplest design choice may be as follows:

$$D_1 – \{Q_i\text{'s Title}\}$$

$$D_2 – \{Q_i\text{'s Title} + Q_i\text{'s Body}\}$$

$$D_3 – \{Q_i\text{'s Title} + Q_i\text{'s Body} + A_j\text{'s Body}\}$$

But in our case, since we are only concerned about questions, we can use D_1 i.e., the title of i^{th} question - Q_i^{Title}. Now this consists of individual words that make up the whole title as shown below:

$$Q_i^{Title}: W_i^1\ W_i^2\ W_i^3 \ldots W_i^d$$

where d is the number of words in the document.

One of the search techniques available is hashing, which has a time complexity of O(1). In Python, these are implemented using dictionaries where against each key we get to have a value. But standard hashing techniques didn't work quite well for this.

Keyword-based search techniques prove to be inefficient because we also need to consider the semantic similarity of the sentences. For instances,

Q_i: Install Elastic Search

Q*: Setup Lucene

Now in the above two questions, the two keywords - install and setup semantically mean the same. Also, Elastic Search is built on top of Lucene. Hence through keyword-based search, Q* won't return Q_i. A software engineering approach to address this can be to set up a set of synonyms using a dictionary (thesaurus).

Another way that can be thought of is using sentence vectors to calculate semantic similarity. There are various ways of embedding these sentences into a vector such as SentenceBERT, Doc2Vec, etc. We used Universal Sentence Encoder from TensorFlow Hub to create 512-D vectors & used cosine similarity as a similarity measure to calculate the similarity between them.

The issue with these ML-based approaches is that the systems generally fall short of memory as well as computation power when dealing with millions of data points. Hence, the indexing is maintained using a specialized data structure called the inverted index.

IV. Inverted-Index

This is a data structure where instead of maintaining the storing of the vectors as mentioned above, we create dictionaries of the terms followed by storing the questions as values in which they occur. Consider the question in consideration below:

$$Q_c – \text{"pip of python"}$$

In an inverted index representation, each word of Q_c is a key, and corresponding to which the values are a list of questions (also referred to as postings) as shown below.

TABLE I. Key-Value pairs for Q_c

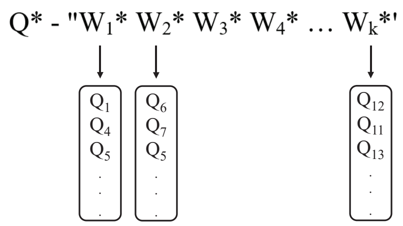
Key	**Value**
pip	Q_1, Q_4, Q_{46}, Q_{32}, Q_{61}
of	Q_1, Q_2, Q_3, Q_4, …, Q_{68}
python	Q_1, Q_4, Q_{32}, Q_{61}

In terms of Information Retrieval, the title of each question is a document & length of the postings list corresponds to the number of documents containing the term, i.e., 'pip' occurs in 5, 'of' occurs in 68 & 'python' occurs in 4 documents.

Now for instance, if someone asks a question Q* where,

$$Q^* - \text{"}W_1^*\ W_2^*\ W_3^*\ W_4^* \ldots W_k^*\text{"}$$

We create a set S of all the potentially relevant documents by performing a union of all Q_ts as shown along with the inverted index representation below.



$$S = \{Q_1, Q_4, Q_5, \ldots, Q_{13}\}$$

Fig. 2. Q* and set of postings (S)

Having known the concept of inverted indexing, below are a few issues & their solutions that arise.

A. How do we order them?

The ordering of similar questions is done based on a scoring function. In our case, we shall use the Term Frequency-Inverse Document Frequency (TF-IDF) as a scoring function, which is defined as follows:

$$\text{TF-IDF} = \text{TF}(t, d) \times \text{IDF}(t)$$

Where TF is the number of times t occurs in document d & IDF is defined as:

$$\text{IDF} = \log\left(\frac{1+n}{1+DF(d,t)}\right)$$

In the equation above, DF is the document frequency of t & n is the total number of terms in the document. The higher the TF-IDF value for a document, the more its relevance. Irrelevant documents ideally have a TF-IDF value of 0.

