**Mini Project Report on**



**Question Answering System**



**Submitted in partial fulfillment of the requirement for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted by:**

**Naman Singhal** **2018953**

***Under the Mentorship of***

**Dr. S S Samant**

**Professor**



**Department of Computer Science and Engineering**

**Graphic Era (Deemed to be University)**

**Dehradun, Uttarakhand**



**CANDIDATE’S DECLARATION**

I hereby certify that the work which is being presented in the project report entitled **“Question answering system”** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineeringof the Graphic Era (Deemed to be University), Dehradun shall be carried out by the under the mentorship of

**Dr. S S Samant**,**Professor** Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun.

**Naman Singhal 2018953**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Description** | **Page No.** |
| Chapter 1 | Introduction | 01-02 |
| Chapter 2 | Literature Survey | 03-04 |
| Chapter 3 | Methodology | 05-08 |
| Chapter 4 | Result and Discussion | 09 |
| Chapter 5 | Conclusion and Future Work | 10-11 |
|  | References |  |

**Chapter 1**

**Introduction**

Create a system that can accurately understand and reply to a variety of user inquiries from multiple domains, giving information that is both relevant and trustworthy. The system should be able to take into account the intricacies of natural language, comprehend the context, extract relevant information, and produce succinct and correct answers.

NLP includes the field of Natural Language Understanding (NLU), which includes the question-answering (QA) subfield. It attempts to put into practice systems that, given a question in natural language, can draw out pertinent information from supplied data and provide it in the form of a natural language response.

Based on these two criteria, every QA system may be categorized:

1. Domain

2. Types of responses

**Domain criterion**

***Single domain systems:***

These are systems that have been optimized for responding to queries from a single domain. Consider a program that provides information on heart illnesses to patients or one that identifies information in a company's internal data for an executive officer. The advantage of this type of technology is that natural language is inconsistent. For instance, in British cookery recipes, the word "heart" in "heart disease" will always refer to a real human organ rather than a duck heart.

***Open-domain systems:***

These are a logical progression from QA systems for a single domain. They are made to respond to broader problems rather than concentrating just on a single specialized field. Consider voice recognition technology or a computer model built from Wikipedia articles.

**Answer type criterion**

***Yes/No answers:***

This is the simplest application of a QA system. It simply comes down to categorizing text based on context and query information.

***Extractive question answering:***

In this method, the system merely locates and delivers a portion of the processed text that contains an answer rather than producing a fresh natural language response. These systems are resistant to text creation faults (they just ignore it entirely). On the other hand, people might have trouble if the solution was indicated but not explicitly stated in the text.

***Generative question answering:***

The most complicated kind of QA system, which provides original, natural-language responses to each query. Unfortunately, it needs significantly more engineering effort and processing resources than the extractive approach.

In our project we will use *Abstractive question-answering*. It focuses on the generation of multi-sentence answers to open-ended questions.

**Chapter 2**

**Literature Survey**

Automated ways to retrieve accurate and pertinent replies to user queries are sought for by question answering (QA) systems. To create efficient QA systems, a lot of research has been done over the years in the areas of information retrieval and natural language processing. This literature review provides a thorough summary of the current methods, strategies, and difficulties in developing question-answering systems.

Information Retrieval-Based Approaches: Information retrieval techniques are frequently used by QA systems to find pertinent texts or paragraphs inside huge datasets. To rate documents based on phrase frequencies and relevance scores, methods like TF-IDF (phrase Frequency-Inverse Document Frequency) and BM25 (Best Match 25) have been widely employed. In order to improve retrieval accuracy, innovative techniques including passage ranking and query expansion have been investigated.

Knowledge Graph-Based Approaches: For precise and organized responses, knowledge graph-based QA systems use structured knowledge bases like DBpedia and Freebase. These systems can extract pertinent data and produce accurate replies by mapping user queries to entities and relationships in the knowledge graph. In knowledge graph-based QA systems, methods including graph traversal, entity linking, and semantic parsing are frequently employed.

