WARWICK BUSINESS SCHOOL
THE UNIVERSITY OF WARWICK

Masters Programmes: Assignment Cover Sheet

Student Number:	**U5538208**
Module Code:	**IBCW9X0**
Module Title:	**Text Analytics**
Submission Deadline:	**6th June 24**
Date Submitted:	**6th June 24**
Word Count:	**1253**
Number of Pages:	**12**
Question Attempted: *(question number/title, or description of assignment)*	**1**
Have you used Artificial Intelligence (AI) in any part of this assignment?	**YES**

Academic Integrity Declaration
We're part of an academic community at Warwick. Whether studying, teaching, or researching, we're all taking part in an expert conversation which must meet standards of academic integrity. When we all meet these standards, we can take pride in our own academic achievements, as individuals and as an academic community.

Academic integrity means committing to honesty in academic work, giving credit where we've used others' ideas and being proud of our own achievements.

In submitting my work, I confirm that:

- I have read the guidance on academic integrity provided in the Student Handbook and understand the University regulations in relation to Academic Integrity. I am aware of the potential consequences of Academic Misconduct.
- I declare that the work is all my own, except where I have stated otherwise.
- No substantial part(s) of the work submitted here has also been submitted by me in other credit bearing assessments courses of study (other than in certain cases of a resubmission of a piece of work), and I acknowledge that if this has been done this may lead to an appropriate sanction.
- Where a generative Artificial Intelligence such as ChatGPT has been used I confirm I have abided by both the University guidance and specific requirements as set out in the Student Handbook and the Assessment brief. I have clearly acknowledged the use of any

generative Artificial Intelligence in my submission, my reasoning for using it and which generative AI (or AIs) I have used. Except where indicated the work is otherwise entirely my own.

- I understand that should this piece of work raise concerns requiring investigation in relation to any of points above, it is possible that other work I have submitted for assessment will be checked, even if marks (provisional or confirmed) have been published.
- Where a proof-reader, paid or unpaid was used, I confirm that the proof-reader was made aware of and has complied with the University's proofreading policy.

Upon electronic submission of your assessment you will be required to agree to the statements above

Table of Contents

1.Introduction

Menstrual awareness entails knowing and addressing the numerous aspects of menstruation with the goal of reducing stigma, promoting health, and ensuring that menstruating individuals have access to the necessary support and resources. Moreover with the advancement in technologies it has now become more easy to monitor and keep a track of it. Nowadays the smart watches are capable enough to monitor the hormonal changes aiding in the management of menstrual disorders. These technological developments are transforming menstrual health management by making it more convenient, sustainable, and informed.

2.Objective

The primary goal of this project is to describe the development of a Retrieval-Augmented Generation (RAG) system that employs a large language model (LLM) and an embedding-based retrieval method. This system is intended to respond to user inquiries by combining the generative capabilities of a language model with the retrieval of relevant information from a specialized dataset. The goal is to build a system that can generate content based on knowledge not found in the LLM's training data, increasing its utility in specialized topics. It showcases the implementation of a Retrieval-Augmented Generation (RAG) system suited for menstruation awareness in healthcare. This system intends to improve the access and accuracy of information linked to menstruation health, giving healthcare professionals, patients, and the general public with trustworthy and tailored insights.

3.Implementation

1.Data Preparation

The dataset used is Menstrual-Health-Awareness-Dataset1. I found this dataset from the Hugging Face. It has a total of 530 rows with 0 null values. It’s a fairly simple dataset with only 2 columns , ‘Instructions’ and ‘Output’. The dataset is in a question and answer format.It has numerous questions with the answers related to the menstrual health.The source of the dataset is not known and nothing much about it has been mentioned in the Hugging Face as well.

2.Model Selection

Embedding Model- sentence-transformers/paraphrase-MiniLM-L6-v2.

- The model was chosen for its ability to produce high-quality sentence embeddings that accurately represent semantic meanings and text relationships.

- It is lightweight and efficient, making it ideal for contexts with limited processing resources, such as running in the T4 GPU of Google Colab.

- It is ideal for tasks requiring semantic similarity and document retrieval.

Generative Model: GPT-2 Medium

- gpt2-medium is a smaller variant of GPT-2 that balances performance and resource consumption.

- It has enough capacity to provide coherent and contextually relevant replies without overburdening the GPU memory.
- Its compatibility with mixed precision (fp16) aids memory management, making it possible to run on a T4 GPU in Colab.

3. Chunking

Combining the Columns:

- To combine columns, each row in the dataset represents a document, with all columns concatenated into a single text string. This ensures that the complete content of each page is used to generate embeddings.

Truncating Texts:

- Texts are limited to a maximum length of 512 characters. This is a reasonable length that strikes a balance between maintaining enough information from each document and keeping the text digestible for embedding and generating chores.

- This truncation helps to minimize memory concerns and makes the embedding process more efficient.

4. Embeddings

Create embeddings.

Batch processing

- It allows for efficient memory management while embedding. To minimize the memory footprint, a batch size of one is chosen.

- The method create_embeddings_in_batches iterates through the texts, encoding each batch into embeddings using the SentenceTransformer model, and then aggregating them.

Embedding Index

- FAISS (Facebook AI Similarity Search) creates an index of embeddings. FAISS is designed for efficient similarity searching, making it ideal for obtaining relevant documents based on query embeddings.
- The embeddings are stored in a flat index (faiss.IndexFlatL2), which is suitable for our needs given the dataset's moderate size.

5. Prompt Template: Formulating Prompts

Context Inclusion

- The obtained documents are concatenated to create a context prompt for the generative model. This context gives the necessary background knowledge for generating appropriate replies.

