CrisisFACTs Track

INTRODUCTION:

The **CrisisFacts Summarisation** task automatically summarizes key facts and details related to a crisis from various sources like **twitter, reddit, news** etc. It uses natural language processing techniques to identify important information, such as crisis type, location, casualties, and actions taken. The summarized information is presented in an easy-to-understand format, aiding decision-making and **emergency response efforts.**

DATASET DESCRIPTION:

- Efficient Data Extraction: By using the CrisisFacts dataset and the folder structure of crisisfacts/<eventNo>/<day>, you were able to efficiently extract data for specific events and days. This approach allows for targeted data retrieval, which can be useful in crisis management scenarios where time is of the essence.

- Effective Text Ranking: By utilizing the PyTerrier library to rank the texts extracted for a particular query, you were able to identify the most relevant information for a given situation. This approach can help to streamline decision-making processes in crisis situations, as decision-makers can quickly access and prioritize the most important information.

- Accurate Information Retrieval: By using both the CrisisFacts dataset and the PyTerrier library, you can be confident that the information you are retrieving is accurate and reliable. This is particularly important in crisis situations, where misinformation can be dangerous and even deadly. By using a data-driven approach to information retrieval, you can help to ensure that the decisions you make are based on the most up-to-date and reliable information available.

T5 - Text-to-Text Transfer Transformer:

Here's an overview of what each function does:

- preprocess(texts) - takes in a list of texts, encodes them using a tokenizer, and returns a dictionary of encoded inputs suitable for input into a summarization model.
- uni_sen(text) - takes in a list of sentences, removes duplicates, and returns a list of unique sentences.
- generate_summaries(inputs) - takes in encoded inputs, uses a pre-trained model to generate summaries, removes duplicates, and returns a list of unique summaries.
- summarize_dataframe(df) - takes in a pandas dataframe with a 'text' column, generates summaries for each text, adds a 'summary' column to the dataframe, and returns the first 20 rows of the updated dataframe.
- filter_summaries(query, texts, summaries) - takes in a search query, a list of texts, and a list of summaries, and returns a filtered list of summaries that contain the query.
- Calls the getDaysForEventNo(event) function to get a list of dates for a particular event.Asks the user to input a date from the list.Loads a dataset for the specified event and date using ir_datasets.load().
- Creates a pandas dataframe of the search results for the input query using a PyTerrier retriever.
- Uses the summarize_dataframe() function to generate summaries for the search results.
- Filters the summaries using the filter_summaries() function based on whether they contain the input query.
- Prints the filtered summaries.

TEXT RANK:

This code defines a function called summary_textrank that takes in two arguments: query and retriever. Here's an overview of what the function does:

- Used GloVe for word embeddings.
- Creates a pandas dataframe of the search results for the input query using the input retriever.
- Tokenizes the text in the dataframe into sentences using sent_tokenize.Cleans the sentences by removing punctuation, numbers, and special characters, making the text lowercase, and removing stopwords.
- Computes sentence embeddings using pre-trained word embeddings.Computes a similarity matrix between the sentences using cosine similarity.
- Constructs a graph using the similarity matrix and computes the PageRank scores for each sentence.
- Sorts the sentences by their PageRank scores and concatenates them into a single summary.

GPT-2:

- The generate_summary function takes in a query and a retriever object and returns a summary of the top-scoring documents retrieved by the retriever based on the query.

- It first retrieves the documents using the retriever and sorts them by their score in descending order. Then it concatenates the text of the top-scoring documents into a single string and encodes it using the tokenizer.

- The model is then used to generate a summary from the encoded input text. The max_length parameter limits the length of the summary to 350 tokens, num_beams specifies the number of beams to use during beam search decoding, no_repeat_ngram_size prevents the same n-gram from appearing twice in the summary, and early_stopping terminates the beam search as soon as the model generates an end-of-sequence token.

- The generated summary is then decoded using the tokenizer and returned as a string.

BART MODEL:

- This is a Python function for generating summaries of input texts using BART, a pre-trained sequence-to-sequence model.

- The function takes a list of texts as input and tokenizes them using the Hugging Face tokenizer.

- It then generates a summary for each text using the BART model, with specified parameters for length, repetition, and penalty.

- The generated summaries are decoded from token IDs to human-readable text and appended to a list of summaries.

- This function returns a concatenated string of all the summaries. This function can be used for various text summarization tasks, such as news articles