



Information Retrieval and Question Answering

Group:

Ky Cuong Pham	a1906313
Robin Viltoriano	a1900159
Dang Thinh Nguyen	a1903686

The University of Adelaide

4333_COMP_SCI_7417 Applied Natural Language Processing

Lecturer: Dr. Alfred Krzywicki

Table of Contents

1. Abstract.....	2
2. Introduction	2
3. Exploratory Data Analysis	3
4. Preprocessing.....	3
4.1. Eliminating duplicate documents	3
4.2. Cleaning dataset	3
4.3. Chunking dataset.....	4
5. System Architecture	4
6. Model Selection and Training.....	5
6.1. Evaluation Matrix	5
Mean reciprocal rank (MRR)	5
ROUGE.....	5
6.2. Information Retrieval	5
6.3. Reader Algorithm.....	7
6.4. Generative Model.....	8
6.5. Whole System Evaluation	10
7. User Interaction with the System	10
8. Conclusion	11
9. References	12
Appendixes.....	13
Appendix A.....	13
Appendix B – NER model example.....	13
Appendix C – Test cases	14

1. Abstract

Recurrent neural networks (RNNs) are deep learning-based models for sequential data by maximizing the capability of learning features and long-term dependencies. After ChatGPT was released to the public in November 2022, different advanced AI Chatbots such as Gemini, Copilot, and Bard are competing with each other on publicly available knowledge for question-answering tasks. However, publicly available knowledge might be too general that in some cases, users may want to ask questions about private knowledge, which requires a question-answering system that works exclusively on a specific database. Addressing this gap, we propose a framework on how to build a Question and Answering system, given a query and a database, users can ask questions only related to a given data. As a result, that system ensures data privacy and convenience by using natural language answers to the conversation. Our system can work on local devices with limited computational availability, simple enough to be maintained by smaller businesses, without sacrificing any performance.

2. Introduction

In this project, we will build a Question-answering system using the Cross-Encoder model as the information retrieval, pre-trained Bidirectional Encoder Representation from Transformer (BERT) using the SQuAD dataset as the reader, and GPT 3.5 as the generator.

Firstly, for each question, the system will retrieve relevant articles using a retrieval model. The retrieval model will return the top 2 articles. Then, the reader model will extract the answer from the articles. Lastly, the answer that has a probability greater than 0.5 will be generated using a generative model.

Furthermore, the news anchor business is one of the industries that could benefit from this system. They can use this Question Answering system in their everyday news, so the user could just ask a question about the news and get the answer more efficiently, instead of reading the whole news. If the system is good enough, it could increase business competitiveness and user experience.

There are some limitations to the system: (1) The system will only be able to answer What, Where, When, and Who questions; (2) The system will not be able to answer Why and How questions; (3) It will also not be able to answer questions that require multiple sentences from a different paragraph to answer, since the reader only gets the snippet of the article.

3. Exploratory Data Analysis

This experiment's data consists of 1000 articles with different topics: lifestyle, business, entertainment, politics, law, art, crime, science, sports, technology, accidents, architecture, and health. The dataset has 7 columns, however, only 2 columns will be used, which are the “id” and “article”. The dataset will be split into training and testing where the first 100 articles will be the testing dataset. Furthermore, we limited the type of question in this project to Who, Where, When, and What, since How and Why questions require complex and longer answers, which the system may not be able to cover.

