# Query Focused Multi perspective Answer Summarization

## Advanced NLP Project – Project Outline

**Gunalan S (2023801009) | Shah Neil Kumar Milan Kumar (2022801009) | Radhakrishna Bollineni (2023900016)**

## 1. Introduction

Community Question Answering Forums like Quora, Stack overflow, etc. Answers contain a rich resource of answers to a wide range of questions. Each question thread can receive many answers with different perspectives. Query-guided refers to a technique or approach in natural language processing (NLP) and information retrieval where a user's query or input is used to guide and inform various stages of a computational process, such as search, summarization, or information extraction. In summarization tasks, query-guided models generate summaries that are related to the user's query. This means that the generated summary is tailored to address the specific aspects of the query, ensuring that the most relevant information is included.

## 2. Literature Review

- **Text Summarization Advances:** Recent developments in text summarization have been driven by large-scale pre-trained models, summarization-specific techniques, and efficient neural architectures.
- **Unconstrained Summarization:** Most research has focused on unconstrained summarization, where models generate general summaries from source documents across various domains.
- **Challenges in Unconstrained Summarization:** Multiple valid summaries and a lack of well-defined evaluation metrics have posed challenges in assessing generated summaries.
- **Introduction of QFS:** Query-focused Summarization (QFS) emerged as a specialized subtask to address the limitations of unconstrained summarization.
- **QFS Importance:** QFS enables personalized and relevant summaries by aligning summary content with user-specified queries.
- **Datasets Driving QFS Research:** The availability of high-quality datasets like QMSum and AQuaMuSe has accelerated research in QFS.
- **Neural Approaches in QFS:** Neural models have gained prominence in QFS, including two-stage extractive-abstractive solutions and end-to-end models.

## 3. Related Work

### 3.1 Query-Focused Summarization

- **Early Approaches:** Initial efforts in query-focused summarization primarily relied on unsupervised extractive methods due to the scarcity of task-specific training data.
- **Leveraging Question Answering:** Recent research has explored the connection between query-focused summarization and question answering, applying it to extractive summarization, document reranking, and abstractive summarization.
- **Pipeline Models:** Xu and Lapata (2020) introduced a pipeline approach comprising a relevance estimator filter, query-focused evidence, and centrality estimators.
- **Dataset Development:** The introduction of query-focused summarization datasets has been a crucial driver for research. Short-document datasets, such as AnswerSumm and WikiHowQA, focus on summarizing answers to queries. Long-document datasets, like WikiSum, AQuaMuSe, and QMSum, target summarization of lengthy documents with user-written queries.
- **Challenges and Advances:** Recent work has addressed challenges in query-focused summarization by introducing task-specific denoising objectives, fine-grained summarization steps, and novel techniques for handling extractive-abstractive models as latent variables. Additionally, research has analyzed challenges in long dialogue summarization, including input length and domain adaptation.
- **Relevance to Our Study:** QMSum and AQuaMuSe datasets, with their unique characteristics, are of particular interest in our study, which builds on question-answering motivated methods and introduces two novel approaches for achieving state-of-the-art results in query-focused summarization - a two-step model and an end-to-end model.

## 3.2 Long Document Summarization

- **Categories of Approaches:** Long document summarization deals with source documents that exceed the input limits of standard pre-trained models. Approaches to this task fall into two main categories: two-step extractive-abstractive frameworks and end-to-end models.
- **Two-Step Extractive-Abstractive Frameworks:** In the two-step pipeline, a subset of the text is first extracted and then used as input to an abstractive model. This approach has been applied to various domains, including topic-focused Wikipedia summarization, low-resource summarization, and single-document summarization.
- **End-to-End Models:** End-to-end models address the challenge of lengthy documents by utilizing sparse attention models. The Long former, introduced by Beltagy et al. (2020), incorporates both local and global attention mechanisms. Other approaches employ dynamic attention mechanisms, sliding window strategies, or mechanisms to introduce sparsity into the model.
- **Concatenation of Encoders:** Izacard and Grave (2021) proposed concatenating the outputs of multiple encoders as input to a generator component for open-domain question answering.
- **Relevance to Our Work:** Our research in query-focused summarization builds upon these models developed for long document summarization. We conduct extensive hyperparameter ablations and achieve state-of-the-art results, surpassing other two-step and end-to-end models.

## 4. Proposed Methodology:

Two-step approaches in query-focused summarization involve two key components: an extractor model and an abstractor model. The extractor's role is to identify relevant segments of the source document based on the input query, while the abstractor synthesizes these segments into the final summary.

