COMP5423 NATURAL LANGUAGE PROCESSING

Group Project Instruction

Topic: Knowledge Based Question Answering

Group Size: Up 3 Students per Group

Due Date: **April 20, 2025 (Sunday, 23:59)**

Objective:

This project aims to develop a Knowledge Base Question Answering (KBQA) system using the NQ10K dataset, which contains 10,000 questions along with their corresponding answers. The objective is to build a system capable of retrieving relevant information from a structured knowledge base and utilizing this information to generate accurate, informative, and contextually relevant answers to user queries. By integrating retrieval-based methods with generative models, the system will improve the precision and richness of responses, ensuring they are both factually correct and contextually appropriate. The NQ10K dataset will serve as the training and evaluation foundation, with a focus on understanding the application of KBQA techniques to large-scale question-answering tasks. The project will optimize both retrieval and generation components to enhance overall system performance for real-world KBQA applications.

NQ10K Dataset:

We provide the NQ10K dataset for evaluating the performance of your systems. This dataset consists of 10,000 questions randomly sampled from the original 320K Natural Questions dataset. The questions in NQ10K are specifically designed to assess a system's ability to generate accurate, contextually relevant answers by leveraging both information retrieval and text generation methods. The dataset is divided into three parts: 8,000 samples for training, 1,000 samples for validation, and 1,000 samples for testing. Each question is paired with relevant documents, which have been curated into a retrieval knowledge base for your system. This structure enables a comprehensive evaluation of question-answering systems in an open-domain context. The dataset is available at /Content/Group Project/Project Description/data_and_code.zip. You can also download the dataset from the following links:

- NQ10k dataset: https://drive.google.com/file/d/1m11QvNRxu8OaAAaeE_y9wjfotkZDY7Of/view?

usp=sharing

- Original NQ320k dataset: https://ai.google.com/research/NaturalQuestions

NQ10K Dataset Statistics and Data Format:

The data is stored in rows, with each row being in JSON format.

File Name	Count	Data Format
train.jsonl	8,000	{"question": str, "answer": str, "document_id": int}
val.jsonl	1,000	{"question": str, "answer": str, "document_id": int}
val_predict.jsonl	1,000	{"question": str, "answer": str, "document_id": List[int]}
test.jsonl	1,000	{"question": str, "answer": null, "document_id": null}
documents.jsonl	12,138	{“document_id”: int, “document_text”: str}

Project Requirements:

Basic Requirements (Compulsory):

To design a KBQA system, you are required to complete three main components: Knowledge Base Retrieval, Answer Generation, and UI Interface Design:

1. *Knowledge Base Retrieval*: You will need to implement two retrieval methods:
   a) Keyword Matching-Based Retrieval: This is one of the simplest retrieval methods, where keywords or phrases are matched to identify documents relevant to the question. This typically utilizes traditional information retrieval algorithms such as BM25 or TF-IDF.
   b) Vector Space Model-Based Retrieval: In this approach, both the question and documents are converted into vectors, and their similarity is computed to perform retrieval. This often uses pre-trained word embedding models like Word2Vec, GloVe, or FastText to represent words and calculate the similarity between question and document vectors.
2. *Answer Generation:* You are required to implement a method for generating answers:
   a) Answer Generation via Prompting Large Language Models (LLM): Using carefully designed instructions, along with the retrieved content and the question, a prompt is created to guide a LLM to generate an appropriate answer directly. To improve the quality of your generated answer, you may need to experiment with various prompts.

      Hint: To ensure fairness and avoid potential biases introduced by the capabilities of different LLMs, we restrict the usage to Qwen/Qwen2.5-7B-Instruct as the LLM for generating answers via prompting. This model supports multiple languages and has a context length of 32k tokens, which provides

sufficient capacity for handling complex questions and retrieval-based information. You have the option to deploy the model locally or use the free API provided by siliconflow under the Free API Resources section to conduct your experiments.

3. *UI Design*: A user interface (UI) will be developed to allow users to interact with the KBQA system, such as a web interface or a Windows program. The UI should be intuitive and capable of displaying the retrieved documents and the generated answers clearly. This means that your interface should include at least three main components:
   a) User Input for Questions: A section where users can enter their questions, allowing them to interact with the KBQA system by submitting queries.
   b) Retrieved Documents: A display area where the system shows the documents retrieved based on the user's question. This allows users to see the context or references that the system has used to generate the answer.
   c) Answer Display: A dedicated area where the system presents the generated answer, providing users with the most relevant and accurate response to their query.

   You can use the 1,000 samples from the provided validation set to test the performance of your system. This will help you evaluate whether the retrieval methods and answer generation techniques you have implemented are both reasonable and accurate. However, when submitting your result, you are required to provide a file that contains the results after testing on the test set. The file should contain 1,000 lines, with each line in JSON line format. Each line should include three key-value pairs: question, answer, and document_id. These correspond to the question, the generated answer, and the IDs of the five most relevant documents to the question (with the most relevant documents appearing first). You can refer to the provided val_predict.jsonl file to better understand the required format and structure for the output. The evaluation script is provided in metrics_calculation.py.

Advanced Requirements (Compulsory):

To further enhance the performance of your KBQA system, you can incorporate dense vector-based retrieval methods. By preprocessing the document corpus, you can map both documents and queries into a dense vector space. Common methods include Dense Passage Retrieval (DPR) and ColBERT. Once the documents and queries are transformed into dense vectors, you can perform efficient retrieval by utilizing Approximate Nearest Neighbor (ANN) search algorithms, such as FAISS, to find the most relevant documents based on their vector similarities. After identifying the relevant documents, you can proceed with the answer generation process.

