POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Topic-Aware Multi-Stream Text Retrieval for Crisis Related Summarization

Supervisors

Prof. Paolo GARZA

Dott. Daniele REGE CAMBRIN

Candidate

Claudiu Constantin TCACIUC

April 2025

Abstract

The amount of information produced daily reaches terabytes of data being able to retrieve important facts timely is proving to be an arduous challenge. The retrieval of key information during crisis events is an important task that can help the survival of people present in the affected areas.

In the last years, with the advance of Natural Languages Processing (NLP) models and retrieval system multiple solutions have been proposed to address this task, also thanks to the TREC CrisisFACTS challenge focusing on temporal retrieval.

In this thesis, we exploited topic modeling techniques like BERTopic, capable of creating clusters of semantically similar data, in combination with dense and lexical retrieval like BM25 and encoders models, and neural reranking exploiting RR (Retrieve & Re-Rank) algorithm to retrieve information from different types of social media (Facebook, Twitter, Reddit, and news outlets) in order to produce a summary of their content in the context of crisis event management.

The experimental results show our solution is able to retrieve useful facts and produce accurate summaries during crisis events. We achieve higher ROUGE and BERTScore than the means results obtained by CrisisFACTS participants without affecting the scalability.

Acknowledgements

I would like to thank professor Paolo Garza and Daniele Rege Cambrin for all the help provided during the development of this thesis, without their guidance some parts of this project would not have been possible to ultimate.

I also would like to thank my family and friends for all the support they gave me during all these years, without them I would not be here writing this thesis. I'm especially glad to my parents, Lidia and Orest, for giving me the opportunity to pursue my dreams and for their unwavering support throughout all these years.

I'm also deeply grateful to my friends Alessio, Oana, Luca C., Leandro, Luca S., Riccardo, Gabriele, Lorenzo and Matteo for their support, encouragement and companionship throughout these years. You made this experience more enjoyable, filling it with unforgettable moments, laughter, and kindness.

I have to thank HPC@POLITO for providing computational resources to work on this project.

Table of Contents

List of Tables

List of Figures

Acronyms

AI
artificial intelligence

IR
information retrieval

NLP
natural language processing

DL
deep learning

BM25
Okapi BM25

IDF
inverse document frequency

RR
Retrieve & Re-Rank

NN
Neural Network

BERT
Bidirectional encoder representation from transformers

SOTA
state-of-the-art

LLM

large language models

Chapter 1

Introduction

1.1 Crisis Event Management

In crisis scenarios, timely and accurate information is crucial for effective emergency response, being able to rapidly assess damage and provide help to those in needs can minimize the loss of life. When dealing with natural disaster such as wildfires, hurricanes and floods or industrial accidents, such as facilities explosions (e.g. 2020 Huston Explosion) or public health crisis (e.g. Covid 19), those who are responsible for allocating resources, coordinate rescue efforts and provide instruction to the population rely on precise and up-to-date information. The ability to rapidly assess damage and provide effective help to those in needs can be the difference between life and death. With the advent of social media platforms, the amount of insights available during a crisis event vastly increases over time due to the fact that nowadays everyone can share information using such platforms [1, 2]. However, as the amount of data increases the critical information becomes more volatile, delays in identifying useful facts can lead to inefficient resource allocation, which leads to an increase in casualties. Being able to quickly retrieve useful information is not only a logistical concern, but also a moral imperative in emergency management.

1.1.1 Social Media Platform in Crisis Communication

In today's world, where everyone has access to the digital world, the spread of information has completely transformed. During a disaster, social medial platforms (Twitter, Facebook, Reddit) and news outlet provide a real time flow of information directly from the affected people, official agencies, and media organizations [3, 4]. These platforms allow emergency responders to track affected areas and communicate safety instruction to the population. However, the unstructured and high-volume of data stream coming from these platform brings significant challenges in information retrieval (IR) and summarization. Emergency responders

have to select useful information from a pool of high density data, which contains redundant, inaccurate and incomplete facts. These is the primary challenge that needs to be address, in order to be able to provide support to the people affected by the crisis rapidly [5]. Another problem derives from the fragmentation of the information due to the use of multiple platforms. Each sources have their own type of language, for example social media post tend to be short and informal, while news outlet tend to be longer and provide a broader view on the events. These challenges make information retrieval during crisis events a difficult task.

1.2 Information Retrieval in Crisis Scenarios

One of the most used retrieval algorithm is BM25 [6, 7], which is used to assign scores to documents based on the frequency of words present in the corpus of the text that matches the query. This algorithm is widely used due to its efficiency in handling a large quantity of documents without affecting scalability. However this algorithm result to have limitation when dealing with crisis events dataset, making BM25 not suitable as it is, to retrieve useful and non redundant information in this context.

1. Lexical dependency: BM25 performs lexical search, meaning that only exact matches of the words present in the query are evaluated for the final score, this leads to poor recall if the dataset is not filtered from redundant facts.

2. Contextual understanding: Since only exact terms are matched, this leads to the inability of capturing semantic similarity, that is crucial in understanding the context of the facts and can lead to high score assigned to non inherent text.

With the advance of natural language processing (NLP), and the introduction of BERT models [8], which are build upon a transformers architecture and provides substantial improvements in the field of information retrieval. This architecture is built upon the concept of self attention [9] and enables machine learning models to understand the context behind the words used in the sentences, this allows the introduction of dense retrieval and cross-encoder models. The first one allows a more effective semantic search by transforming the data and the query into embedding, which are a vector space representation of the element, however, its more computationally intensive. The second is used to refine the scores assigned by the system by computing a more direct comparison between query and text producing a more accurate scores. This two elements combined together allows to improve precision without sacrificing recall, ensuring a more accurate and efficient information retrieval during crisis.

Our approach tries to integrate the best part of both approaches. We started by filtering the initial dataset, then applying topic modeling, in order to further reduce the dataset and select only topic related documents. Afterwards we applied a double retrieve and re-rank strategy using BM25, to select possible candidates for the final summary followed by dense retrieval to enhance the quality of retrieved facts, and in the end we compose the final summary. With the implementation of this systems, we achieved a more accurate and reliable summary compared to the baseline results, providing a robust and scalable solution for real world crisis information retrieval.

1.3 Thesis structure

The structure of this thesis is organized into chapter, each focusing on a different aspect related to the core objective of our work, the main aspects discussed in each chapter can be summarized as follows:

- Chapter 2: In this section the main objective is to grasp a general outlook conducted in the space of information retrieval (IR), diving in the field of natural language processing (NLP), retrieval systems and the methodologies used to assert their performance.
- Chapter 3: This chapter focuses on providing an overview of the methodology applied in this research in order to solve the required task, including the dataset, the preprocessing technique, and the pipeline employed.
- Chapter 4: This part covers the experimental finding conducted in this research using the previously formalized mehodologies, including a presentation of the experimental setup and a comparison of the obtained results.
- Chapter 5: In this chapter, we present a summarization of the core findings of this research, together with suggestions of possible future works.

Chapter 2

Related Works

In this second chapter we will be able to grasp a general outlook about the inquiry conducted in the space of IR (information retrieval), we will dive into the field of NLP (natural language processing), retrieval system and the methodologies used to assert their performance. Our tasks consisted in retrieving useful data in order to composed a summary that could be used in crisis event cases, a IR system provides the necessary tools in achieving our goal. The information is passed to the system that uses the query to retrieve the most related documents, that are later aggregated in order to produce the required output.