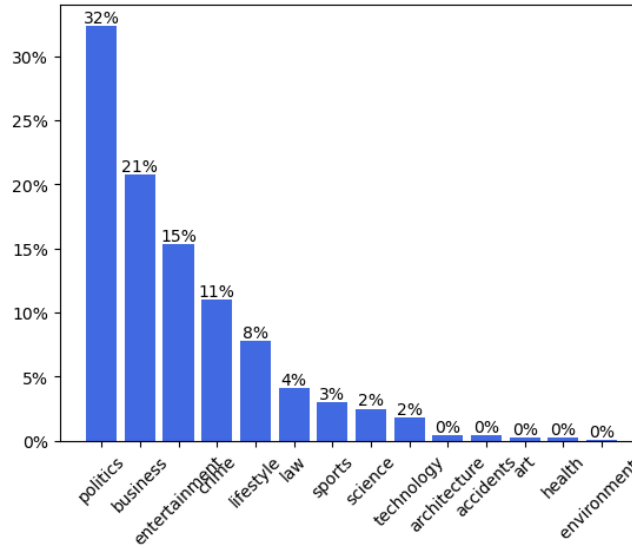


Figure 1. Topic distribution

In this dataset, the articles mainly are from politics, business, entertainment, crime, and lifestyle with 32%, 21%, 15%, 11%, and 8% respectively. Therefore, it was expected that some of the questions would have these topics. Furthermore, the average number of words per article is around 1121 with a 617 standard deviation. The minimum number of words in an article is 185 and the maximum number of words in an article is 5579.

4. Preprocessing

4.1. Eliminating duplicate documents

In the dataset, there are several duplicate documents, so we eliminated them to reduce the processing time, improving efficiency.

4.2. Cleaning dataset

Data contains non-ASCII characters such as ÿ, ®, ^a. Removing them can reduce noise in the dataset. The dataset also contains multiple question mark issues that need to be addressed: (1) a single quote

is placed by a question mark; (2) a comma followed by a question mark; (3) more consecutive question marks.

4.3. Chunking dataset

The average length, which has been mentioned in the EDA section, is greater than the sequence length limitation in the BERT model. Thus, we break the text into 100-word sequences with 50-word overlap; therefore, the chunk data is appropriate for further processing and ensures that each word is embedded in multiple contexts, capturing a wider range of semantic information.

5. System Architecture

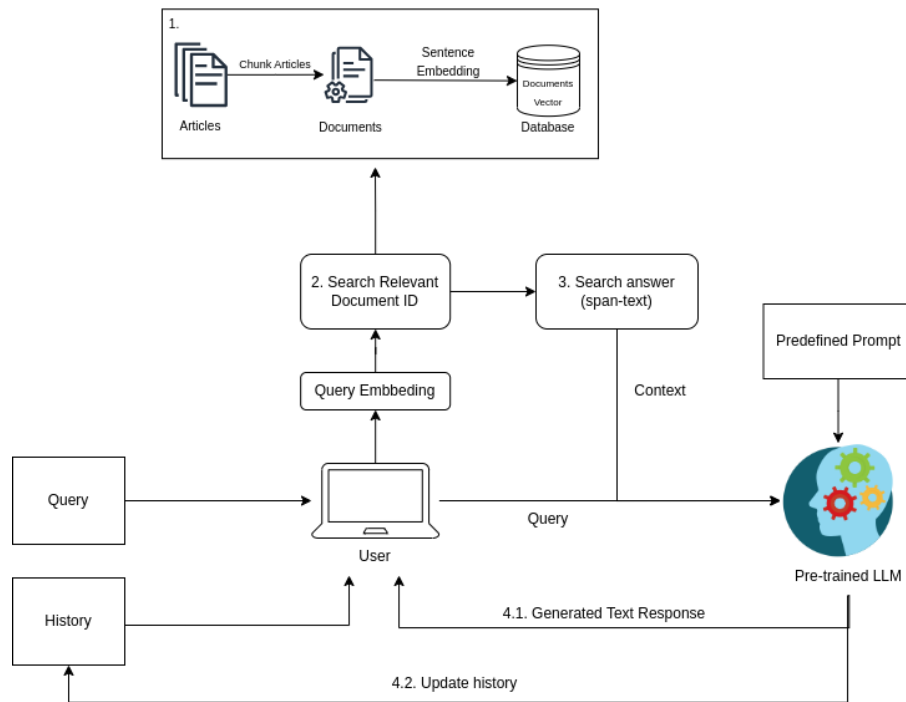


Figure 2. System Architecture

There were four main parts in the system architecture. Firstly, all of the articles will be chunked into 100 words with 50 words of intersection. Since there were approximately 1000 words for each article, the number of documents will be doubled. All of these documents will be transformed into a vector using a word embedding model, BERT, and stored in the database. Secondly, when the user inputs the query, the generative model will reformulate that query, if needed, based on conversation history, then that reformulated query will also be transformed into a vector using the same BERT model as before. The most relevant document will be searched in this step and return the document ID as well as the document passage. Moreover, given the limitations of the generative model in token acceptance, the span text of the answer will be determined using QA-BERT. This step aims to mitigate noise from the provided context, enhancing the accuracy of the response. Before the answer is used

in the next step, the surrounding words (5 words before and after the answer) were added to the context. Finally, the query and the context will be inserted into the pre-trained LLM model, thus it could respond in a natural language.