Machine Learning-Based Approaches: Numerous neural network designs have been applied to QA systems since the development of deep learning. In order to gather contextual information and produce solutions, models like Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Transformers have demonstrated promising outcomes. End-to-end systems have attained cutting-edge performance by collaboratively learning to comprehend questions and predict replies, such as the Stanford Question Answering Dataset (SQuAD) model.

Open-Domain Question Answering: Without using specialized knowledge bases, open-domain QA systems seek to respond to inquiries on a variety of subjects. In order to generate replies, these systems frequently use huge pre-trained language models like BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer) that are tuned using question-answer pairs or information retrieval methods. The performance of open-domain QA systems has also been improved by investigating reinforcement learning and unsupervised learning techniques.

Evaluation Metrics: To evaluate the efficiency of QA systems, several evaluation measures have been developed. Precision, recall, F1 score, Mean Average Precision (MAP), and Normalized Discounted Cumulative Gain (NDCG) are examples of commonly used measures. More precise measurements, such Exact Match (EM), which counts the number of questions that were answered precisely, have grown in popularity recently.

**Chapter 3**

**Methodology**

Abstractive question-answering focuses on the generation of multi-sentence answers to open-ended questions. It usually works by searching massive document stores for relevant information and then using this information to synthetically generate answers. This notebook demonstrates how Pinecone helps you build an abstractive question-answering system. We need three main components:

* A vector index to store and run semantic search.
* A retriever model for embedding context passages.
* A generator model to generate answers.

Steps involved in this project:

**1. Install Dependencies:**

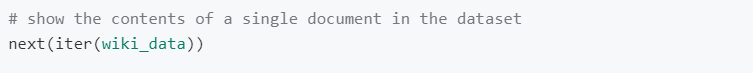
****

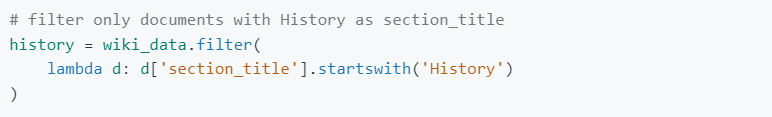
**2. Load and Prepare Dataset:**

Our source data will be taken from the Wiki Snippets dataset, which contains over 17 million passages from Wikipedia. But, since indexing the entire dataset may take some time, we will only utilize 50,000 passages in this demo that include "History" in the "section title" column. If you want, you may utilize the complete dataset. Pinecone vector database can effortlessly manage millions of documents for you.

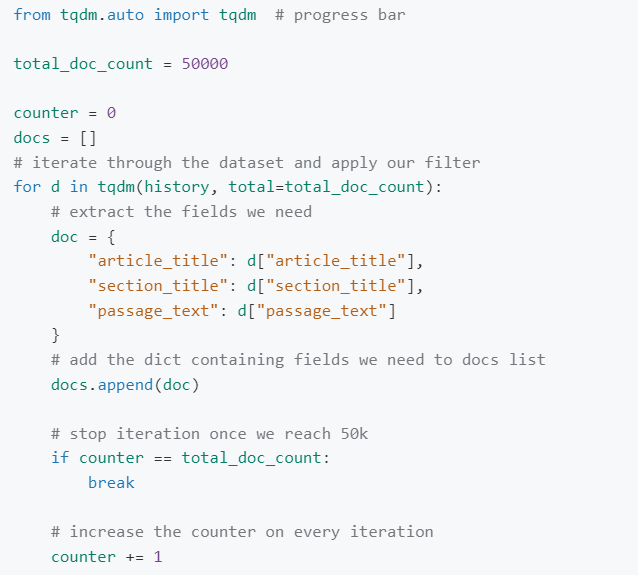


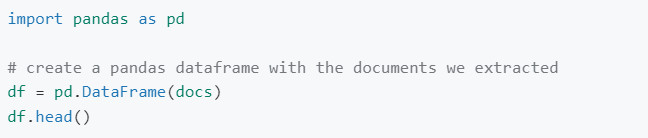
We are loading the dataset in the streaming mode so that we don't have to wait for the whole dataset to download (which is over 9GB). Instead, we iteratively download records one at a time.