2.1 Natural Language Processing (NLP)

Natural Language Processing NLP is a field of AI (artificial intelligence) that enables computers to understand, interpret, generate, and interact with humans using natural language. It combines linguistic computer science, and machine learning to process and analyze text and speech data. Common tasks were NLP is used in today's world are:

1. Text Classification: The process of categorizing sentences by assigning a label, it is often used to filter documents (e.g. spam filtering)
2. Topic Modeling: The process of generating clusters of similar documents by extracting keywords and generating relation between entities.
3. Text Generation: The process of generating text that is indistinguishable from human writing (e.g. paraphrasing sentences).
4. Part-Of-Speech tagging: The process of identifying entities by labeling each word in a sentence with their corresponding grammar part (e.g. noun, verb, adjective)

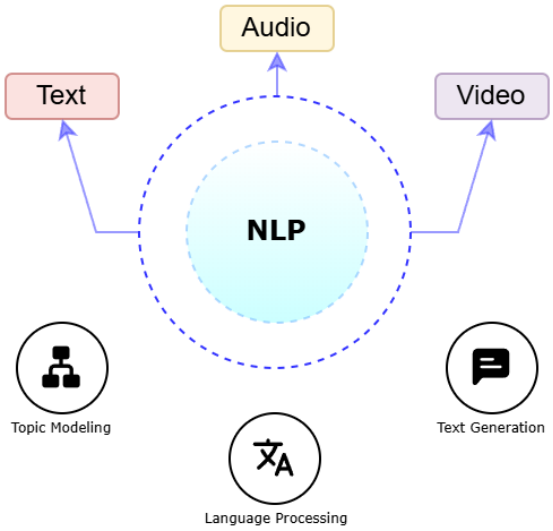

Figure 2.1: Common task in NLP.

5. Sentiment analysis: The process of identifying the emotional tone of a sentences by assigning a label (e.g. positive, negative, neutral).

6. Neural Machine Translation: The process of translating text from one language to another maintaining the original linguistic nuances.

7. Question Answering: The process of answering a question based on a query and a corpus of data.

8. Named entity recognition: The process of identifying and classifying named entities (e.g. locations, persons, organizations) in a sentence in order to extract knowledge graph.

9. Text summarization: The process of generating a concise version of a document maintaining key information. This can be extractive, the summaries are generated by extrapolating text from the corpus of data, or abstractive, new text is generated preserving the meaning or the original data.

In this thesis, the main NLP topic we are going to tackle in are Topic modeling and Text summarization.

2.1.1 Neural Network

NLP finds its roots in the 1950s with Alan Turing and the introduction of the Turing test [10], it received an incredible boost with the advance of the NN (Neural

Network), which are models inspired by the biological functioning of the human brain and are the foundation for modern AI models. The simplest NN model that exist is the perceptron [11], that is composed by a series of neuron connected between them (Fig: 2.2), the algorithm iteratively adjusts the weight vector w using an update rule till the function converges, the equation can be defined as follows:

$$f(x) = h(w \cdot x + b) \tag{2.1}$$

- h: Heaviside step function, the input is mapped to 1 if the value is higher than 0, otherwise is 0.
- $w \cdot x$ is the product between the input values and the weight of the perceptron.
- b is the bias term.

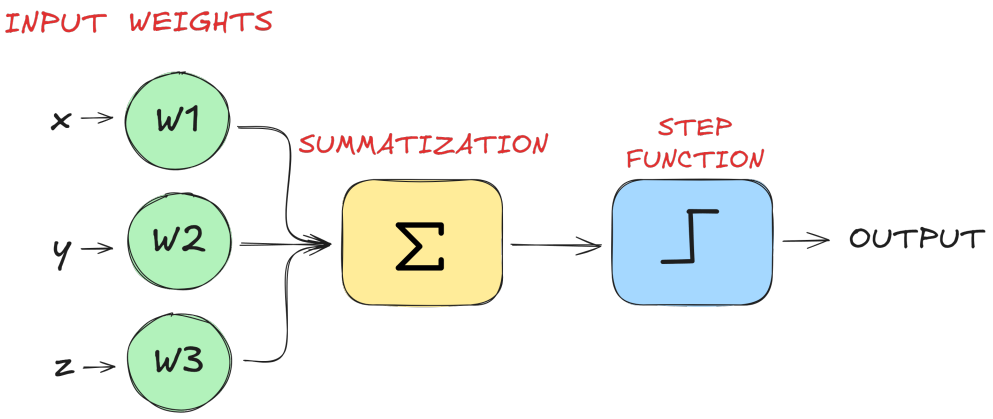

Figure 2.2: Base configuration of a perceptron [11], a linear classifier that takes decision based on a combination of input weights and a step function. It is a binary classifier that decides if an input vector belongs to a determined class.

The next big step was made by Recurrent neural networks (RNNs), which are NN models designed to process sequential data (e.g. text, speech) by feeding back to the network the output of each neuron allowing to capture temporal patterns, and an a consequence allows a form of memory. However they struggle with capturing long range dependencies, the vanishing gradient problem. To deal with this problem Long Short-Term Memory (LTSM) [12] were introduced which solved this problem by integrating gated components in order to regulate the information flow, allowing the use of a longer short memory. Although their success, LTSM models still faced limitation handling very long sequence, for this reason transformers architecture were introduced, addressing the issue by relying on self attention mechanism [9],

which introduce the concept of attention that helps a model understands the meaning of the words based on the context (i.e. "She carried a light bag at the airports" and "The sun provides light during the day", the word "light" although is the same has two different meaning based on the context, for the first phrase it means not heavy, for the second it means illumination).

BERT (Bidirectional encoder representation from transformers)

BERT [8] is a language model built upon transformers architecture (Fig: 2.3) it became a SOTA models in text processing and is a staple baseline in NLP. This model is designed to create pre trained models using only one additional output layer in order to achieve a series of models for each use case. Due to his architecture BERT is able to represent text using embedding, this is useful because it can take into account not just a single word but an entire phrase in order to better understand the context behind the sentence.

While BERT significantly improved text NLP task with the introduction of its architecture, it was inherently limited to task such as classification and text completion, where text in generated by predicting missing token in a sentence. The next upgrade was made by large language models.

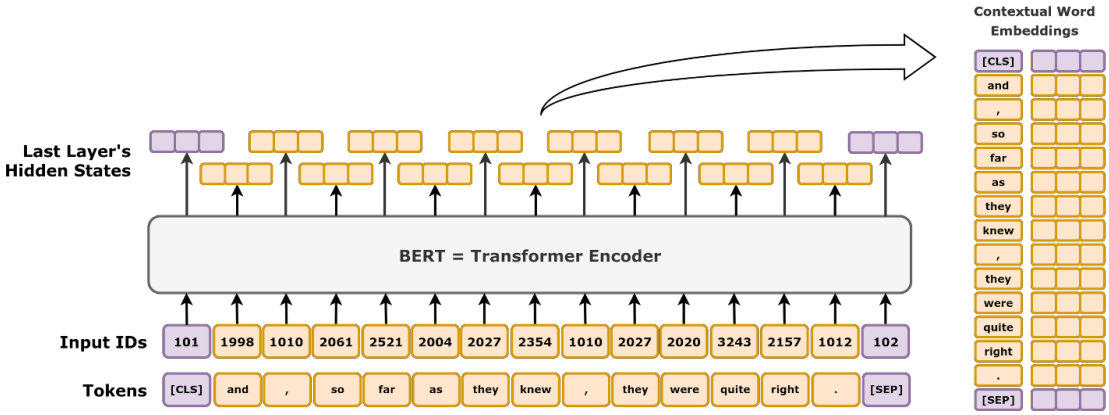

Figure 2.3: The base implementation of BERT is composed by a tokenizer, that converts the input text into a vector containing the words, then an embedding model is used to converts text into number so the model can be used, an encoder that encodes the embedding and a Task head which can be seen as a simple decoder [13].

Large Language Models (LLM)

Large Language models LLM are a class of AI model which are trained to process natural language and are able to generate text from scratch indistinguishable from

human like text. The architecture of this models is based primary on transformers [9]. A significant improvements in the field of LLM was made by OpenAI which introduce GPT-4o [14] in 2024, this models improves upon its predecessor by enhancing efficiency and capabilities making it a powerful tool for NLP tasks, although due to its lack of scalability and because its purpose is generating text this LLM model is not primary used for IR (information retrieval).