6. Model Selection and Training

6.1. Evaluation Matrix

Mean reciprocal rank (MRR)

Mean reciprocal rank is a metric used to evaluate ranked retrieval. This evaluation only returns the most relevant document d from a corpus c to a single query, which is reciprocal rank [1]. For a set of queries Q , calculate the average over these reciprocal ranks to get the MRR.

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}$$

ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a metric designed for evaluating translation and summarization [2]. This project will use ROUGE for fine-tuning **google/flan-t5-base** in Section 6.3 and evaluating the reader algorithm. There are several variants of ROUGE; however, in this project, we will mainly focus on ROUGE-1 and ROUGE-L. ROUGE was originally developed for evaluating text summarization tasks, it has been adapted and used in other NLP tasks beyond summarization, including question answering (QA). However, it's important to note that ROUGE is not specifically designed for evaluating question-answering systems, and its application to QA tasks may have limitations.

6.2. Information Retrieval

The main goal of Information Retrieval (IR) in this system is to estimate the most relevant documents to the given query. An approach to implement is word embedding with the BERT model. The query and corpus were encoded separately using the same BERT model [1].

After encoding, the embedding is stored and searched by Facebook AI Similarity Search (FAISS). Faiss is a library developed for efficient similarity search and clustering of dense vectors [3]. Flat index, 'IndexFlatIP', is a basic Faiss index to fix and store vectors in an array [3]; hence, the dataset needs to be encoded by another algorithm (such as BERT) before implementing FAISS index. During the search process, each indexed vector is decoded sequentially and then compared to the query vector, which calculates the distance between each indexed vector and query vector [3]. After calculating all these distances, k nearest matches are returned. This step narrows down the search space.

With k candidate documents obtained from Faiss search, a Cross-Encoder, which is a re-ranker, is used to fine-tune the ranking of these documents. A pair of the query and each document are passed through the transformer network to return a relevant score between 0 and 1 of the pair. This step re-ranks and selects the most relevant documents to the given query.

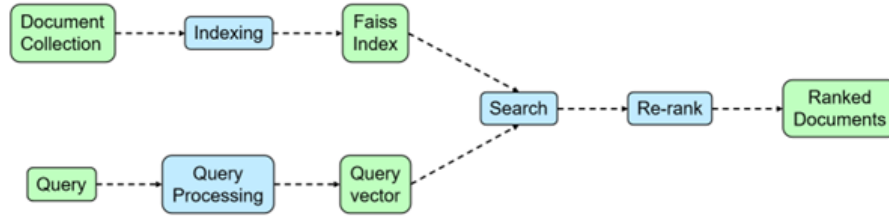


Figure 3. The architecture of IR system

The IR system will be tested with the dataset of 100-word chunks and a dataset of 100 questions as inputs. The output is the MRR score (0.84) and time consumption (35.06s).

To optimize the performance of the IR system, various cross-encoder models will be tested. The results are shown in figure 4. The cross-encoder model ‘*cross-encoder/ms-marco-MiniLM-L-6-v2*’ returns the second best MRR score (0.84) with half of the time cost (19.19s) compared to the model that achieves the best MRR score (34.93s).