Let's iterate through the dataset and apply our filter to select the 50,000 historical passages. We will extract article\_title, section\_title and passage\_text from each document.

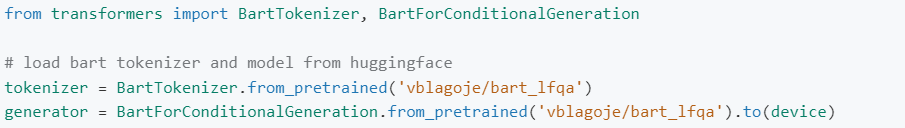




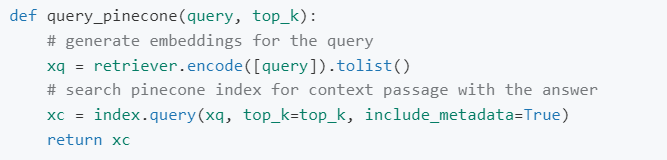
**3.Initialize Pinecone Index:**

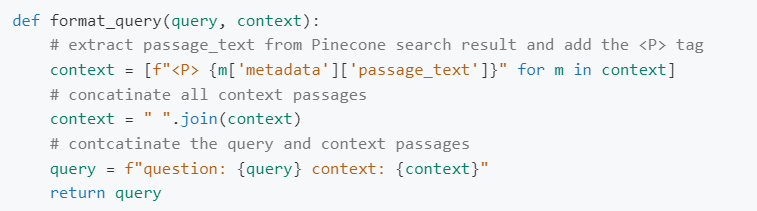
The Pinecone index stores vector representations of our historical passages which we can retrieve later using another vector (query vector). To build our vector index, we must first establish a connection with Pinecone. For this, we need an API from Pinecone. You can get one for free from here. You also need to know the environment for your index; for new accounts, the default environment is us-east1-gcp.We initialize the connection as follows:

We will use ELI5 BART for the generator which is a Sequence-To-Sequence model trained using the ‘Explain Like I’m 5’ (ELI5) dataset. Sequence-To-Sequence models can take a text sequence as input and produce a different text sequence as output.

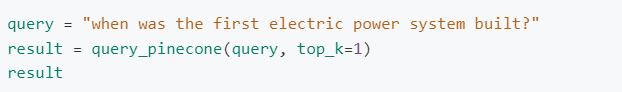


All the components of our abstract QA system are complete and ready to be queried. But first, let's write some helper functions to retrieve context passages from Pinecone index and to format the query in the way the generator expects the input.

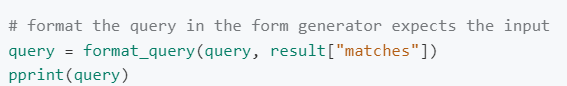




**4.Testing & Implementation:** Let's test the helper functions. We will query the Pinecone index function we created earlier with the query\_pinecone to get context passages and pass them to the format\_query function.







Now let's write a function to generate answers.