In our context, this takes the form of

$$\text{TF-IDF}(Q^*, Q_i)\ \forall\ Q_i \in S$$

When it comes to scoring, there's a common misconception of removing the stop words to acquire more relevant results. But while creating sentence vectors this might create a problem where the whole sequence of words is converted into a vector. But TF-IDF takes care of it automatically.

B. Load Balancing

When we have too many questions to consider for our task, it is not practical to serially process all of them.

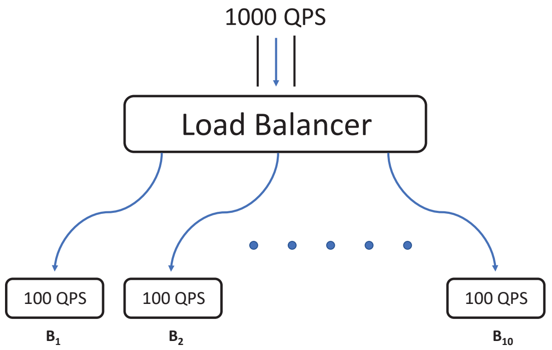


Fig. 3. Load Balancer

We need a balancer where the questions are processed in batches. In [11], we have a multithreading approach where the data has been processed parallelly using all the cores available in the CPU. But since we are reading the data through the 'csv' module in RAM as mentioned above, we tweak the balancer for processing as shown in Figure 3. We can see that the task of processing 1000 questions per second (QPS) can be segregated into 10 batches of 100 questions each, hence processing 1000 QPS parallelly. This reduces the computation time and memory space consumption during processing.

V. Implementation

We use an Apache Lucene-based search engine called Elastic Search that provides a distributed search and analytics engine for almost all varieties of data & uses inverted indices by default. An index in elastic search is equivalent to a database in Relational Database Management Systems. We read each question and index them. For our low latency requirement, we indexed only the titles, but ideally, the body should also be included in the indexing and then the index must be defined.

A schematic representation of the requirement is shown in the figure below.

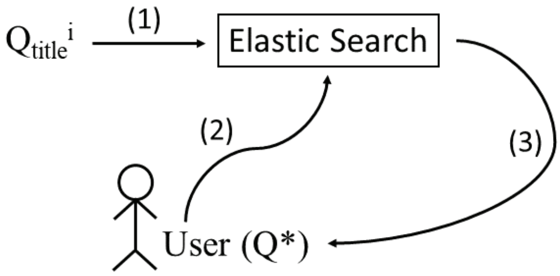


Fig. 4. Schematic Flow Structure of the Process

Step (1) is the ingestion task where all the question titles are indexed to the Elastic Search engine, followed by a search question (Q*) by a user in step (2). There might be misspelled words, for which we can run a spell check and proceed ahead. The final step (3) includes sorted results being returned to the user.

The granular details of the process above are shown in Figure 5.

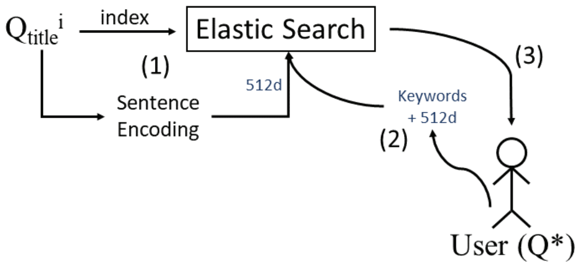


Fig. 5. Granular details of the Flow Structure

Like Figure 4, the three steps shown above include:

(1) Ingestion

(2) Search query

(3) Ranked Results

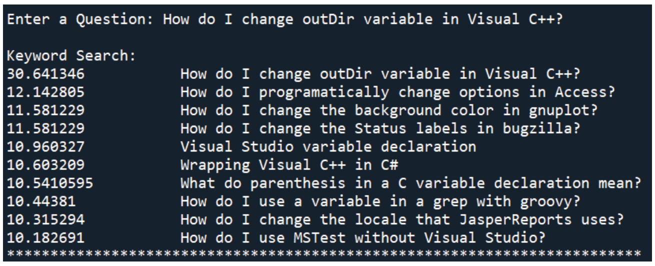
We used Universal Sentence Encoder (USE-4) to convert the question into a vector of 512 dimensions, based on which the ES engine acquires data from steps (1) and (2) to process the ranked results in step (3).

