Assignment Report: Question Answering System

Submitted by :-

Name : Samyak Jain

Mail ID : samyak.jai@students.iiit.ac.in

Roll No. - 2022201048

Repository Link

Dataset

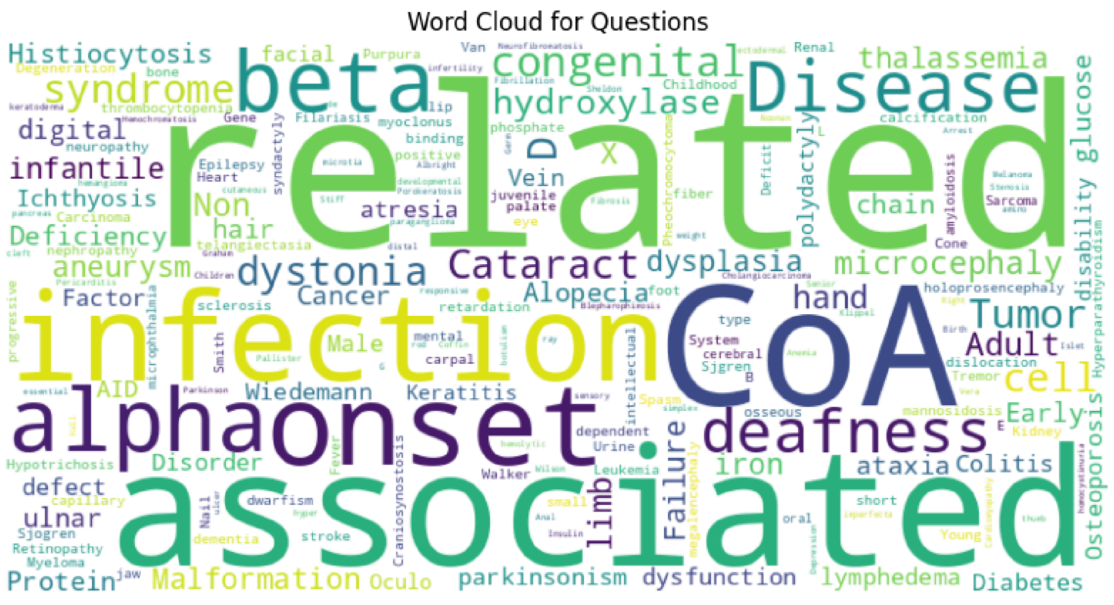
- The dataset provided in the problem statement is called "**MedQuad-MedicalQnADataset**" and is published on HuggingFace. The given dataset can be found here.
- Dataset contains question-answer data of the medical domain. Each sample has **three features**: **['qtype', 'Question', 'Answer']**.
- It has in total **16,407 samples.**

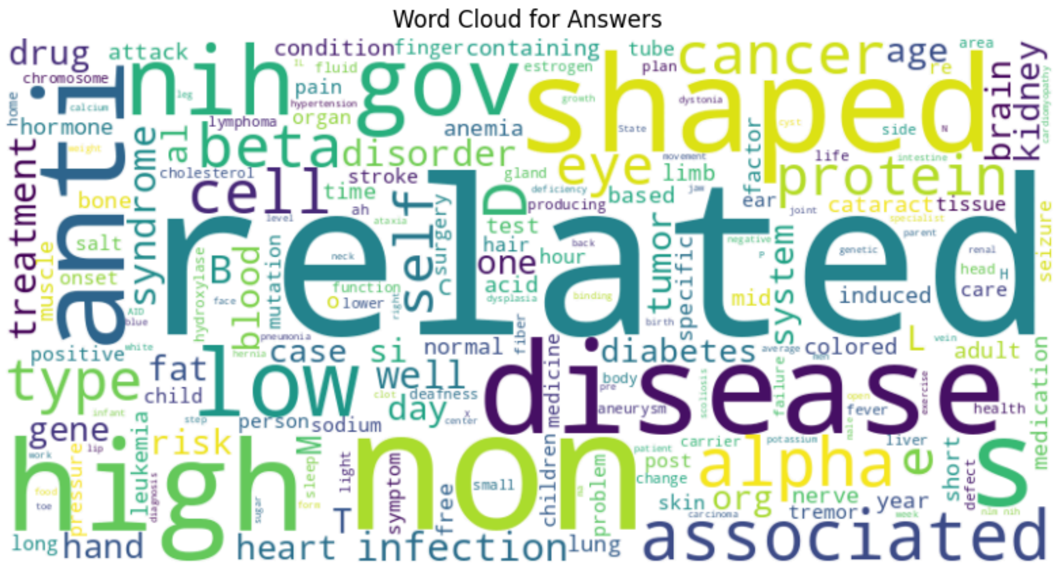
EDA on dataset:

Exploratory Data Analysis (EDA) involves several key techniques, each with its own purpose and utility. Here's a brief overview of each of the techniques applied in the proposed model:

1) Wordcloud:

- **Purpose:** Wordclouds are visual representations of word frequency in a given text dataset. More frequent words appear larger in the cloud, offering a quick visual insight into the most common themes or terms.
- **Implementation:** Using Wordcloud library

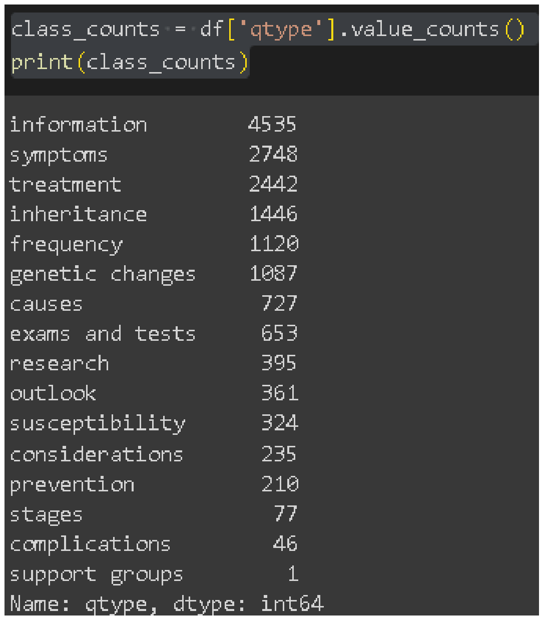




Note: Plotted this using WordCloud library, it internally performs stemming, lemmitizations and stopword removal.

2) Type Imbalance:

- **Purpose:** In the context of EDA for text data, identifying type imbalance involves examining the distribution of classes in a dataset, such as the frequency of different categories in a classification task. Helps ecognizing type imbalance..
- **Implementation:** In the dataset we consider 'qtype' feature as type of question.

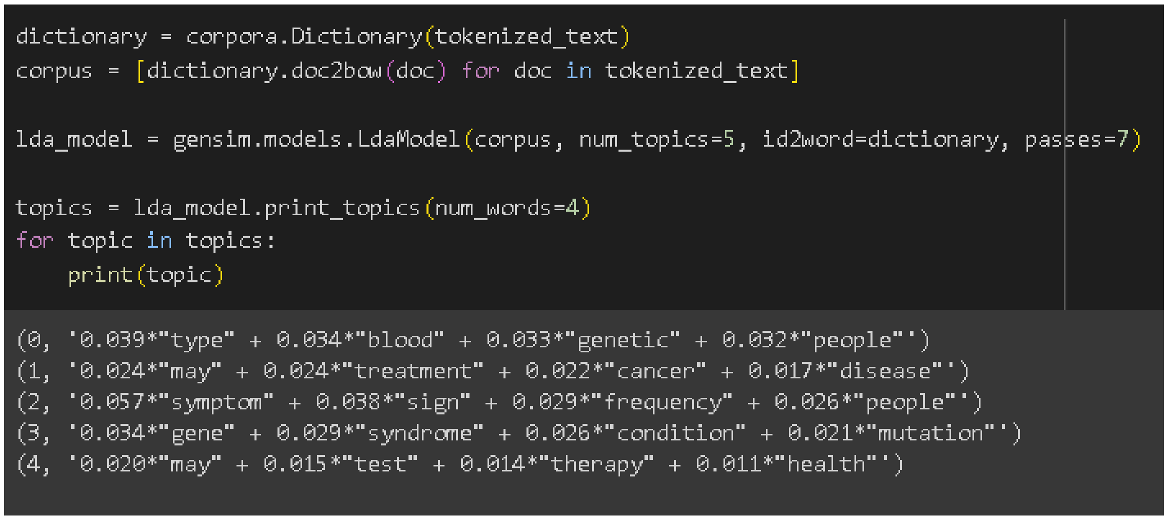


```
class_counts = df['qtype'].value_counts()
print(class_counts)
```

```
information          4535
symptoms             2748
treatment            2442
inheritance          1446
frequency            1120
genetic changes      1087
causes                727
exams and tests       653
research              395
outlook               361
susceptibility        324
considerations        235
prevention            210
stages                 77
complications          46
support groups          1
Name: qtype, dtype: int64
```