2.1.2 Summarization

As previously state text summarization is one of the main task of NLP, it consist in reducing a corpus of text by selecting only the key information present in the collection of data. There are two main types of summarization extractive and abstractive: the first one, consist in extracting key sentences from the original corpus of data and aggregates them without generating new text, the second one generates text by using the key sentences from the original documents, maintaining the original meaning.

When dealing with summaries models in order to asses their performance and understand if it can generate text that can be indistinguishable from a human made summaries there are various metrics that can be used:

- ROUGE [15]: Recall-Oriented Understudy for Gisting Evaluation is based on calculating the syntactic overlap between the candidate summaries and the evaluation one.
- BERTScore [16]: Uses the embedding representation of a text to calculate the semantic distance between the candidate summaries and the evaluation one.
- BLEU [17]: Bilingual Evaluation Understudy is used to compare translation candidates to a reference translation.

Since in this thesis the main topic is text retrieval in order to compose an extractive summary we will make use of ROUGE and BERTScore to evaluate our pipeline.

ROUGE

ROUGE [15] score can be utilize in order to assert the quality of a generated summary comparing it to a reference one created by humans. The computation of rouge score revolves around calculating the n-gram, a sequence of n adjacent word in order, recall between the candidate and reference summaries.

$$ROUGE_N = \frac{\sum_{S \in \{ReferenceSummaries\}} gram_n \in S \sum Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} gram_n \in S \sum Count(gram_n)} \tag{2.2}$$

- n: length of the n-grams
- $gram_n$ $Count_{match}(gram_n)$: max n-gram occurrences in a candidate summary
- $Count_{match}(gram_n)$: set of reference summaries

In this thesis we employed ROUGE-2, where the recall is computed by confronting the bi-gram of the relative summaries.

BERTScore

BERTScore [16] leverages the power of embedding and computes the similarity between two summaries based on the cosine similarity of the candidates (Fig: 2.4). This scoring function makes use of pre-trained BERT models in order to compute the necessary embedding for the evaluation, then computes the semantic similarity, however, due to their innate computational heaviness long summaries require a considerable computing power.

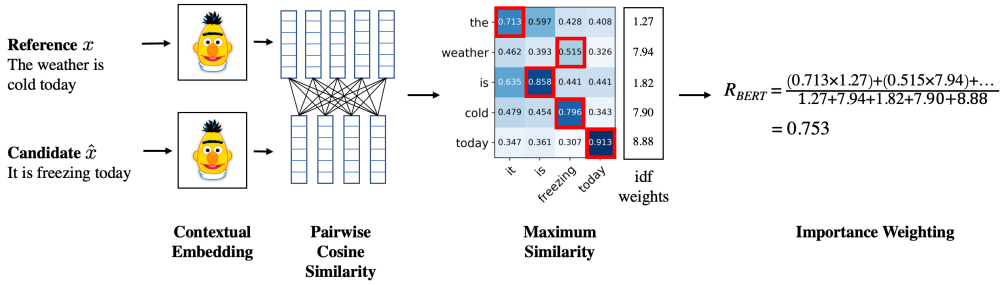

Figure 2.4: Schematic visualization of BERTScore [16] metric: From the reference and candidate summaries contextual embedding are retrieved and pairwise cosine similarity is computed, then the maximum similarity is calculated and the score is computed.

2.2 Information Retrieval (IR) And NLP

In computer science information retrieval refers to the task of finding documents from a corpora of text that responds to an information need, usually in the form of query. This task was first introduced in the 1950s and evolved over time from manual indexing and searching to an automated system. Search engines can be considered a tool for information retrieval, the first one used keywords to find relevant documents, although very useful at the time their capabilities were very limited. Nowadays IR incorporates NLP technique in order to maximize the retrieval step and recover facts that matches lexical keywords and semantic meaning. In our

research we tackle two main type of IR system, the first is BM25 (performs lexical search) and the second is Dense retrieval (performs semantic search)

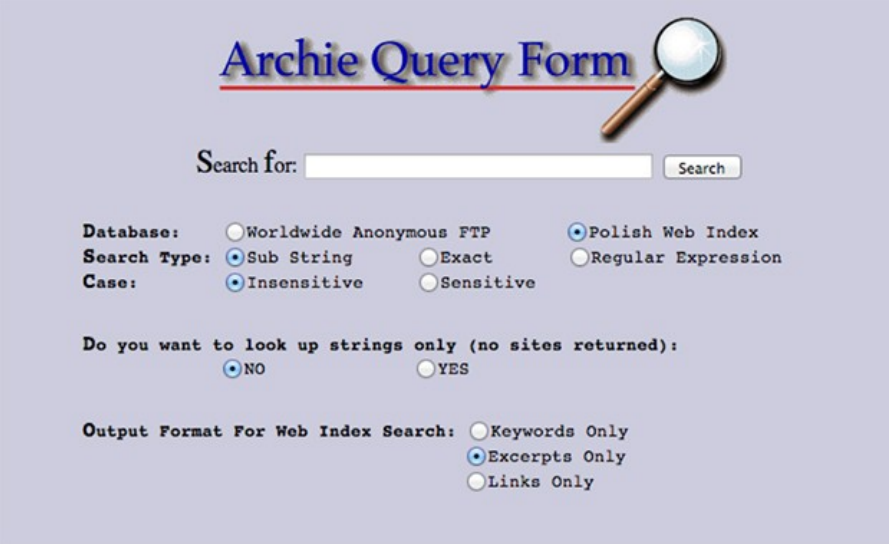

Figure 2.5: Archie, one of the first browser ever existed, query form for finding information [18]

2.3 Text Retrieval

Text Retrieval can be considered as a subcategory of IR where the core task is retrieving text from a set of documents in an unstructured way, then this texts can be used to find the searched information from a smaller collection or to produce summaries. Data is said to be retrieved in an unstructured way because the query produce do not follows exact rules, like in structured data retrieval, the most famous being SQL, where the query needs to follow certain rules in order to retrieve requested entries in the dataset.

2.3.1 BM25

Okapi BM25 [6] is a ranking function developed in the 1970s by Stephen E. Robertson and others. This function is used to perform lexical search on a set of documents, it estimates the relevance of the document to a given query, it is employed in many search engines, like Google or Bing in their retrieval system in order to return the best results. The implementation of this algorithm is based on a probabilistic retrieval framework [7], which is a theoretical model that estimates the probability that a document d_j is relevant to a query q [19]. It assumes that

from a set of documents only a portion of it is relevant to the user, and that part maximizes the overall probability score for the user. This model falls under the probabilistic category of the IR system, where the score each document is assigned during retrieval, based on the query, is computed using the probabilities that the document is relevant to the query.

BM25 ranking function works by classifying each document based on a set of keywords present in the query, this terms are search inside the corpora of each documents and a score, regardless of the proximity within the text of the desired keyword, is given to each document, from then the top results are collected. The most used ranking function can be described as follows.

Given a query Q containing $q_1, ..., q_n$ keywords the score of the document d can be calculated as follows:

$$BM25(d) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i, d) \cdot (k_1 + 1)}{f(q_i, D) + k_i \cdot (1 - b + b \cdot \frac{|d|}{avg(d)})} \tag{2.3}$$

- $f(q_i, d)$ is the frequence of times a keyword q_i is repeated inside the document D.
- $|d|$ indicates the length of the document.
- $avg(d)$ indicates the average length of the document inside the collection .
- k_1 and b are parameters decided by the specific implementation of the equation.
- $IDF(q_i)$ is the inverse document frequency weight of the query term q_i.

IDF indicates the inverse document frequency and is a measure of importance of words inside a collection of documents. Usually is computed as follows:

$$IDF(q_i) = \ln \left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1 \right) \tag{2.4}$$

- N indicates the total number of documents inside the collection.
- $n(q_i)$ indicates the number of times q_i is repeated inside the document.

The correct implementation of the equations (2.3) and (2.4) depends on the used variation of the BM25 algorithm.