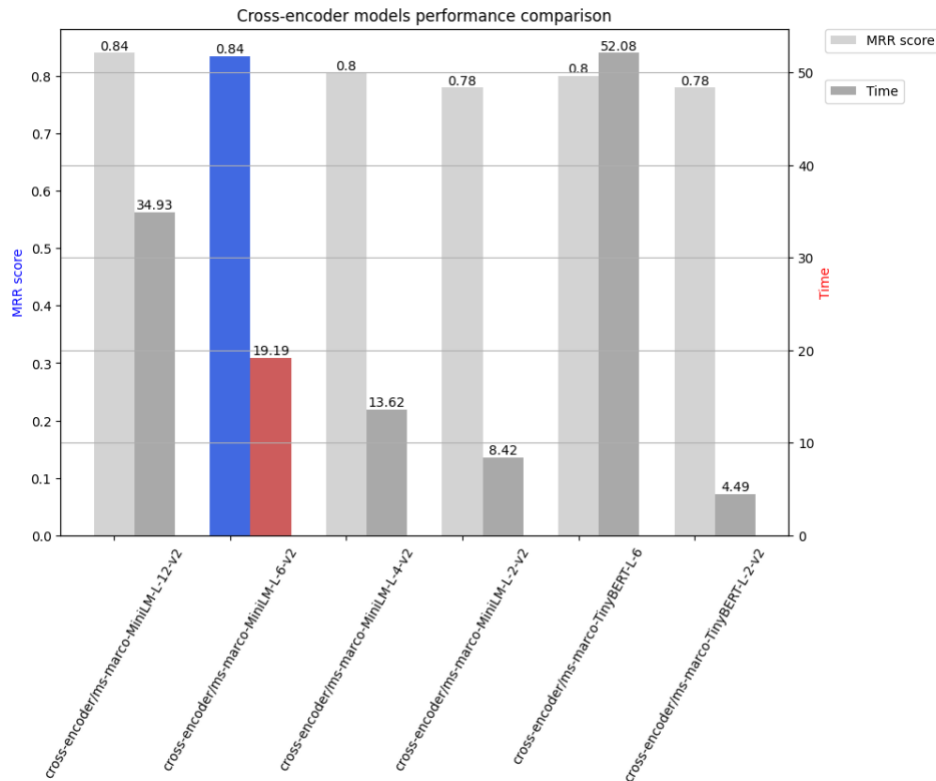


Figure 4. Cross-Encoder models comparison

6.3. Reader Algorithm

The goal of the reader algorithm is to answer an article given a question [1]. In this project, the span of the passage will be extracted as the answer or *span extraction* reader. For example, given a question such as “Who is the CEO of Google?” and a passage about Google organization, the output should be “Sundar Pichai”. Two models will be tested, a simple model Named Entity Recognition (NER) and BERT. The question and article are the input for this model and the output is the span of the answer.

These are the step-by-step for the NER model:

1. Find the entities in the question
2. Find which entities should be found based on the question (*target entities*)
3. Find the entities in the article
4. Find the answer entities in the article given the target entities and the question entities
5. Get the most common answer

The first assumption is the question that the user asks should be related to an entity. Thus, in the first step, the NER model is used to find the entities in the question, we named this *question entity*. For the second step, we created a rule-based model that follows these rules:

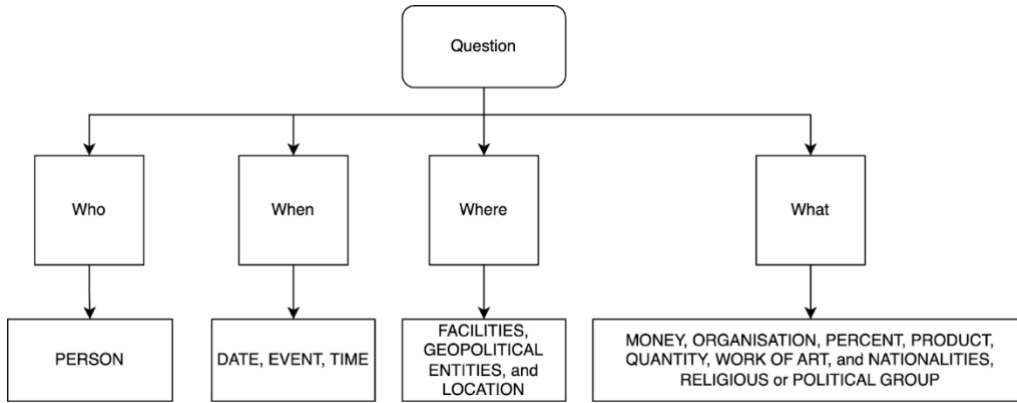


Figure 5. NER *target entity* diagram rule

These are the *target entity* or the entity that the model should find in the article. In the third step, we only used the NER model to find all the entities in the article, we called this *article entity*. In the fourth step, we try to find the answer by assuming that it is located near the *question entity* in the article. However, there could be a lot of entities near the *question entity*, thus the *target entity* role comes. We only care about certain entities related to the user question. The search will expand before and after the *question entity* in the article; in this experiment, we limit it to 15 words. For the last step, if there are several answers with the same entity, the most common word is chosen to be the answer. The example can be seen in Appendix B.