3) Topic Modeling:

- **Purpose:** Topic modeling is a method for identifying and grouping similar themes or topics within a large set of text data.
- **Utility:** It's used to uncover hidden thematic structures in a text corpus, enabling the discovery of patterns and trends that might not be apparent from a superficial reading.
- **Implementation:** Using **Gensim** python library and **Latent Dirichlet Allocation (LDA)** approach.



```
dictionary = corpora.Dictionary(tokenized_text)
corpus = [dictionary.doc2bow(doc) for doc in tokenized_text]

lda_model = gensim.models.LdaModel(corpus, num_topics=5, id2word=dictionary, passes=7)

topics = lda_model.print_topics(num_words=4)
for topic in topics:
    print(topic)
```

```
(0, '0.039*"type" + 0.034*"blood" + 0.033*"genetic" + 0.032*"people"')
(1, '0.024*"may" + 0.024*"treatment" + 0.022*"cancer" + 0.017*"disease"')
(2, '0.057*"symptom" + 0.038*"sign" + 0.029*"frequency" + 0.026*"people"')
(3, '0.034*"gene" + 0.029*"syndrome" + 0.026*"condition" + 0.021*"mutation"')
(4, '0.020*"may" + 0.015*"test" + 0.014*"therapy" + 0.011*"health"')
```

Data Preprocessing:

1. **Tokenization:**
   - **Purpose:** Tokenization is the process of breaking down text into smaller units, typically words or phrases. It's a fundamental step in text preprocessing.
   - **Implementation:** Using nltk library

2. **Lemmatization:**
   - **Purpose:** Lemmatization involves reducing words to their base or root form. For example, "running" would be lemmatized to "run". . It allows the model to treat different forms of the same word as one, improving the efficiency and accuracy of the analysis.

Problem Statement

The task is to develop a Question Answering system capable of understanding and responding to questions posed in natural language utilizing the **Medical QnA Dataset**.

Understanding of Problem Statement

In the given data set, we do not have features like 'context' or 'passage' or 'document'. So, it can be treated as an Open **Domain Question Answering task**, utilizing the large language models (like BERT, T5, llama-2, etc).

My Approaches

1. Fine Tuning LLM

One of the first approaches that came to my mind was finetuning a large LLM on the given QnA dataset. I have used "**flan-t5**" which is an enhanced version of T5 that has been fine tuned in a mixture of tasks.

Literature:

T5 - Text-To-Text Transfer Transformer

The Text-to-Text Transfer Transformer (T5) is a flexible and impactful NLP model. It treats all NLP tasks as **text-to-text problems**, streamlining its application across various tasks such as **translation**, **summarization**, **question answering**, and **classification**. Built on the Transformer architecture, T5 is efficient in processing sequential data and recognizing text dependencies. It undergoes a pre-training on a vast corpus (the "**Colossal Clean Crawled Corpus**") using self-supervised learning tasks, followed by fine-tuning for specific tasks.

T5's scalability is reflected in its range of sizes, catering to both less demanding applications and high-end tasks. Notably, its task-agnostic design utilizes a consistent architecture for different tasks, varying only in task framing. T5 has shown exceptional performance in several NLP benchmarks, like **GLUE** and **SQUAD**.

Rouge

ROUGE, which stands for **Recall-Oriented Understudy for Gisting Evaluation**, comprises a group of metrics designed to **assess automatic summarization and machine translation programs** in the field of natural language processing.

These metrics evaluate the quality of an automatically generated summary or translation by comparing it with a reference or set of references that are created by humans. ROUGE is not sensitive to case differences, it treats uppercase and lowercase letters as equivalent.