Encouraged Requirements (Optional, Encouraged for Implementation):

You are encouraged to explore additional effective retrieval methods beyond the basic&advanced requirements, such as Hybrid Retrieval techniques, which combine different retrieval strategies for improved performance. In your report, you should analyze the effectiveness of the new retrieval methods you explore. This includes providing a clear explanation of why the method is effective, detailing your exploration process, and comparing the performance of the new approach with the baseline methods (basic&advanced). Be sure to quantify the improvements in retrieval quality and discuss how the new method enhances the overall system performance.

References:

Related Dataset:

1. Natural Question Visualization
2. Natural Questions: A Benchmark for Question Answering Research Existing Approaches

Codes:

1. Text_Retrieval_BM25
2. Fasttext
3. SiliconFlow API Document
4. Dense Passage Retrieval
5. ColBERT
6. MR@5 and MRR@5

Related Work (Reference Papers):

1. Reading Wikipedia to Answer Open-Domain Questions
2. End-to-End Open-Domain Question Answering with BERTserini
3. REALM: Retrieval-Augmented Language Model Pre-Training
4. Dense Passage Retrieval for Open-Domain Question Answering
5. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering
6. Qwen2.5 Technical Report

Free GPU Resources:

1. Google Colab: Tutorial Usage
2. Kaggle Kernel: Tutorial Usage

Free API Resources:

1. https://github.com/xtekky/gpt4free
2. https://siliconflow.cn/zh-cn/models

What to Hand In:

1. The Source Code of Your Program
   a) The source files of all your code.
   b) A **readme** file that describes the structure of your system, the environmental dependencies (the used packages and their versions), and how to run your system step by step.
   c) Add **annotations** at the beginning of all code files to indicate their usage and any necessary explanatory comments.
   d) Put all the files into a directory named "code". To avoid submitting excessively large files, your model weight files do not need to be included. If necessary, please submit them in the form of a Google Drive link.
   e) Place the results generated by your system on the NQ10K test set into a file named **test_predict.jsonl** in the **root directory**. Each line should contain a JSON-formatted entry with the fields question, answer, and document_id. The question and answer should be in string format, and document_id should be a list containing **Five** integer values. The TA will use the extract match method to evaluate your answers, and will assess the retrieval results using R@5 and MRR@5. The evaluation script is provided in metrics_calculation.py.
   f) TAs will run the code. If there are problems, TAs will contact your group to demonstrate online. And if the submitted code cannot achieve the reported performance finally, there will be some deductions from your grade.
2. Written Report (At least 6 pages with 12-point font size and single line spacing, excluding the cover page. Please submit the PDF file)
   a) A cover page with **project topic** and **group member' names** and **student IDs.**
   b) The **role** and the **contribution proportion** of each group member. For example, student A 30%, student B 25%, student C 25%, student D 20%.
   c) The method you used to design your system.
   d) Flowchart diagram of your system.
   e) Results analysis. You can analyze the differences in the results generated under different experimental parameter settings (e.g., different instructions).
   f) The UI design of your system.
   g) Anything else you would like to share with us (like the additional exploration).
   h) At the end of the report, provide the electronic signatures of all group members

to ensure the accuracy of the information.

3. Demo Video

   Less than ten minutes

   a) Demonstrate the real-time response capability of the KBQA system and highlight the key aspects of the project solution, such as the main challenges and primary achievements (you can use a PowerPoint presentation to showcase the work).

   b) Provide an appropriate interpretation of the code, such as explaining which part of the code handles data preprocessing, which part is responsible for retrieval, and which part is used for question answering.

Pack all the submissions in one zipped file and submit to Blackboard. As described above, the file contains the "test_predict.jsonl" (file), "code" directory, the report (pdf), and the video file (mp4).

Remark: Please do remember to click the "Submit" button after you upload the file.

Grading Scheme:

We will grade the project according to the following schemes:

1. System Implementation: 40%
2. Written Report: 20%
3. Demo Video: 20%
4. System Performance: 20%

   The TA will compare your provided test_predict.jsonl file with the standard answers in test_ground_truth.jsonl (which is not publicly available) to calculate your system's ranking among all groups. The ranking will be determined based on both answer generation and retrieval performance, with each component contributing 50% to the final score. The final score will be assigned based on your ranking as follows:

   a) Top 10% (including the 10%): Full score (100%).
   b) Rank 10% to 20% (including the 20%): 90% of the total score.
   c) Rank 20% to 30% (including the 30%): 80% of the total score.
   d) Rank 30% to 40% (including the 40%): 70% of the total score.
   e) Rank 40% to 50% (including the 50%): 60% of the total score.
   f) Rank 50% to 60% (including the 60%): 50% of the total score.
   g) Rank 60% to 70% (including the 70%): 40% of the total score.
   h) Rank 70% to 80% (including the 80%): 30% of the total score.
   i) Rank 80% to 90% (including the 90%): 20% of the total score.

j) Rank below 90%: 10% of the total score.

k) Blank submissions will not receive any score.

Assuming the total number of groups is 40, and your system's overall ranking is 16, you can earn 20 * 70% = 14 points in this part.

Note that we will assess your system implementation based on the written report and the code.

Others:

1. The project encourages team collaboration, which means that even if you complete it individually, you will not receive a "solo completion" reward. You will only be graded based on the criteria outlined in the Grading Scheme.
2. Please clearly specify the proportion of each team member's contribution in the report. If the contributions are highly imbalanced, an official report must be provided to explain why the contributions are so uneven. Please note that such imbalances may affect the final evaluation and scoring.
3. At the end of the report, please include an electronic signature to confirm that the report is completed accurately and truthfully.

Contact Information:

- Dongding LIN (22037064r@connect.polyu.hk)
- Heming XIA (23123186r@connect.polyu.hk)