Algorithm 1 BM25 Retrieval Algorithm

Input: Query Q, Corpus $D = \{d_1, d_2, \ldots, d_n\}$, Parameters k_1, b
Output: Ranked list of documents

Tokenize Q and D
Compute document length $|d|$ for each $d \in D$
Compute average document length avgdl
for each $d \in D$ **do**
    Initialize $BM25(d) = 0$
    for each term $t \in Q$ **do**
        $f(t, d) \leftarrow$ frequency of t in d
        $IDF(t) \leftarrow \log \frac{N - n(t) + 0.5}{n(t) + 0.5}$,
        $BM25(d) \leftarrow BM25(d) + IDF(t) \cdot \frac{f(t,d) \cdot (k_1 + 1)}{f(t,d) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avg(D)}})}$
    end for
end for
Rank D by $BM25(d)$ in descending order **return** Ranked list of documents

The first step consist in representing the tokenized version of the corpus and the query. Each document is split into token:

```
Query: ['what', 'is', 'the', 'capital', 'of', 'italy', '?']

Doc 1: ['the', 'capital', 'of', 'italy', 'is', 'rome', '.']
Doc 2: ['italy', 'is', 'a', 'beautiful', 'country', '.']
Doc 3: ['i', 'went', 'to', 'italy', 'during', 'the', 'summer', '.']
Doc 4: ['the', 'capital', 'of', 'france', 'is', 'paris', '.']
Doc 5: ['i', 'went', 'to', 'the', 'beach', 'during', 'the', 'summer', '.']
```

Document	**BM25 Score**
Doc 1: The capital of Italy is Rome.	0.9407
Doc 4: The capital of France is Paris.	0.8570
Doc 2: Italy is a beautiful country.	0.1784
Doc 3: I went to Italy during the summer.	0.1575
Doc 5: I went to the beach during the summer.	0.1090

Table 2.1: BM25 Scores for each document sorted by score

As we can see from table (Tab: 2.1) BM25 is very effecting in finding the required information based on the query, the problem arise when in the corpus of

the document there are details that matches the request (the capital of a city in our case) but the answer is not correct, as we can see both doc 1 and 4 have a very high score, while only doc 1 answers directly to the query.

2.3.2 Dense Retrival

While BM25 algorithm is very powerful and fast has its limits because it only performs lexical search, this means that it is not able to understand the context and the semantics behinds certain words. To overcome this issue we can make use of dense retrieval [20] to compute semantic similarity and retrieve more related facts to the query. To understand real world objects computers makes use of embedding, that can be defined as a mathematical representation of the information as a high dimensional vectors in a continuous space.

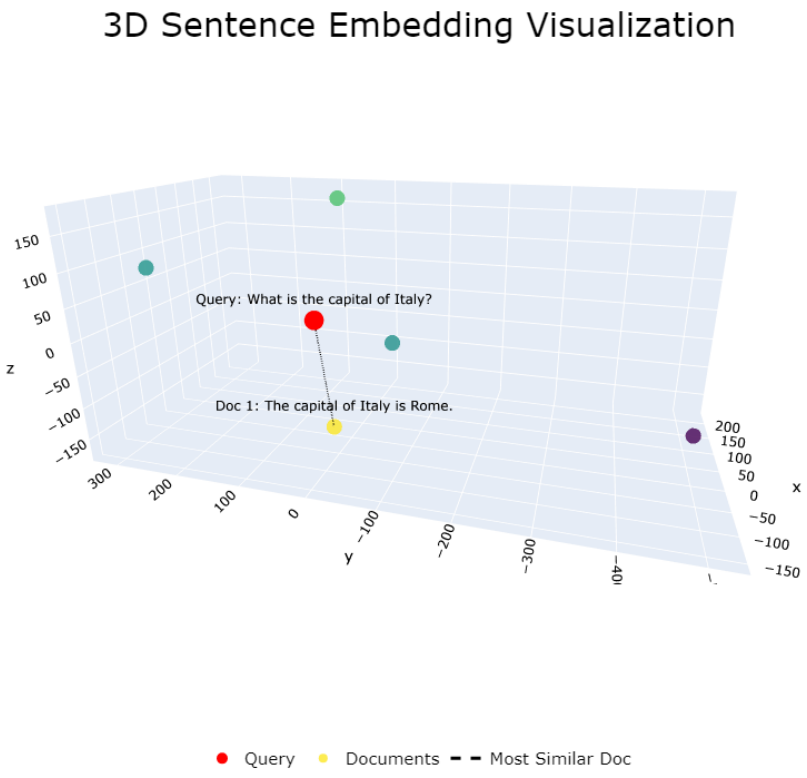

Figure 2.6: 3D plot visualization of embedding for the query and a set of document. The red dot represent the query, while the other points represent the doc with colored based on their cosine similarity to the query (lighter color indicates higher scores).

Since real world object can be represented as vectors, computing their similarity

means calculating the distance between them. In order to perform the retrieval we can compute the distance between the query and the document of the collection from which to find the required information, to achieve our goal we use cosine similarity to perform distance between two non-zero vectors.

Given two n-dimensional vectors Q and D, the cosine similarity between them is defined as following:

$$score(Q, D) := \cos\theta = \frac{\boldsymbol{Q} \cdot \boldsymbol{D}}{\|\boldsymbol{Q}\| \cdot \|\boldsymbol{D}\|} = \frac{\sum_{i=1}^{n} Q_i D_i}{\sqrt{\sum_{i=1}^{n} Q_i^2} \cdot \sqrt{\sum_{i=1}^{n} D_i^2}} \tag{2.5}$$

- Q and D represent the term frequencies vectors of the documents for which the distance is computed.

The dense retrieval algorithm makes use of the cosine similarity to compute the distance between the query and the corpora of the text giving a score which can be normalize between 0 and 1 to each document, we can see an implementation of the algorithm in the following section:

Algorithm 2 Dense Retrieval Algorithm

Input: Query Q, Corpus $D = \{d_1, d_2, \ldots, d_n\}$, Encoder model E
Output: Ranked list of documents

1: Encode query Q as vector q: $q \leftarrow E(Q)$
2: Encode documents D as vectors: $v_d \leftarrow E(d), \forall d \in D$
3: Initialize similarity scores $S = []$
4: **for** each $v_d \in D$ **do**
5: Compute similarity $S_d = \frac{q \cdot v_d}{\|q\| \cdot \|v_d\|}$ (cosine similarity)
6: Append S_d to S
7: **end for**
8: Rank D by S_d in descending order **return** Ranked list of documents

As we can see from table (Tab: 2.2) using the same queries and documents used for BM25 the dense retrieval algorithm is able to better understand the context of the sentence, given a higher score to document related to the query and not based on the lexical similarity. However, this improved semantic understanding comes at a higher computational requirements.

Encoders

Embeddings are the core part of semantic similarity and serves as a dense vector representation of text, they can be computed in various ways depending on the

Document	Dense Score
Query: What is the capital of Italy?	1.0000
Doc 1: The capital of Italy is Rome.	0.8728
Doc 2: Italy is a beautiful country.	0.6344
Doc 3: I went to Italy during the summer.	0.4763
Doc 4: The capital of France is Paris.	0.4649
Doc 5: I went to the beach during the summer.	0.0498

Table 2.2: Dense Retrieval Scores for each document and query sorted by score

retrieval approach. In the context of IR, encoders are designed to be efficient for this reason they compute embeddings for queries and documents separately and combines them into a shared vector space, afterward cosine similarity can be employed in order to perform retrieval. However, since the encoding process is performed separately for queries and facts, encoders may struggle with fine-grained relevance matching. It is often use as the first step in the process of IR to filter down the number of document to analyze.

Cross-Encoder

Cross-Encoders in the context of IR, uses deeper transformers models to compute embedding and produce a more accurate score. Differently from encoders which computes embedding independently, cross-encoders performs pairwise embedding computation in order to have a better understanding of the context from the query and the document. They obtain excellent performance in re-ranking tasks, with a small number of candidates. It is often use as the second step in the process of IR due to its high precision.