This method is not the most accurate since we need to know the relationship between the entities.

Knowledge Graph could be the solution to this problem, where it could capture the relationship between entities. However, it is not practical to build the Knowledge Graph for each article in this project. Therefore, a more sophisticated method, QA-BERT, will be used.

The chunk of the article from the previous step will be the input of the model as well as the question. The purpose of this part is to make the answer shorter since the generative model has a limited number of tokens that could be inputted into the model. To evaluate QA-BERT, it predicted answers from the testing dataset and evaluated the result using the ROUGE-1 score. This is the model comparison for the BERT and NER models.

Model	ROUGE-1 Precision	ROUGE-1 Recall	ROUGE-1 F1
NER	0.208	0.179	0.188
BERT	0.785	0.806	0.779

Table 1. Reader model evaluation

The NER model precision and recall are 20% and 17.8% respectively, indicating a tendency to predominantly predict incorrect answers and frequently fail to provide answers altogether. In contrast, the BERT model achieved a precision of 78% and a recall of 80%, which is quite respectable considering it was not fine-tuned on the dataset.

6.4. Generative Model

Building a Question-Answer system requires an appropriate generative model as an end product for interacting with users. As can be seen in figure 2, given a query and a snippet of text containing the correct answer, the generative model will answer in a natural language. This part of the report will elaborate on the process of choosing and fine-tuning AI generative models systematically. First, **google/flan-t5-base** was chosen as the initial choice of model. After testing it with a test dataset with 50 questions, and evaluating the results, we decided that it needs to be finetuned. However, even after fine-tuning with the **toughdata/quora-question-answer-dataset** dataset, we found that **google/flan-t5-base** is not appropriate for the task. Finally, we ended up using **GPT-3.5 turbo** as the final generative model for superior performance.

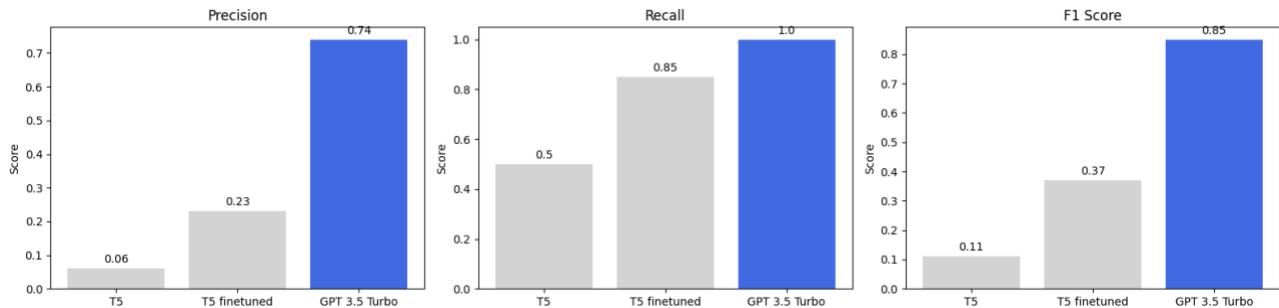


Figure 6. Choosing generative model results

Test google/flan-t5-base

Flan-T5-base has 248M params with a size of 300MB, which is relatively small. Consequently, we expected the model to have a low runtime, which is appropriate for running locally without a powerful server as medium to small businesses.

google/flan-t5-base results are quite poor, we expect the model to perform more robustly. As can be seen from the code, the model performed poorly on reformulating the query given history. The model failed to understand that “he” refers to “Yves Ubelmann”, which means the AI reformulated the question should be “How old is Yves Ubelmann?”. As a result, the generative model cannot answer the question properly.

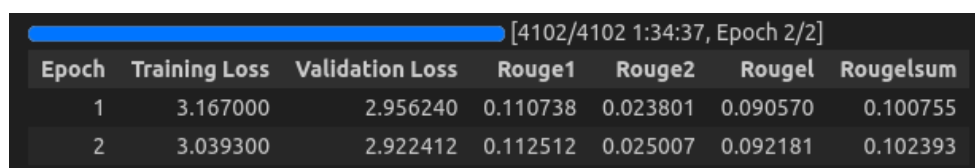
	Input	Output
Question 1	>Human: Who got a call from Syria's director? >Context (from IR and Reader)	>AI: Yves Ubelmann.
Question 2 (following up)	>Human: Who got a call from Syria's director? >AI: Yves Ubelmann. >Human: How old is he?	>AI reformulated the question: How old is he ? (expect to be: How old is Yves Ubelmann?) >AI: None.