VI. Results & Discussion

After the question/query (Q*) has been entered by the user, the glimpses of results are shown below. In our case,

Q*: How do I change OutDir variable in Visual C++?

The results for a simple keywords-based search are given below.

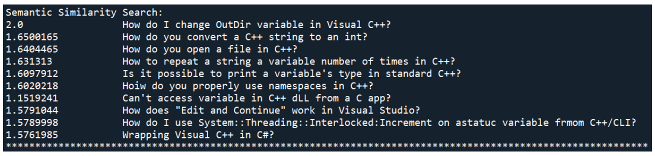


```
Enter a Question: How do I change outDir variable in Visual C++?

Keyword Search:
30.641346         How do I change outDir variable in Visual C++?
12.142805         How do I programatically change options in Access?
11.581229         How do I change the background color in gnuplot?
11.581229         How do I change the Status labels in bugzilla?
10.960327         Visual Studio variable declaration
10.603209         Wrapping Visual C++ in C#
10.5410595        What do parenthesis in a C variable declaration mean?
10.44381          How do I use a variable in a grep with groovy?
10.315294         How do I change the locale that JasperReports uses?
10.182691         How do I use MSTest without Visual Studio?
*************************************************************************
```

Fig. 6. Ranked results for keyword-based search

The figure below shows us results for the same Q* but with a semantic similarity search.



```
Semantic Similarity Search:
2.0                How do I change OutDir variable in Visual C++?
1.6500165          How do you convert a C++ string to an int?
1.6404465          How do you open a file in C++?
1.631313           How to repeat a string a variable number of times in C++?
1.6097912          Is it possible to print a variable's type in standard C++?
1.6020218          Hoiw do you properly use namespaces in C++?
1.1519241          Can't access variable in C++ dLL from a C app?
1.5791044          How does "Edit and Continue" work in Visual Studio?
1.5789998          How do I use System::Threading::Interlocked:Increment on astatuc variable frmom C++/CLI?
1.5761985          Wrapping Visual C++ in C#?
*******************************************************************************************************
```

Fig. 7. Ranked results for semantic-based search

The cosine similarities have been used to fetch similar questions and their ranking is based on TF-IDF scores as discussed in Section IV. We can see that the cosine distances for the results given by semantic-based search are less than those given by the keywords-based search technique. This means that the similarity of the semantics-based search technique is greater than that of the simple keyword-based search technique.

VII. Conclusion & Future Scope

The semantic search execution was carried out using Elastic Search which suggested similar question titles based on our query title. Based on the acquired results, we can conclude that the search based on semantics is more efficient compared to simple keywords-based search. The same concept can be extended and used for questions from other platforms such as Quora. Ambiguity issues related to the context of the text are addressed well by BERT models and may be integrated with the pre-existing models. Also, for sarcastic texts present in the questions, we might need separate models to first separate a sarcastic question from a normal-toned one & then work with them.

Also, there are other ranking metrics such as Normalised Discounted Cumulative Gain (NDCG) that can be considered. Also, for questions including equations such as $y = mx + c$, we need to leverage techniques such as tokenization or n-gram-based indexing. Other libraries such as Facebook AI Similarity Search (FAISS) can also be used to implement our task at hand, but that needs a GPU-based system to run.

This model in real-time can be deployed by creating a flask API. We can perform visualization using Prometheus or Kibana. Also, memory consumption for the whole process can be studied and analyzed using Grafana.

References

[1] Y. Ou, S. Chuang, W. Wang and J. Wang, "Automatic Multimedia-based Question-Answer Pairs Generation in Computer Assisted Healthy Education System," 2022 10th International Conference on Orange Technology (ICOT), Shanghai, China, 2022, pp. 01-04, doi: 10.1109/ICOT56925.2022.10008119.