2.4 Temporal Summarization

When dealing with on-going events such as protest, accidents or natural disaster the available relevant information is proportional to the time passed since the event. For example immediately after the amount of relevant facts available to assert the damage will be low, highly inaccurate and redundant. But in order to provide timely assistance in an emergency case a disaster manager would need to have precise information in the less time possible.

TREC (Text REtrieval Conference) proposed several tracks dealing with Temporal Summarization (Fig: 2.7). Differently from classical summarization where all the context is present from the beginning and the summary is composed by

selecting key facts in the corpus of data, temporal summarization systems need to select information as they are release on the internet. The goal of this approach is twofold, (1) to monitor efficiently the stream of data and select useful and non redundant information and (2) to produce a comprehensive summary that can support decision making in emergencies cases as they are developing. In this thesis, the main objective is temporal summarization across multiple streams of data, we tried to developed a system that efficiently selected relevant information in an evolving environment. The formalization of our approach will be developed in the methodologies chapter.

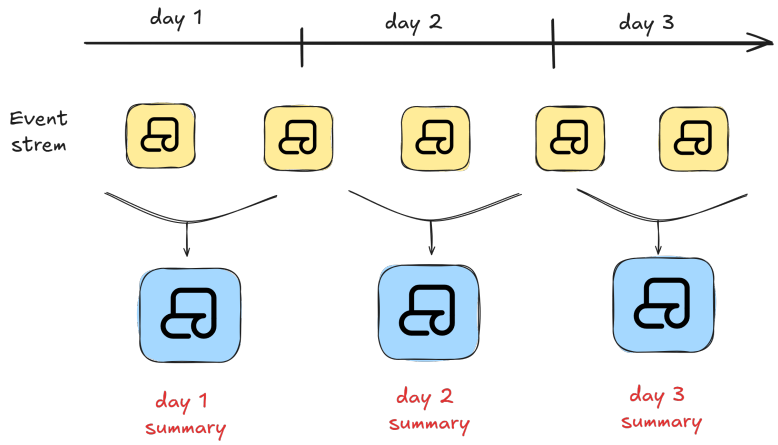

Figure 2.7: Visual representation of IR over time: the information for an event is collected from a stream of data and for each day a summary is composed.

Some of the tracks proposed by TREC are:

- Temporal Summarization 2013 [21]: This track main topic is temporal summarization in the context of an event developing over time.
- Real-time Summarization 2017 [22]: This track main topic is temporal summarization by monitoring social media posts.
- Incident Streams 2020 [23]: This track main topic is temporal summarization over an abundant quantity of data.
- CrisisFACTs 2023 [24]: This track main topic is temporal summarization for emergencies case over a multi stream of data. This is the main track on which this thesis is developed.

Chapter 3

Methodology

In this chapter we are going to provide an overview of the dataset used for the research, how it was obtained and all the preprocessing applied to it, and describe the pipeline used to achieve the final results for our task.

3.1 Problem Statement

The main topic addressed in this thesis focuses on fact extraction, where the system consumes data coming from a dataset for a given disaster, split in a day-event pairs. For each event there are a set of query, defining the information needed to achieve a useful and non redundant summary of the disaster event.

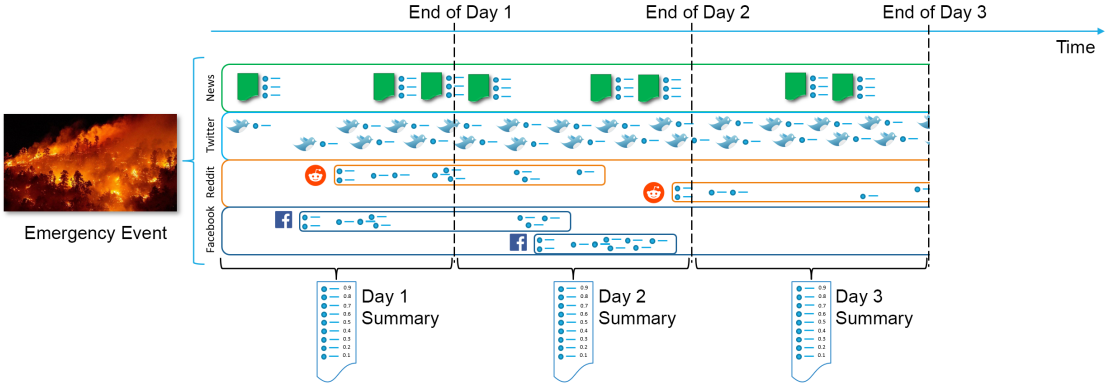

Figure 3.1: High-Level System Overview of the task: information are retrieved from a stream of data which are taken from different sources which are news outlets, Twitter post, Reddit threads and Facebook posts, for each day a summary of the event composed with useful information is produced and ultimately all facts are aggregated to compose the final summary for the event in analysis.

Given a set of events $E = e_1, ..., e_n$ about different hazards, where each event e_i last at least two days within the reference time period T and contains a set of documents stream $D = d_1, ..., d_n$ collected from different sources such as Tweeter, Facebook, Reddit and news outlets. Each document d_i arrives at a timestamp t_i where $d_i \in D$ and $t_i \in T$. For each event e_i a set of query $Q = q_1, ..., q_n$ is defined, representing the key information to emergency responders (e.g. affected areas, number of casualties, infrastructure damage). Given E, D and Q, the objective is to generate a temporal summary $S = s_1, ...ms_n$ where s_i represent the selected documents per day. This project will explore efficient retrieval methods to construct S while addressing challenges such as information overload and redundancy, the proposed pipeline will be further formalize in this chapter.

3.1.1 Framework Overview

To perform the assigned task we employed different components into our main pipeline taking advantage of some techniques like clustering and lexical and dense retrieval:

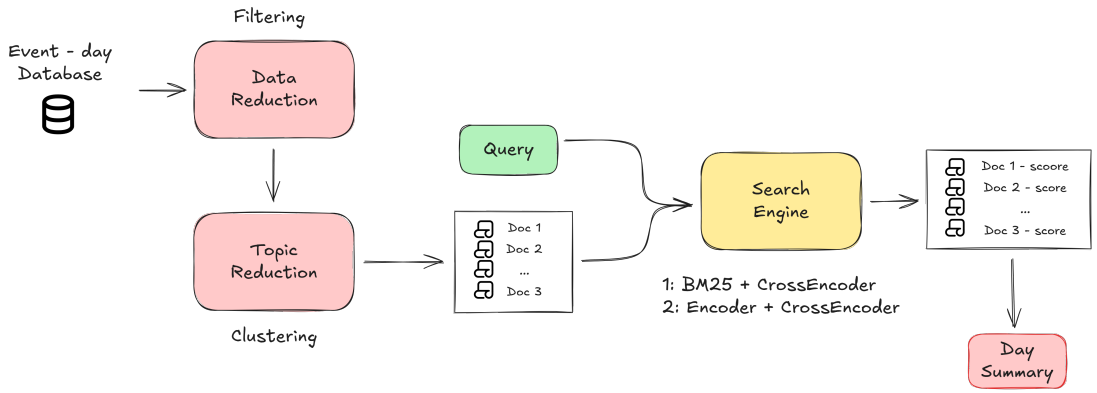

Figure 3.2: Visual Representation of the implemented pipeline.

1. Initial Reduction: Since the dataset is filled with a lot of non inherent facts, the first step consist in reducing the number of facts in order to maintain useful and non redundant data.

2. Topic Modeling: We employed BERTopic [25], which is a powerful tool for creating cluster of data in order to select only the collection of data related to the event in analysis. We also tried using Top2Vec [26] in order to create contextual clustering but decided to exploit BERTopic due to its modularity, that allows to select the best algorithm for the use case at each step.

3. Retrieval step: This step of the pipeline is composed of a double retrieval passage where we employed RR (Retrieve & Re-Rank) algorithm with BM25, for lexical retrieval followed by another instance of RR with a BERT [8] encoder for dense retrieval.

4. Summary: The last step consist in collecting all the retrieval data for each day and assemble it into a single summary for the event.