Steps in Details:

- Load the dataset into the HuggingFace **DataSet**.
- Created a split of 80 : 10 : 10.
- Used the "flan-t5" variant of T5.
- Preprocessed the data and added the prefix "answer the question" to all the questions. T5 is trained on various custom prefixes like "summarization", "classification", etc. So, I am finetuning the T5 for my **custom prefix** "answer the question".
- Hyperprameters:

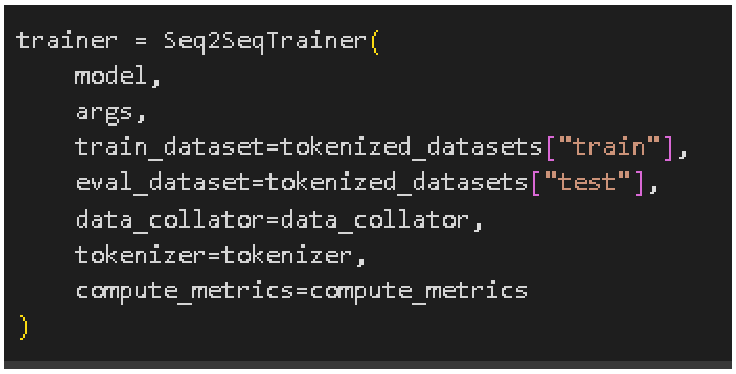
  `max_input_length = 128`, it defines the maximum length question input the model expects, more than that will get trimmed.
  `max_target_length = 512`, it defines the maximum answer length for generation and labels
  `batch_size = 4`, batch size for mini batch gradient descent: the higher the batch size training will be faster but fewer backprop steps.
  `prefix = "answer the question:`", custom prefix used for training
  `no_of_train_epochs = 3`,
  `learning_rate = 3e-4`,
  `per_device_eval_batch = 4`,
  `weight_decay=0.01`,

  These hyperparameters were decided in the basis of some experimentations.

- Training:

  Used **Seq2SeqTrainer** training loop from transformers library itself. With the above parameters and trained on 80% split of total dataset i.e 13125 samples and validated on 10% split i.e 1641 samples.

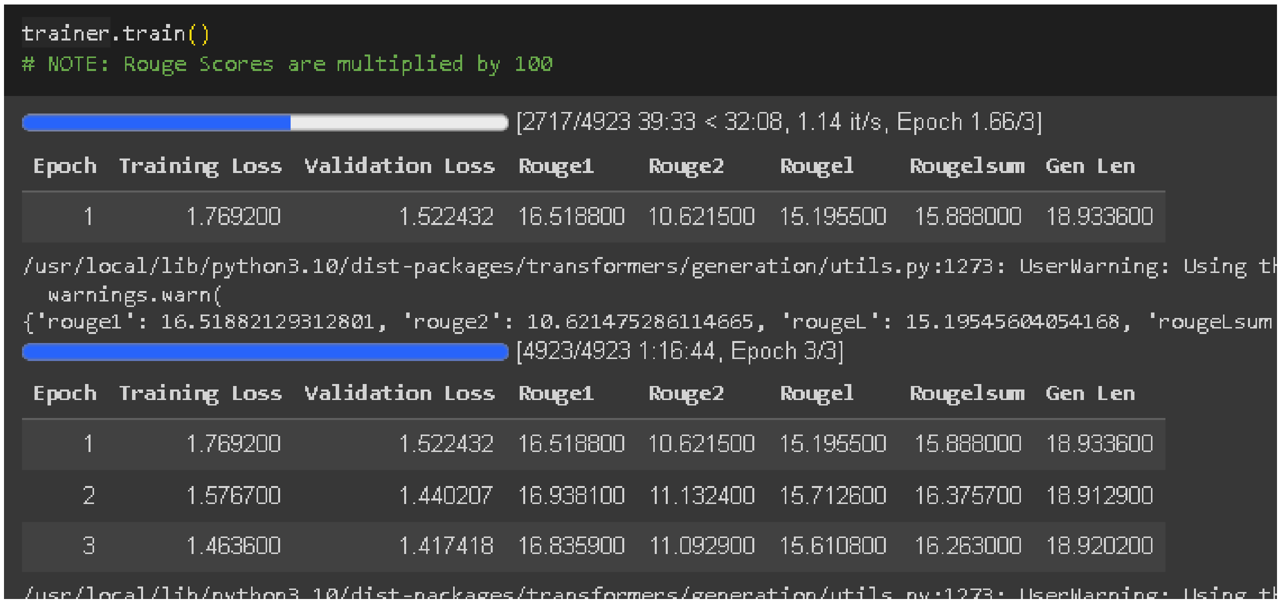
  Apart from the above parameters (as **training arguments**), it requires **model**(flan-t5, which is already imported), **tokenized training** & **validation** data(tokenizer corresponding to the model:flan-t5), **data_collator** which using transformer's class `DataCollatorForSeq2Seq` which requires tokenizer and model as argument, and **compute metric** which is method to calculate loss using the **rouge** as evaluation metric



```
trainer = Seq2SeqTrainer(
    model,
    args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["test"],
    data_collator=data_collator,
    tokenizer=tokenizer,
    compute_metrics=compute_metrics
)
```