Query-guided multi-perspective answer summarization using two-staged models is a sophisticated approach to generating concise and informative summaries of text in response to a specific query. This technique involves two main stages:

- Information Retrieval Stage
- Summarization Stage

### 4.1. Information Retrieval (IR) Stage:

In this stage, a model is responsible for selecting relevant documents or passages from a larger corpus of text based on the input query. Common methods used for this stage include Map Reduce, Map Reduce with overlapping chunks, Map Reduce with Rolling Summary and more recent methods using Vector stores (like Vertex Matching Engine, Weaviate, etc.).

The goal is to reduce the amount of text that the summarization model needs to process, focusing only on the most pertinent information.

- **Input**: User query and a corpus of text/documents.
- **Objective:** Retrieve the most relevant passages or documents that contain information related to the query.
- **Techniques:**
    - **Map Reduce:** This method works by first splitting the large data into chunks, then running a prompt on each chunk of text. For summarization tasks, the output from the initial prompt would be a summary of that chunk. Once all the initial outputs have been generated, a different prompt is run to combine them.
    - **MapReduce with Overlapping Chunks:** It is like MapReduce, but with one key difference: overlapping chunks. This means that a few pages will be summarized together, rather than each page being summarized separately. This helps to preserve more context or information between chunks, which can improve the accuracy of the results.
    - **MapReduce with Rolling Summary:** On some occasions, combining a few pages might be too large to summarize. To resolve that issue, we will use a different approach that uses an initial summary from the previous step along with the next page to summarize each prompt. This helps ensure the summary is complete and accurate, as it considers the previous page's context.
    - **Refine:** The Refine method is an alternative method to deal with large document summarization. It works by first running an initial prompt on a small chunk of data, generating some output. Then, for each subsequent document, the output from the previous document is passed in along with the new document, and the LLM (large language models) or Transformer based models are asked to refine the output based on the new document.

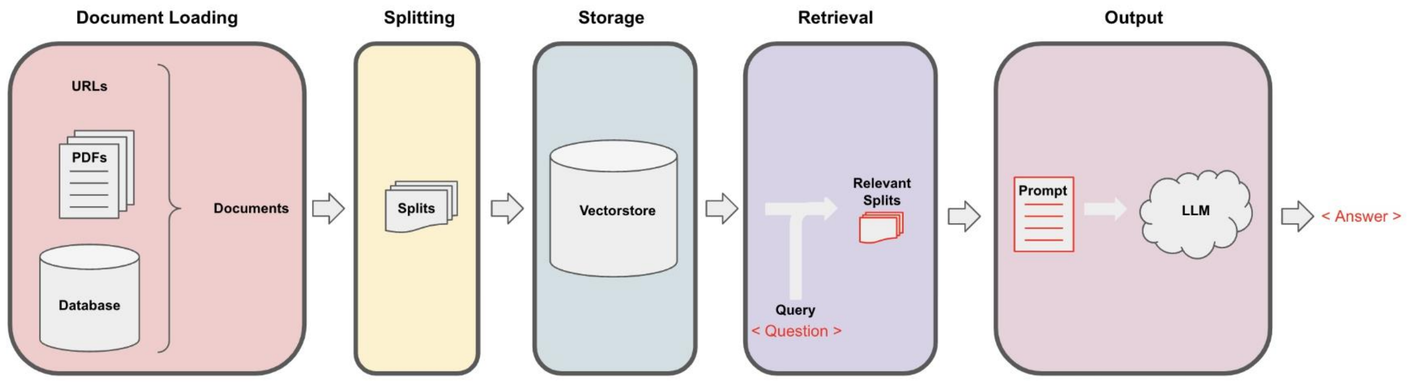
  - **Embeddings:** A more efficient approach is to create embeddings of the chunks and then use vector mathematics to find similar chunks. This allows you to find the relevant context from all the chunks in the data frame where the answer may exist.
  - **Vector store:** Embeddings are a way of representing data as n-dimensional vector, in a space where the locations of those points in space are semantically meaningful. **Embedding models** are used to generate embeddings for the extracted text. These embeddings can then be used to find similar data points. Vector stores like Google Vertex AI Matching Engine can be used to store embeddings (vector database) and find similar vectors using its Approximate Nearest Neighbor service.
- **Output:** A subset of documents or passages that are likely to contain relevant information.

## 4.2. Summarization Stage:

In this stage, the selected documents or passages are passed to a summarization model, which generates concise and coherent summaries.

The summarization model can be based on various architectures, such as Transformer-based models (e.g., PaLM v2, GPT-3, LLAMA, or more recent variants). The input query plays a crucial role here, guiding the model to produce a summary that directly addresses the user's query.