3.2 Dataset

The dataset is composed by a series of documents that comes from different sources, for each day during an event this information was produced and deposited on the internet, where was collected and composed by CrisisFACTS [27]. The data present in the collection has not been filtered, for this reason there is a lot of noise present inside of it. For each day, during an event, the following content is available:

1. Twitter stream: Twitter posts were collected based on keywords relevant to the tragedy events in analysis.

2. Reddit stream: From Reddit threads relevant to the emergency top-level posts with all subsequent comments were extracted and included into the dataset.

3. Facebook stream: Facebook/Meta posts are provided based on relevance to each disaster events from public pages.

4. News stream: News article are an excellent source of information during catastrophic events, a small numbers of pieces was included in the dataset.

Query defining the needed information are provided with the dataset, and they are extracted from the FEMA ICS 209 forms [28]. These queries should be integrated with the IR pipeline to give relevant information to the stakeholders and filter out redundant and non useful information that is produced during these events and are not essential to responders or disaster-response manager. These queries capture what a responder might consider important, such as [27]:

- Risks from hazardous materials
- Damage to key infrastructure, evacuations, or emerging threats
- Statistics on casualties or numbers missing
- Weather concerns
- Restriction on or availability of resources

The eighteen events, eight composed in 2022 and ten in 2023 (Fig: 3.3), covers different type of events such as wildfires, hurricanes, floods and accidents. Each entry in the dataset is characterized by the following information (Tab: 3.1) retrieval pipeline will leverage mainly text field in order to set a score for each document and the selection is based on the top most relevant facts.

Field	Value
Event	CrisisFACTS-001
Stream ID	CrisisFACTS-001-Twitter-13116-0
Unix Timestamp	1512677202752
Text	Mandatory evacuations west of 395 including SUllivan Middle School due to #LilacFire near I-15 and SR-76
Source Type	Twitter

Table 3.1: Sample Social Media Post

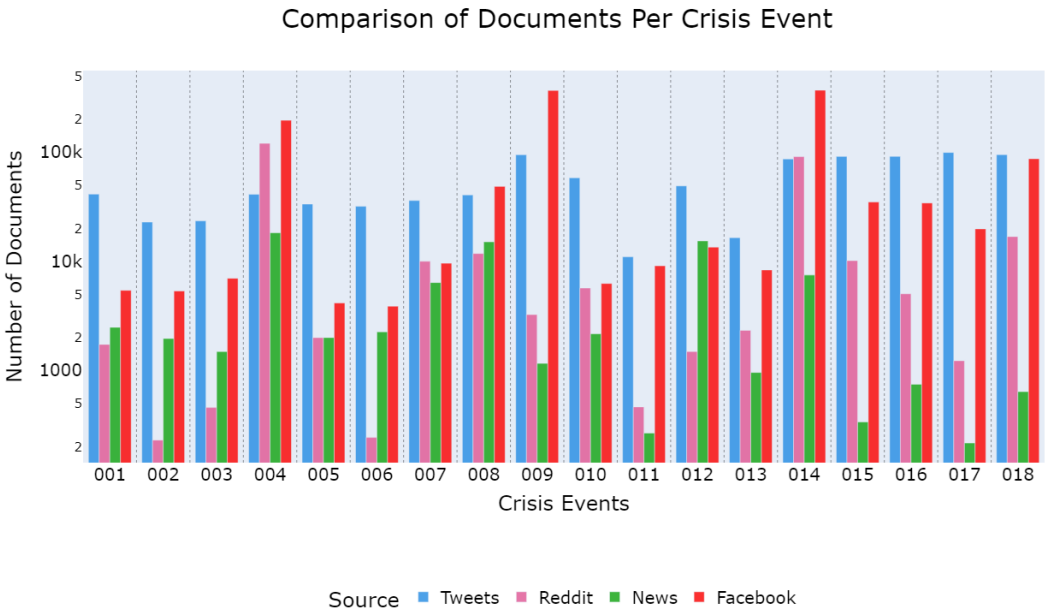

Figure 3.3: Visual representation of data distribution for the event present in the 2023 TREC CrisisFACTS Track

3.3 Preprocessing and Data Cleaning

In this section we will illustrate all the preprocessing methods and how data was cleaned to remove redundant data and noisy (not related to the event) documents.

Part of the work was dedicated to reduce the dataset, since the quantity of data is very large (i.e. event 014 - Hurricane Dorian, 2019 - has over 500k documents) and a lot of it can be considered noise. The first part of our job was to reduce the total number of documents that would be passed to the retrieval pipeline, we considered data to be irrelevant if the context was out of relevance, or if the information was already present in the collection or if the fact was not crucial for assessing the damage for the crisis in analysis.

3.3.1 Initial Data Filtering

To better reduce the dataset we started by sanitizing the text in each document, since a lot of the data comes from social networks, the use of "#", "@", emojis and links is prevalent in a lot of texts, we removed those characters, as well as extra spaces to better perform the initial reduction and to create more coherent embedding using transformers model for the retrieval pipeline. Sanitizing the text also made similar text, that only differs in different links, or emojis and extra spaces equals to each other. The initial filter was done by removing:

- Duplicated documents.
- Short text (less than two words per documents).
- Documents that contains question or generally if contains the "?" character.

Short text documents were removed because in most cases (i.e. event 004 text: "Following... https:...") they didn't provide any important information or contained only the title of the event which is not useful at all for a crisis agent that needs information to assert the damage done during a catastrophic event. Documents that contains question mark are also not useful in a context were we need actual information, it is also true that having a "?" symbol in the text does not mean for sure that the text is a question, but running all the data through a intent model, which would be able to detect the intention behind a phrase, takes time, and in a context like crisis event where the response time is crucial to speed up computation we decided to remove all documents that contains the question character. After the initial data cleaning we removed redundant data present in the dataset, in order to not occupy space for other useful fact that may be left behind by the retrieval pipeline since this works by selecting the top k (based on metrics that depends on the model) most useful facts.

3.3.2 Query Reformulation

One of the most important thing when working with retrieval are the queries, if they are not formulated correctly, and do not contain the desire keyword we can

end up with documents that are not the one we wanted, furthermore having the query too precise could lead to a loss of important informations, since we could lose documents that do not correspond to the specific query but are needed to assest the damages during an crisis event.

As we said before the dataset is supplemented with queries defining the basic information needed for a disaster-response stakeholder, and is extracted from the FEMA ICS 209 forms [28]. During our evaluation trials we noted that queries needed to be reformulated in order to add more context to them. Using a language model (GPT 4.o [14]) we asked this model to reformulate the query by adding more context to them, and selected the most fitting one that we believed added more context to the query, in this way we end up with the queries being like this:

- "Have airports closed": "Are airports closed or damaged due to Hurricane?"
- "Have railways closed": "Are railways disrupted or closed because of Hurricane?"
- "How many people are trapped": "How many people are trapped because of Hurricane? Include rescue details.",
- "Are helicopters available": "Are helicopters available for emergency response during Hurricane?",

Another challenge related to the queries were their inability of covering all aspects of the crisis event. To address this, we generated multiple candidate questions, selected the most relevant ones, and expanded the query collection accordingly. The additional queries covered both general crisis-related issues and event-specific aspects. In total, we incorporated:

- 20 general queries addressing broad concern common to crisis situations.
- 10 event specific queries related to the characteristic of each particular event.

Thus, for each event, a total of thirty new queries were added, ensuring a more comprehensive and context aware retrieval process. As an example of general queries we have:

- "Are there any security risks or criminal activities occurring due to Hurricane?"
- "Are ATMs and banks functioning after Hurricane?",
- "What medical services are available for affected individuals?",

As for event specific query we have:

- Tornado: "Are there any reports of multiple tornadoes forming during Tornado?"
- Wildfire: "What weather conditions are affecting the wildfire's spread?"
- Hurricane: "How far inland is flooding occurring due to Hurricane?"
- Flood: "Are any dams at risk of overflowing or failure due to Flood?",
- Storm: "How long is the storm expected to last?"
- Accident: "Has the air quality been affected by Accident?"