- Training Loss:

Below snapshot displays the loss and rouge score mutiplied by 100 for all the epoch.

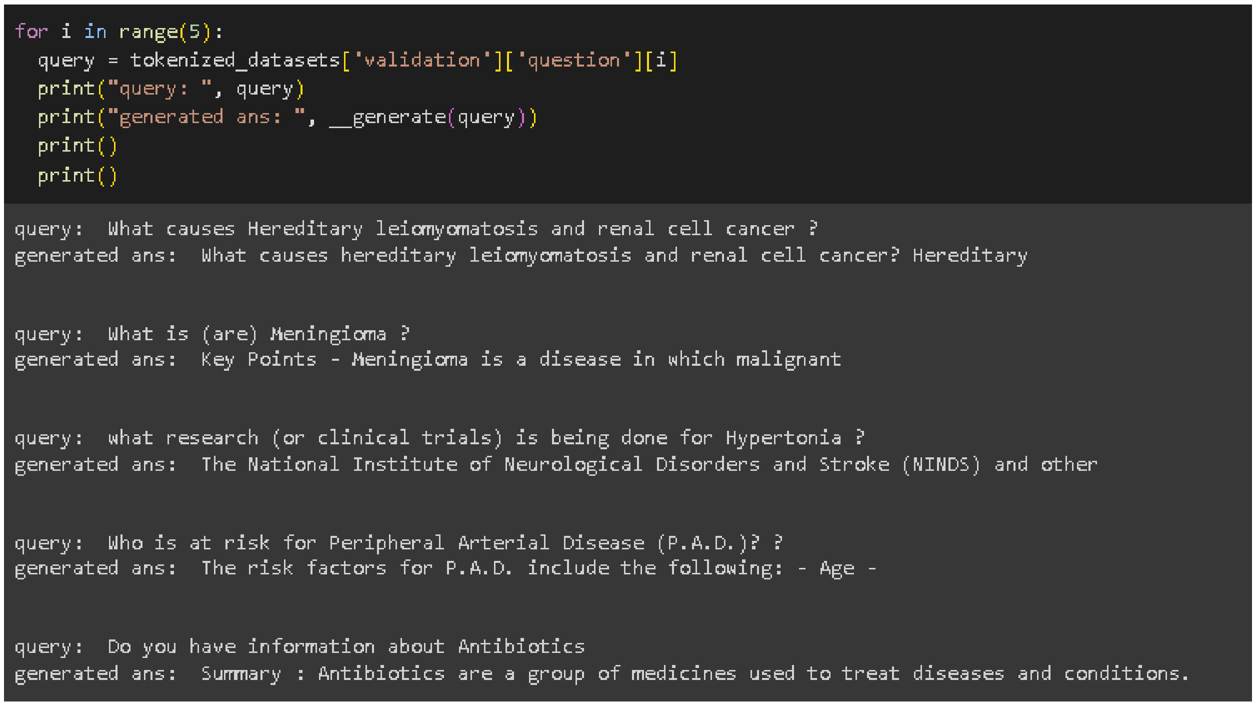


```
trainer.train()
# NOTE: Rouge Scores are multiplied by 100

[2717/4923 39:33 < 32:08, 1.14 it/s, Epoch 1.66/3]
Epoch  Training Loss  Validation Loss  Rouge1     Rouge2     Rougel     Rougelsum  Gen Len
    1       1.769200         1.522432  16.518800  10.621500  15.195500  15.888000  18.933600
/usr/local/lib/python3.10/dist-packages/transformers/generation/utils.py:1273: UserWarning: Using th
  warnings.warn(
{'rouge1': 16.51882129312801, 'rouge2': 10.621475286114665, 'rougeL': 15.19545604054168, 'rougeLsum
[4923/4923 1:16:44, Epoch 3/3]
Epoch  Training Loss  Validation Loss  Rouge1     Rouge2     Rougel     Rougelsum  Gen Len
    1       1.769200         1.522432  16.518800  10.621500  15.195500  15.888000  18.933600
    2       1.576700         1.440207  16.938100  11.132400  15.712600  16.375700  18.912900
    3       1.463600         1.417418  16.835900  11.092900  15.610800  16.263000  18.920200
/usr/local/lib/python3.10/dist-packages/transformers/generation/utils.py:1273: UserWarning: Using th
```