[2] J. Zhang, "Application Research of Similarity Algorithm in the Design of English Intelligent Question Answering System," 2022 IEEE 2nd International Conference on Mobile Networks and Wireless Communications (ICMNWC), Tumkur, Karnataka, India, 2022, pp. 1-4, doi: 10.1109/ICMNWC56175.2022.10031708.

[3] J. Wu, F. Ge, P. Shu, L. Ma and Y. Hao, "Question-Driven Multiple Attention(DQMA) Model for Visual Question Answer," 2022 International Conference on Artificial Intelligence and Computer Information Technology (AICIT), Yichang, China, 2022, pp. 1-4, doi: 10.1109/AICIT55386.2022.9930294.

[4] S. Acharya, K. Sornalakshmi, B. Paul and A. Singh, "Question Answering System using NLP and BERT," 2022 3rd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2022, pp. 925-929, doi: 10.1109/ICOSEC54921.2022.9952050.

[5] Q. Wu, C. S. Lee and D. H. -L. Goh, "Asking for Help in Community Question-Answering : The Goal-Framing Effect of Question Expression on Response Networks," 2022 ACM/IEEE Joint Conference on Digital Libraries (JCDL), Cologne, Germany, 2022, pp. 1-5.

[6] V. Bali and A. Verma, "An enhanced hybrid approach for building of Question Answering Prototype," 2021 First International Conference on Advances in Computing and Future Communication Technologies (ICACFCT), Meerut, India, 2021, pp. 85-90, doi: 10.1109/ICACFCT53978.2021.9837348.

[7] W. Tang, Q. Kong, W. Mao and X. Wu, "Contrastive Semantic Similarity Learning for Multi-Hop Question Answering over Event-Centric Knowledge Graphs," 2022 IEEE 8th International Conference on Cloud Computing and Intelligent Systems (CCIS), Chengdu, China, 2022, pp. 360-364, doi: 10.1109/CCIS57298.2022.10016335.

[8] W. Kano and K. Takeuchi, "Data Augmentation for Question Answering Using Transformer-based VAE with Negative Sampling," 2022 12th International Congress on Advanced Applied Informatics (IIAI-AAI), Kanazawa, Japan, 2022, pp. 467-470, doi: 10.1109/IIAIAAI55812.2022.00097.

[9] C. Zheng, Z. Wang and J. He, "BERT-Based Mixed Question Answering Matching Model," 2022 11th International Conference of Information and Communication Technology (ICTech)), Wuhan, China, 2022, pp. 355-358, doi: 10.1109/ICTech55460.2022.00077.

[10] T. ul Haq, D. Anand and N. Kapoor, "The manuscript evaluation through Artificial Intelligence using Natural Language Processing and Machine Learning," 2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2022, pp. 1-6, doi: 10.1109/GCAT55367.2022.9971874.

[11] S. Banerjee, B. B. Dash, M. K. Rath, T. Swain and T. Samant, "Malware Classification using Bigram BOW, Pixel Intensity Features, and Multiprocessing," 2022 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 2022, pp. 1-5, doi: 10.1109/CONECCT55679.2022.9865764.

[12] U. C. De, S. Banerjee, M. K. Rath, T. Swain and T. Samant, "Content Based Apparel Recommendation for E-Commerce Stores," 2022 3rd International Conference for Emerging Technology (INCET), Belgaum, India, 2022, pp. 1-6, doi: 10.1109/INCET54531.2022.9824870.

[13] U. C. De, B. B. Dash, T. M. Behera, T. Samant and S. Banerjee, "Text-Based Recommendation System for E-Commerce Apparel Stores," 2023 International Conference for Advancement in Technology (ICONAT), Goa, India, 2023, pp. 1-4, doi: 10.1109/ICONAT57137.2023.10080436.

[14] B. B. Dash, U. C. De, T. M. Behera, T. Samant and S. Banerjee, "Recommendation System for E-Commerce Apparel Stores based on Text-Semantics," 2023 2nd International Conference for Innovation in Technology (INOCON), Bangalore, India, 2023, pp. 1-5, doi: 10.1109/INOCON57975.2023.10101358.