3.4 BERTopic

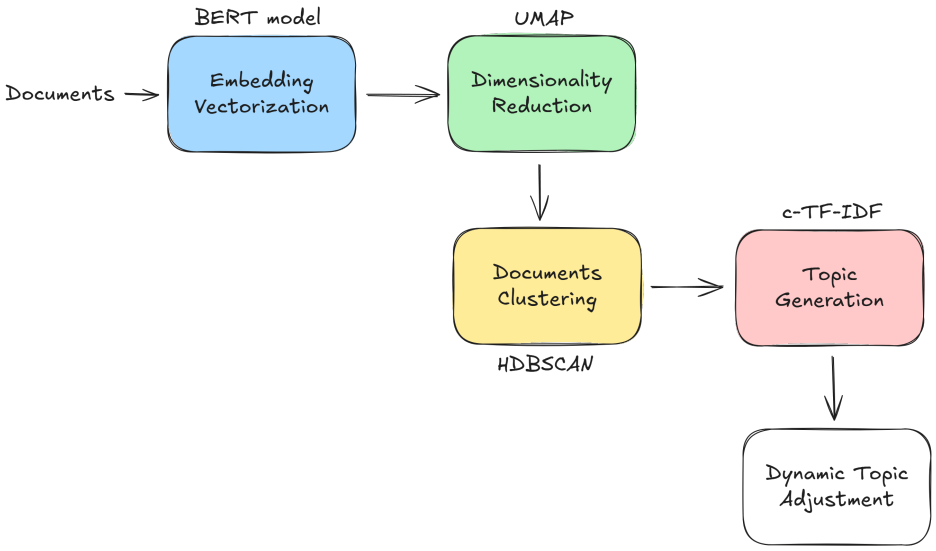

Figure 3.4: BERTopic base implementation: The document are vectorized with a BERT model in order to create the embeddings, thus the dimensionality reduction algorithm is applied and after the documents are clustered using a clustering algorithm, furthermore topic are created and lastly adjustment are made based on the requirements.

BERTopic [25] is a useful topic modeling algorithm that allows clustering the dataset into categories of similar meaning data using an embedding model and c-TF-IDF, a class-based variation of TF-IDF [29], which is a measure of the importance of a word taking into consideration the frequency of the various terms present in the corpus of the collection of documents. This allows for easily interpretable topic

modeling keeping important words into the topic description making it more simple to verify their relevance to the event in analysis. The algorithm can be divided into different independent step (Fig: 3.4), also allowing for modularity, in this way for each the best algorithm can be selected. Firstly each document is converted to its embedding representation, then their dimensionality is reduced to optimize the clustering process, and lastly from the cluster the topic representation is extracted using c-TF-IDF. Afterwards each topic can be processed in the desired way.

To create better topic representation we fine-tuned our model using the following algorithm offered in the BERTopic implementation:

- KeyBERTInspired [30]: Creates a set of representative documents per topic and uses those as our updated topic embedding, in this way the model computes similarity between candidate and topic embedding using the same model used for the documents, it is modeled after KeyBERT [31] algorithm.
- PartOfSpeech [32]: Creates a subset of keywords and documents that best represent a topic, removing stop words, that are common used words (i.e. "a", "the", "is", "are") and can overshadow the relevant information that best describe the topic.
- Maximal Marginal Relevance [33]: Increase diversity between selected keywords in order to better represent all the subset of documents clustered togheter.
- UMAP [34]: Uniform Manifold Approximation and Projection (UMAP) is used to performs non-linear dimensionality reduction on the dataset, it results to be very useful when working with a large collection due to its fast computational speed.
- HDBSCAN [35]: Hierarchical Density-Based Spatial Clustering of Applications with Noise is an extension of DBSCAN [36] algorithm which creates clusters by aggregating points that are close together (high density) and marks points that are in a low density region as outliers. HDBSCAN improves DBSCAN by making it into hierarchical, allowing for clustering with different density.
- CountVectorizer [37]: Vectorizer model that converts text documents into a numerical matrix and handles the task of removing stop words during the c-TF-IDF step.

We leveraged the power of BERTopic modeling to select documents based on their relevance to the crisis event in analysis. To achieve our goal, we used the event description present in the dataset (although it just needs to be a general description of the event in analysis), and extracted key words, from there we computed dense similarity between these keywords and the topic words produced by the topic model.

Then we selected the most relevant topic, we decided, based on various test, to select a topics only if its similarity score is higher than the q-th percentile of all the score computed for all topics in the event-day dataset, but also if its higher than a predetermined threhsold. With the help of BERTopic we further filtered our data removing non inherent to the crisis event data.

3.5 Retrieval Pipeline

In this section we will illustrate how the main retrieval pipeline works, to do so we first have to introduce a key part to retrieval, SentenceTransformers [38]. This library gives easy access to state of the art pre trained models, that we used since it makes available to us simple and efficient ways for computing embedding and semantic search.

3.5.1 Retrieve & Re-Rank

Retrieve & Re-Rank (RR) [39] was the main component of our pipeline and helped us achieve our goal of finding useful information for crisis event. This algorithm leverages two main key component a retrieval and a re-ranker. The first is used to retrieve the most relevant documents based on the query and returns the top-k most relevant results, this can be either a dense retrieval (retrieves data based on embedding similarity) or a lexical one. Afterwards the selected facts are passed to the second component, a cross-encoder, also called re-ranker, that for each query-document pair outputs a relevance score that can be used as an indication of how well the retrieved fact answers the query. The usefulness of this approach stands from the fact that when working with a large dataset using only a retrieval, that needs to be efficient, could lead to retrieve irrelevant documents. To overcome this issue the first component is paired with a re-ranker, that has to work with fewer documents and can produce a more accurate score for each query-document pair, further for this reason the number of retrieved documents from the first component should be limited to a small amount. In our case, we worked with a large dataset and had to retrieve a limited amount of documents to pass to the re-ranker for this reason the initial data filtering based on duplication was essential since with too many similar facts we would have restrain the top selected documents to a lesser number and lose key information.

The usefulness of this approach stands from the fact that when working with a large dataset using only a retrieval, that needs to be efficient, could lead to retrieve irrelevant documents. To overcome this issue the first component is paired with a re-ranker, that has to work with fewer documents and can produce a more accurate score for each query-document pair, further for this reason the number of retrieved documents from the first component should be limited to a small amount. In

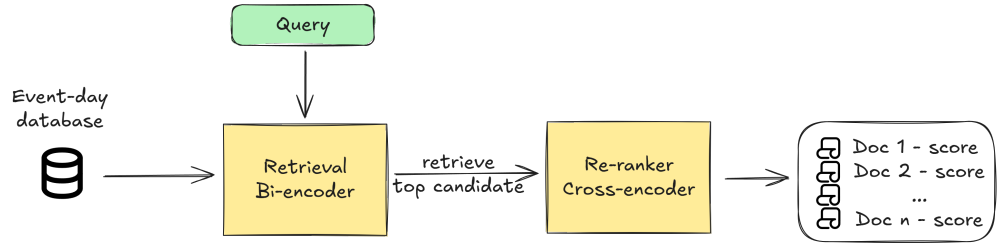

Figure 3.5: RR overview: the collection of data is passed with the query to the retrieval that can be lexical (BM25 or dense which retrieves the top candidate facts for the query, afterwords they are passed to the re-ranker that has to deal with less documents and can be more precise in order to retrieve the most useful facts based on the query.)

our case, we worked with a large dataset and had to retrieve a limited amount of documents to pass to the re-ranker for this reason the initial data filtering based on duplication was essential since with too many similar facts we would have restrain the top selected documents to a lesser number and lose key information.