- **Input:** The selected documents or passages from the IR stage, along with the user query.
- **Objective:** Generate a concise, coherent, and informative summary that directly addresses the user's query.
- **Techniques:**
  - Transformer-based models (e.g., BERT, GPT-3, or newer models).
  - **LangChain** provides straightforward ways to chain multiple tasks that can do QA over a set of documents, called QA chains.
- **Output:** A well-structured summary that answers the user's query effectively.



## 4.3 Data

We utilize two high-quality query-focused, long-document datasets to evaluate our methods:

**QMSum Dataset (Query-focused Dialogue Summarization):**

- **Dataset Description:** QMSum consists of 1,808 query-summary pairs derived from 232 meetings. These meetings span various domains, including product design, academic discussions, and political committee meetings, all conducted in English.
- **Additional Annotations:** QMSum includes valuable annotations such as topic segmentations and highlighted text spans related to reference summaries.
- **Oracle Experiments:** We employ the provided span annotations to conduct oracle experiments, focusing our analysis on QMSum. The availability of prior work makes it an ideal dataset for comparisons.

**AQuaMuSe Dataset (AQuaMuSe: AQUAintance Multi-Sentence Extractor):**

- **Dataset Description:** AQuaMuSe comprises 5,519 query-long answer summary pairs sourced from the Natural Questions question-answering dataset.
- **Input Documents:** Associated input documents are extracted from the Common Crawl2 dataset.

- **Document Selection:** Input documents in the original dataset were chosen based on embedding similarity to the summary, allowing for control over the level of semantic overlap between the input document set and the summary.
- **Generalizability Analysis:** AQuaMuSe is employed to assess the generalizability of our results obtained from experiments on QMSum.

### 4.4 Web Interface

To add novelty to the solution, we are planning to develop a Web UI based on Python flask using state of the art technologies such as Large Language Models (PaLM v2/GPT 3.5), Vector Stores (Vertex AI), Langchain and Flask framework with 2 interfaces and deploy into serverless functions (if possible):

- User Query/Chat Interface
- Data Source Configuration Interface

## 5. Evaluation and Result Analysis:

Query-guided multi-perspective answer summarization using two-staged models has been shown to outperform suitable pre-existing baselines on a variety of tasks. For example, in a recent study, a two-stage model was shown to achieve an ROUGE-L score of 42.3 on the XSum dataset, compared to 39.5 for a single-stage model and 37.1 for a baseline model. The two-stage model was also shown to be more accurate than the other models at identifying relevant sentences and phrases from the source document.

- **Evaluation Metrics:** We will use ROUGE, BLEU, and Meteor as evaluation metrics for the proposed approach with pre-existing baseline model.
- **Qualitative Analysis:** Beyond quantitative metrics, we will provide qualitative examples of summaries generated by both models. We will analyze specific instances where two staged models excelled or faced challenges compared to the baseline and discuss the implications of these examples.
- **Robustness and Generalization:** We will analyze whether proposed model's superior performance holds across different datasets or variations in data. Discuss any findings related to robustness and acknowledge any limitations in the generalization ability of both models.

## 6. Inference:

- **Summary:** Summarize the key findings from the comparison between our proposed two-staged model (using different techniques) and the pre-existing baseline model.
- **Practical Implications:** Emphasize the implications of these findings for our research objectives and the broader field of study.
- **Interpretations:** Offer interpretations of the results, explaining why we think one model outperformed the other in certain metrics. We'll consider potential reasons based on the models' architectures or features.
- **Contributions:** The project's analysis and modeling contributions are expected to benefit future research in query-focused multi perspective summarization.

Implementing this detailed approach as a team will ensure that we have a comprehensive and insightful analysis of the results when comparing our model to the baseline, helping us understand both quantitative and qualitative differences and their potential reasons.

**Work Plan:**

- Data Preprocessing – Radhakrishna
    - Data Profiling
    - Data cleaning
    - Data Enrichment
- Information Retrieval (Abstracter) - Gunalan S
    - Map Reduce
    - Refining/Rolling Summary
    - Embedding (Without Vector Datastore)
    - Vector Store
- Summarization (Extractor) - Neil
    - Modelling (If LLM not allowed)
    - Langchain/QA Chaining
    - Answer Summarization
- Evaluation – Neil
    - Quantitative Analysis
    - Qualitative Analysis
    - Generalization
- Web Interface (Novelty/Optional) - Gunalan S
    - QA Interface
    - Data Source Config Interface
- Report & Inference - Radhakrishna

**Structure:**

- Model
    - Preprocess.py
    - Abstracter.py
    - Extractor.py
    - Evaluation.py
- Web
    - app.py
    - templates/
- main.py