Inference:

Below snapshot displays inferences for the 5 random query from validation data.



```
for i in range(5):
  query = tokenized_datasets['validation']['question'][i]
  print("query: ", query)
  print("generated ans: ", __generate(query))
  print()
  print()
```

```
query:  What causes Hereditary leiomyomatosis and renal cell cancer ?
generated ans:  What causes hereditary leiomyomatosis and renal cell cancer? Hereditary


query:  What is (are) Meningioma ?
generated ans:  Key Points - Meningioma is a disease in which malignant


query:  what research (or clinical trials) is being done for Hypertonia ?
generated ans:  The National Institute of Neurological Disorders and Stroke (NINDS) and other


query:  Who is at risk for Peripheral Arterial Disease (P.A.D.)? ?
generated ans:  The risk factors for P.A.D. include the following: - Age -


query:  Do you have information about Antibiotics
generated ans:  Summary : Antibiotics are a group of medicines used to treat diseases and conditions.
```

Observation & Conclusions:

- The observation that the answers provided are **grammatically correct** indicates that the **system in question possesses the ability to generate coherent text**. This ability reflects a fundamental understanding of language structure and syntax, essential components of text generation processes.
- However, the fact that **these answers don't really connect with the questions** raises a different concern. This mismatch between getting the grammar right and actually addressing the queries indicates that although the system can put together sentences that sound good, but it might be missing a more thorough understanding..
- The reason for this **knowledge limitation might be due to less domain specific training data** of the Large Language Model (LLM).

- It implies that while the system can process and produce language, it **struggles to effectively understand and interpret the specific content** the queries.

2. Retrieval Augmented Generation(RAG) Approach

Literature Survey:

Retrieval-Augmented Generation (RAG) is the technique to provide LLMs with additional information from an external knowledge source. This allows them to generate more accurate and contextual answers while reducing hallucinations. combines the strengths of two different approaches: neural language models and information retrieval systems. This method enhances the capabilities of AI models, especially in tasks like language understanding and generation

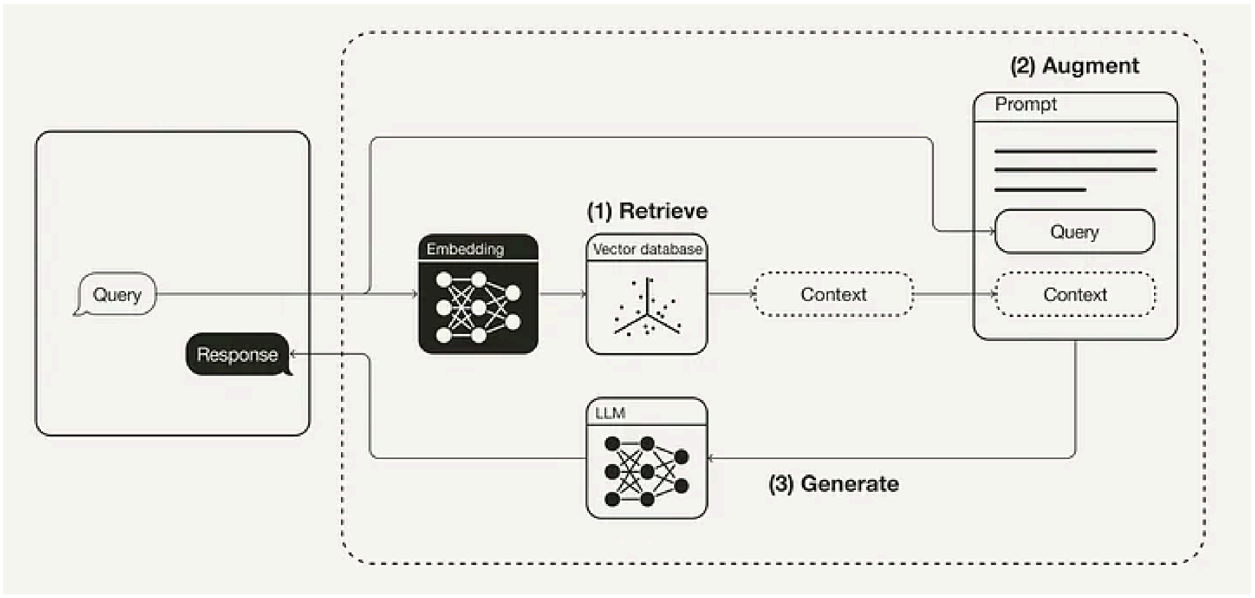
Problems with Finetuning LLMs:

State-of-the-art LLMs are trained on large amounts of data to achieve a broad spectrum of general knowledge stored in the neural network's weights (parametric memory). However, prompting an LLM to generate a completion that requires knowledge that was not included in its training data, such as newer, proprietary, or domain-specific information, can lead to factual inaccuracies (hallucinations)

RAG Approach:.

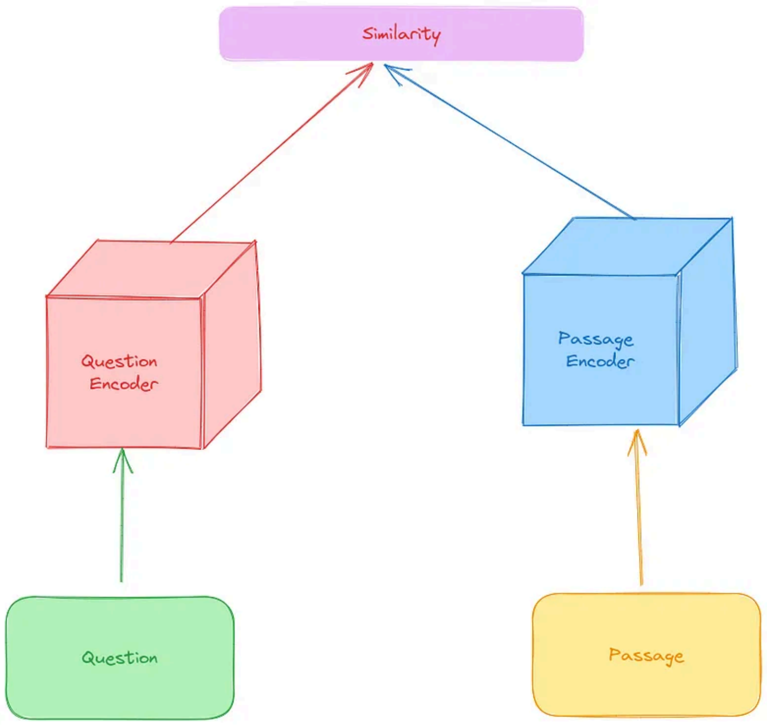
Here's a breakdown of how it works:

1. **Retrieve:** The user query is used to retrieve relevant context from an external knowledge source. For this, the user query is embedded with an embedding model into the same vector space as the additional context in the vector database. This allows to perform a similarity search, and the top k closest data objects from the vector database are returned.
2. **Augment:** The user query and the retrieved additional context are stuffed into a prompt template.
3. **Generate**: Finally, the retrieval-augmented prompt is fed to the LLM.



Retriever:

For retriever model, I have implemented **DPR**(Dense Passage Retrieval) approach. DPR is a **dual-encoder framework**. Meaning there are two encoders present in this architecture. One encoder is used to encode a question, while the other encoder is used to encode the passages into dense embeddings as follows:



Reader:

For reader model, I am using the “flan-t5” model.

References:

1. https://arxiv.org/abs/2312.10997
2. https://arxiv.org/abs/2005.11401
3. https://towardsdatascience.com/retrieval-augmented-generation-rag-from-theory-to-langchain-implementation-4e9bd5f6a4f2