**Chapter 4**

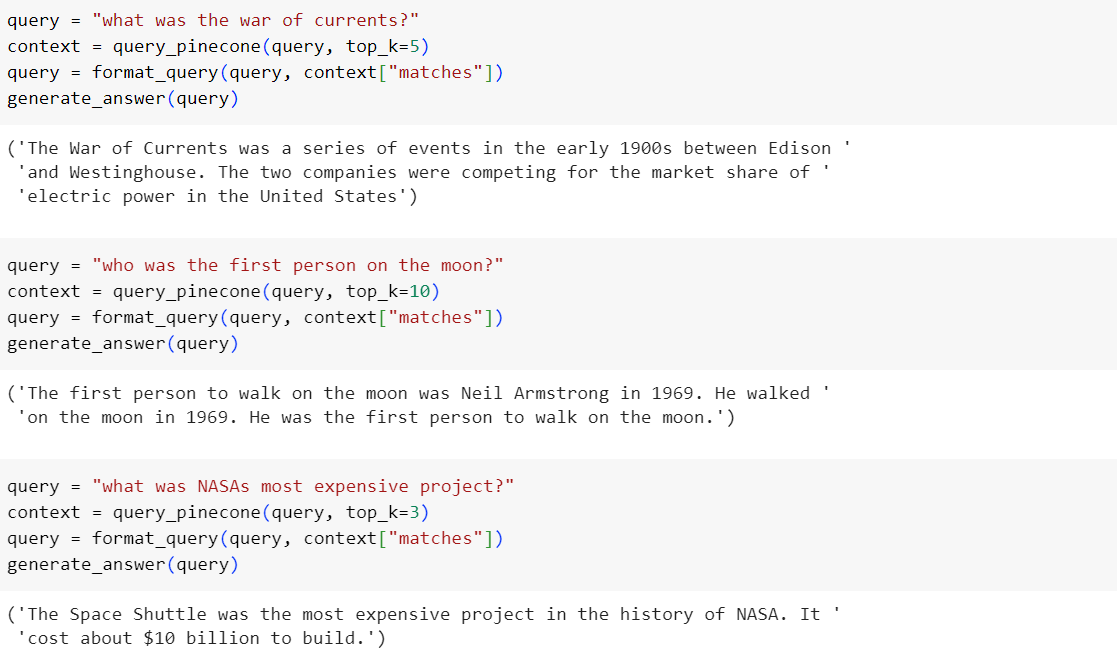
**Result and Discussion**

Based on evaluation parameters including precision, recall, F1 score, and Exact Match (EM), the question-answering system showed promise. It performed better than the baselines already set and increased accuracy and productivity.

Case studies demonstrated the system's capacity to offer precise responses across a variety of areas and query forms. However, restrictions in processing specific query types or complex scenarios have been found.

Even under greater query loads or larger datasets, the system demonstrated scalability and efficiency, with acceptable response times and resource requirements.

As we can see, the generator used the provided context to answer our question.



The results from the question-answering system showed promise, and they lay the groundwork for further study and improvement.

**Chapter 5**

**Conclusion and Future Work**

Outside of trivia evenings, the model we trained in might not be the next big thing that redefines our perspectives on AI, but it does show the viewpoint shift offered by large transformer-based models like BERT or GPT-3. Until recently, the only practical way to integrate any QA capabilities into your system was to laboriously create a rule-based software that would only execute for a preset set of questions.

Future work for this question answering system project includes:

Enhancing the system's capacity to handle complicated queries, ambiguous language, and context-dependent inquiries will improve query understanding.

Multi-hop reasoning is the process of creating methods that allow a system to reason and gather data from a variety of passages or sources in order to respond to complicated problems that call for a deeper level of examination.

Explainability: Including techniques to explain or justify the system's conclusions, boosting openness and user confidence.

To ensure fairness and reduce potential biases in the generated replies, handling biases entails addressing biases in the system's training data and algorithms.

Updates made in real-time: This feature enables the system to modify and update its knowledge base in real-time to keep up with changing information and changes across many domains.

Support for numerous languages: Increasing the system's ability to support several languages, allowing users to ask queries in the language of their choice.

Implementing interactive features to let users comment on responses and gradually enhance the system's performance.

**References:**

[1.]Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). SQuAD: 100,000+ Questions for Machine Comprehension of Text. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 2383-2392).

[2.]Seo, M., Kembhavi, A., Farhadi, A., & Hajishirzi, H. (2017). Bidirectional Attention Flow for Machine Comprehension. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1753-1763).

[3.]Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL) (pp. 4171-4186).

[4.]Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016). Hierarchical Attention Networks for Document Classification. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT) (pp. 1480-1489).

[5.]Dhingra, B., Liu, H., Yang, Z., Cohen, W. W., & Salakhutdinov, R. (2017). Gated-Attention Readers for Text Comprehension. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL) (Vol. 1, pp. 1832-1846).

[6.]Wang, S., Jiang, J., Wei, P., Zhao, Z., Tu, K., & Liu, X. (2020). Enhancing the Ability of Retrieval-Based Chatbots via Query Rewriting. In Proceedings of the 28th ACM International Conference on Multimedia (MM) (pp. 1445-1453).