Dynamic Prompt Construction

- The prompt structure changes based on the type of query. Different inquiry types necessitate distinct prompt styles to direct the language model in producing the proper type of response.

6. Querying

The system is tested with 3 different types of queries:

Factual queries, the prompt asks a straight question based on the given context. The model is meant to provide a factual response based on the information obtained from the papers.

Summary Queries: The prompt demands a summary of the given context. This instructs the model to produce a brief summary of the retrieved documents.

Clarification Queries: The prompt demands additional information on a certain issue specified in the query. This pushes the model to elaborate on the topic by utilizing the context.

Query Processing:

Embedding the Query: The user query is transformed into an embedding using the same model.

This assures that the question and document embeddings are in the same vector space, which allows for more accurate similarity searches. FAISS retrieves the top k most comparable documents based on query embedding. This guarantees that only the most relevant papers are reviewed when generating the response.

Similarity Check: The query and retrieved documents are compared. If the highest similarity falls below a predetermined level (relevance threshold), the algorithm considers the query uninteresting to the dataset and replies appropriately.

4.Generating the response

The prompt is created based on the context and query type. The response is then generated using the generative model (gpt2 medium). This stage makes advantage of mixed precision inference to improve memory utilization.

5.Limitations

The implemented RAG model has following limitations such as the gpt2-medium model has a small context window (usually 512 tokens). This means that for questions with context lengths greater than the model's capacity, some significant information may be missed, thereby impacting the quality of the returned results.The quality of generated responses can vary. While gpt2-medium strikes a fair compromise between performance and resource utilization, it may not always produce the most accurate or coherent results when compared to larger models such as GPT-3 or GPT-4.Document embeddings are static, which means they do not adapt to new data or changes in the dataset. This can limit the system's capacity to give current information.The present implementation supports three sorts of queries: factual, summary, and clarification. More sophisticated or multi-part inquiries may not be adequately addressed.

6.Future Improvements

This RAG system can be further improved by using more advanced models such as GPT-3, GPT-4, or domain-specific models if computational resources allow. Using these models can help improve the quality and relevancy of generated responses. Moreover it will improve the system's capacity to interpret and handle increasingly complicated queries, such as multi-part or conditional inquiries. Apart from that we can implement tools for collecting and integrating user feedback, so that the system may learn from user interactions and improve over time.Also.we can create interactive query refinement processes in which the system can ask additional questions to clarify or narrow the user's inquiry, resulting in more accurate and helpful replies. Investigate multi-modal approaches that combine additional types of data (such as images and audio) into the retrieval and creation processes, resulting in more comprehensive solutions to user inquiries.

7.Conclusion

This RAG system demonstrates the benefits of integrating dense retrieval methods and generative models to improve information retrieval systems. By addressing current constraints and researching future enhancements, such systems can become more robust and versatile, making them useful tools in a variety of disciplines.

Overall, combining retrieval and generating capabilities results in more accurate, contextually appropriate, and coherent solutions to user inquiries by using the characteristics of both methodologies. This research emphasizes the relevance of model selection, timely engineering, and resource optimization in the development of efficient AI-driven solutions.

8.Appendix

Error Handling

OutOfMemoryError: CUDA out of memory. Tried to allocate 20.00 MiB. GPUfurther optimize

To further optimize the script and reduce GPU memory usage, you can:

1. **Reduce `max_new_tokens`**: Lower the number of tokens generated by the model.
2. **Use Mixed Precision:** Enable mixed precision training/inference to reduce memory usage.
3. **Disable unnecessary components:** Make sure all unnecessary components are offloaded from the GPU.

ValueError: Input length of input_ids is 200, but `max_length` is set to 200. This can lead to unexpected behavior. You should consider increasing `max_length` or, better yet, setting `max_new_tokens`.

The error indicates that the input length for the tokenized prompt is reaching the `max_length` limit of 200 tokens, which may result in truncation or other unexpected behaviors. To address this, you can set `max_new_tokens` instead of `max_length`. `max_new_tokens` specifies the maximum number of tokens generated by the model, allowing you to control the length of the generated output without affecting the input prompt length.

```
KeyError                              Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py
in get_loc(self, key)
   3652         try:
-> 3653             return self._engine.get_loc(casted_key)
   3654         except KeyError as err:
```

It seems there might be a problem with the assumption about the column name in your CSV file. Let's update the script to handle the CSV file more flexibly and to ensure the document column is correctly identified.

Here's the revised script:

1. Upload the CSV file and inspect its structure.
2. Use FAISS for document retrieval.
3. Use Hugging Face's `transformers` libra[...]or the generation with GPT-Neo.

Additional Model Research

For embedding

other open souce models than ollama

Certainly! Below are some open-source models that you can use for embedding and generation, which are alternatives to the Ollama models:

Embedding Models

1. **GloVe (Global Vectors for Word Representation)**
   - **Description:** GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus.

For Generation

Generation Models

1. **GPT-2 (DistilGPT-2)**
   - **Description:** DistilGPT-2 is a smaller, faster, and lighter version of GPT-2 that retains much of the original model's generation capabilities.
   - **Advantages:** Faster inference, reduced resource consumption.

9.References

1. gjyotk/Menstrual-Health-Awareness-Dataset: https://huggingface.co/datasets/gjyotk/Menstrual-Health-Awareness-Dataset

2. sentence-transformers/all-MiniLM-L6-v2: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

3. openai-community/gpt2-medium: https://huggingface.co/openai-community/gpt2-medium