Table 2. *flan-t5-base* problem

Runtime is 76.5s for 50 queries, which means an average of 1.5s per query, which is as fast as we expected. As a result, the only concern was the performance of the model, so we decided to finetune it with **toughdata/quora-question-answer-dataset**.

Fine Tuning google/flan-t5-base with toughdata/quora-question-answer-dataset

After fine-tuning with 2 epochs, **google/flan-t5-base** results are still quite poor, we expect the model to perform more robustly. ROUGE scores in the finetuning process all indicated that the performance still not as good as expected.



[4102/4102 1:34:37, Epoch 2/2]

Epoch	Training Loss	Validation Loss	Rouge1	Rouge2	RougeL	RougeSum
1	3.167000	2.956240	0.110738	0.023801	0.090570	0.100755
2	3.039300	2.922412	0.112512	0.025007	0.092181	0.102393

Runtime is 128s for 50 queries, which means an average of 3s per query, which is two times slower than the base model. Additionally, requesting time is limited as shown below.

HfHubHTTPError: 429 Client Error: Too Many Requests for url

For those problems with runtime, number of requests, and overall performance, we had to find a model that could resolve all of those problems, and we found GPT 3.5 turbo.

GPT 3.5 turbo

Not only the accuracy of **GPT 3.5 turbo** is reliable, but also the runtime is as fast as *google/flan-t5-base*, so it does not need to be fine-tuned. Furthermore, the number of requests that could be used is higher than *google/flan-t5-base*. Precision is 0.74; Recall is 1.0; F1-score is 0.85. After considering the result and runtime, we decided to use **GPT 3.5 turbo** as our final generative model.

6.5. Whole System Evaluation

In this part the whole system, from IR, Reader to Generative Model, will be tested. The 100 questions in testing data are utilized as the gold label. However, since the gold label is a snippet from an article and the answer is generated the ROUGE-L value is used to evaluate this part. This is the result:

Precision	Recall	F1-Score
0.7	1	0.82

Table 3. System evaluation

We evaluated the complete system using the dataset used in section 6.4 without “context” columns. The F1-score decreased from 0.89 to 0.82; however, by introducing Reader Algorithm in section 6.3, the number of tokens as a context that is given to the generative model dropped significantly. That might be a concern to businesses that want to use this system for their products as GPT3.5 is paid by number of tokens. We are excited about the system that might help small companies utilize this system to facilitate their work.

7. User Interaction with the System

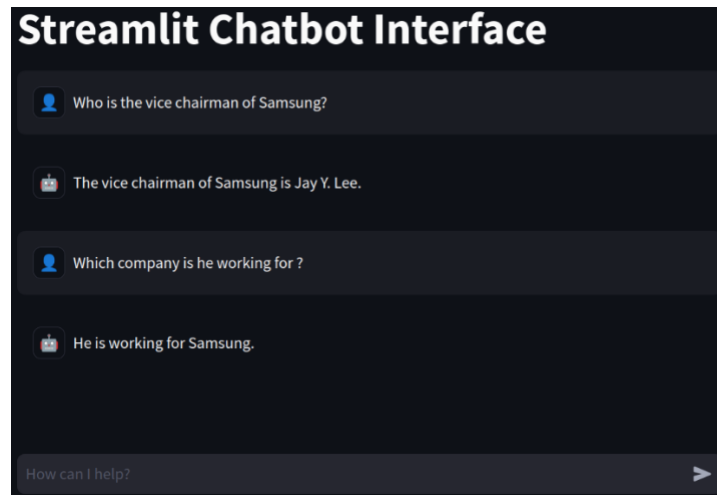
User Interaction is an important feature we introduced to this Question-Answering system. Because we found it takes a long time to run cell by cell in a notebook file, we developed a Streamlit app for running the system on the browser.

- Step 1: Install all dependency in requirements.
- Step 2: Run this following command

```
$ streamlit run app.py
```

```
(.venv) james@james:/media/james/2b44c141-eec6-4c63-a888-30e9ac5660bd/git/NLP_Assignment_2$ streamlit run app.py
You can now view your Streamlit app in your browser.
Local URL: http://localhost:8501
Network URL: http://10.13.107.82:8501
```

The default browser will be automatically opened along with the history.



- Step 3: Enjoy conversation with the agent using the typing box.

8. Conclusion

In this work, we presented a step-by-step framework on how to build a Question-Answering system using a specific database. Given a query, the system will find the answer only from that database and minimize hallucination. The proposed final system has effectively addressed the limitations of current AI chatbots by ensuring data privacy and heavy computational power requirements. Our final product F1-score is 0.85 and runtime overall is 1.5s per query. The system utilized 4 models: FAISS, Cross-Encoder, BERT-QA and GPT 3.5 Turbo. Based on runtime and model size, this system can be used in local compact devices. As a result, this work promoted an ideal solution suited for small to medium-sized enterprises seeking to enhance their data privacy without compromising on performance.

In further research, we recommend that (1) the system should be trained to be able to answer Why and How questions; (2) the history of ranked documents could be saved to retrieve for following questions; (3) implement a generative model that has fewer number of parameters so that it can be run locally without internet connection.

9. References

- [1] Jurafsky, D. & Martin, J. H. 2024. *Speech and Language Processing*. Available at: <https://web.stanford.edu/~jurafsky/slp3/>
- [2] Lin, C.-Y. Rouge: A package for automatic evaluation of summaries. Text summarization branches out, 2004. 74-81.
- [3] Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L. & Jégou, H. 2024. The faiss library. *arXiv preprint arXiv:2401.08281*.

Appendixes

Appendix A

Data and code are available at https://github.com/robinviltoriano/NLP_Assignment_2.git

Appendix B – NER model example

Question: Who is the chairman of Samsung?

Question entity: **Samsung (ORG)**

Target entity: PERSON

Article:

SEOUL (GPE), South Korea (GPE) A special prosecutor investigating the corruption scandal that led to President Park's (PERSON) impeachment summoned the de facto head of **Samsung (ORG)** for questioning on Wednesday (DATE), calling him a bribery suspect. The de facto leader, Jay Y. Lee (PERSON), the vice chairman of **Samsung (ORG)**, will be questioned on Thursday (DATE), according to the special prosecutor's office, which recommended that he also be investigated on suspicion of perjury. Mr. Lee (PERSON) effectively runs **Samsung (ORG)**, South Korea's (GPE) largest conglomerate he is the son of its chairman, Lee (PERSON) who has been incapacitated with health problems.

Number of words search: 15

Answer array: [Jay Y. Lee (PERSON), Park's (PERSON), Mr. Lee (PERSON)]

Answer: Jay Y. Lee

Appendix C – Test cases

	Input	Expected Output
Question 1	Who old was Betsy Morris?	Betsy Morris was 70 years old
Question 2 (test Coreference Resolution and history)	What impact did the opening of the first segment of the Second Avenue subway line have on Lexington Avenue?	It promised to reduce crowding on Lexington Avenue trains
Question 3	Who destroyed Palmyra?	The Islamists destroyed Palmyra
Question 4	What kind of weapon did they use?	The Islamists destroyed Palmyra using explosives
Question 5	Who founded Iconem company?	Yves Ubelmann founded the company Iconem
Question 6	What was Zoraida's early life in Puerto Rico like?	Zoraida's early life in Puerto Rico was like something from a tropical Dickens novel
Question 7	What did she and her siblings doo after a hurricane?	Zoraida and her siblings had to build a house by hand after a hurricane
Question 8	When did she move to New York	Zoraida moved to New York in 1983
Question 9 (test hallucination)	What is the population of Syria in 2024?	I don't know
Question 10	Who is Pascal Butterlin?	Pascal Butterlin is a professor of archaeology at the Sorbonne in Paris
Question 11	What did he say about satellite images?	Pascal Butterlin said people are exchanging satellite images and data on blogs and other research platforms, but we have no real assessment yet because so many ancient sites are not accessible.
Question 12	Where does Sumana Harihareswara come from?	Ms. Harihareswara comes from Astoria, Queens