3.5.2 Double Step Retrieve and Rerank

The selection of the right model was crucial in obtaining a good results in retrieve useful facts. We tested our pipeline with a lexical search (BM25) and an encoder one. Since a major part of our work was dedicated in reducing the dataset to contain only documents related to the crisis event we opted for a double RR in our system, with the intention of further reducing the data. After the initial data filtering and topic modeling, we still had a lot of documents remaining containing a lot of noise, for this reason we decided to use BM25 with re-rank solution, which is very fast and efficient and retriever a set of inherent documents, also when a document responded to multiple query we saved the highest results, since this would meant that the selected document better responded to the new query instead of the previous one. The problem was the we still had a consistent number of documents and a fair amount of noise in our reduced dataset, for this reason we applied RR a second time, however this time with a small amount of documents we could use a stronger retrieval and we opted for a encoder one. After the data was selected to further reduce documents and achieve a summary that can be easily consumed by a human we removed all documents with a score lower than a defined threshold, and limited our selection to the small number of documents per day. With the remaining data we produced the final summary.

To better represent our selection pipeline and summarize it, we can analyze the algorithm [3]. We started with a considerable amount of data, to facilitate the

Algorithm 3 Crisis Event Data Processing Pipeline

```
for each event event_no in event_list do
    Load crisis dataset for event_no
    for each day day in event data do
        Preprocess text and prepare queries
        ▷ Initial Filtering and Topic Modeling
        Remove short texts, duplicates, and questions
        Assign topics using topic model
        Select relevant topics based on queries
        ▷ First Document Selection
        Apply BM25 and cross-encoder
        Rank and filter top-scoring documents
        ▷ Final Document Selection
        Apply sentence retriever and cross-encoder
        Rank and filter top-scoring documents
    end for
end for
```

retrieval step we reduced it by preprocessing the text field removing unnecessary elements. Then we moved to topic modeling, each element in the filtered data was assigned to a topic, and based on the topic relevance to the crisis event in analysis we selected only the most useful topics. After we applied RR using BM25 for the retrieval step and further reduce our collection of data. However since the amount of data was still large, we applied a second instance of RR where this time the retrieval step was done by a dense model.

Summary Composition

When working on this thesis we followed CrisisFACTS Track request which was to produce a list of facts containing the following information (Tab: 3.2). Each fact needs to paired with an importance score representing the usefulness of the fact in analysis, the scoring function we employed was based on the relevance the RR algorithm assigned to the fact during its second iteration using the dense retrieval, which do not take into consideration the source of the fact. We analyze the composition of each dataset and decided to boost scores based on the type of source each fact belong, we prioritize Twitter and Facebook scores, as these platforms facilitate faster information flow, so since the object of this thesis revolved around being able to retrieve core information in low time we boosted the RR score of this sources. Lastly we composed the final summary by grouping facts based on their relative query and selected the most useful information, which were

the ones with the highest score assigned by the retrieval pipeline. The evaluation consisted in comparing the aggregation of our selected facts with the information that assessor from NIST deemed to be core for the event in analysis, the main metrics use were ROUGE and BERTScore.

Field	**Value**
Request ID	CrisisFACTS-001-r3
Fact Text	Increased threat of wind damage in the San Diego area.
Unix Timestamp	1512604876
Importance	0.71
Sources	CrisisFACTS-001-Twitter-14023-0
Stream ID	Null
Information Needs	CrisisFACTS-General-q015

Table 3.2: Crisis Fact Information

Chapter 4

Experiments

In this chapter we are going to present the results of the experiments conducted in our research using the previously presented methodologies. We begin by presenting the experimental setup, followed by an in-depth analysis of the obtained results. Afterwards we are going to discuss the implications of our findings, comparing them to baseline approaches and highlighting key observation. Finally, we conclude with an evaluation of the experiment's strengths and limitation.

4.1 Settings

The setup used to perform all the experiments was the following: NVIDIA RTX 3060 with 12 GB of RAM, an AMD RYZEN 7 5800X on a Windows 11 machine with 32 GB of RAM utilizing Pytorch [40], SentenceTransformers [41] and HuggingFace [42] library in order to perform the various step of our pipeline and achieve our goal of retrieving useful facts from an enormous corpus of data.

4.1.1 Models

We employed HuggingFace [42], which is a python library that provides access to a vast collection of pre-trained models for NLP tasks, in order to access the trained models we used in our pipeline. This models allowed us to benefit from transfer learning, reducing computational cost and enhancing performance across multiple steps in our pipeline. To ensure optimal results at each stage of our research, we selected and employed different models based on the specific tasks. Below we provide an overview of the models used in our pipeline.

Sentence Embedding Models

For representing documents as vector, we utilize CrisisTransformers CT-mBERT-SE model [43], which is a fine-tuned and multilingual version of BERT [8], optimized for crisis-related texts. We chose this model due to its ability to capture contextual semantics related to crisis events in an effective way, which was crucial in order to perform clustering on a large corpus of data. We mainly employed this model into our BERTopic [25] implementation in order to compute the necessary embeddings to perform dimensionality reduction on the data corpus.

Retrieval Models

SentenceTransformers [41] is a python library that provides simple access to a variety of functionalities, including sentence embedding, semantic similarity computation, and cross-encoder models for pairwise ranking.

In order to efficiently retrieve relevant documents, we employed a two stage retriever approach based on RR. The first stage utilized BM25, which was implemented using the rank_bm25 library [44], that provides a strong lexical baseline by ranking documents based on term frequency and inverse document frequency (TF-IDF). The second stage leveraged a dense retriever to refine the ranking, focusing on semantic relevance to improve precision over purely lexical methods.

We experimented with different dense retrieval models, each designed to optimize performance at different stages in our pipeline. RR algorithm requires a cross-encoder model at the end of each retrieval step, for this reason we exploited different models and choose the best based on a combination of efficiency and accuracy. Following there is a list of the employed dense models in our pipeline.

- BAAI/bge-m3 [45] from Beijing Academy of Artificial Intelligence (BAAI) [46] this model was used as a dense retriever for first-pass retrieval, leveraging multi-query capabilities to enhance document ranking.

- mixedbread-ai/mxbai-rerank-xsmall-v1 [47] from MixedbreadAI [48] a light and efficient re-ranker model employed in the second stage of the RR algorithm that improves retrieval through pairwise ranking.

The combination of lexical (BM25 and re-ranker) and semantical search (encoder and re-ranker) allowed us to balance efficiency and accuracy utilizing lexical search on the initial corpus of data, which contained a large quantity of data, in order to maintain high recall, and semantic search on a less large corpus of data in order to refine precision.

Evaluation Models

As mentioned in the methodologies chapter (3), we employed BERTScore [16] to evaluate the quality of retrieved summaries. BERTScore compares token embeddings between a submitted and a reference summary, making it a strong alternative to traditional n-gram-based metrics. Since BERTScore relies on contextual embeddings, it requires a pre-trained transformers model in order to encode the text. We mainly utilize deberta-xlarge-mnli [49] from Microsoft in order to maintain consistency with CrisisFACTS evaluation methodology. However due to its high computational requirements we encountered challenges when processing long summaries, and had to downsize the summary. To address this issue, we also employed distilbert-base-uncased [50] from DistilBERT community model, which is a lighter model and permitted us to evaluate the entire submitted summary without the needing to truncate or downsize the reference and submitted summaries.

With these models integrated in our pipeline we aimed to balance efficiency, accuracy and interpretability in retrieving the most useful facts to achieve a reliable summary for each crisis event in analysis. In the following table (Tab: 4.1) we can see a recap of the model described previously.

Category	**Model**	**Purpose**
Sentence Embeddings	CT-mBERT-SE [43]	Text Embeddings for Clustering
Retrieval Models	BM25 [44] bge-m3 [45] mxbai-rerank-xsmall-v1 [47]	Lexical Retrieval Dense Retrieval Re-ranking
Evaluation Models	deberta-xlarge-mnli [49] distilbert-base-uncased [50]	BERTScore Evaluation Alternative BERTScore Evaluation

Table 4.1: Comparison of Models Used in the Pipeline

4.2 Filter setting

As stated in the methodologies chapter the first part of our work consisted in reducing the amount of data by applying a filter in order to remove low-quality or duplicated documents, ensuring a more refined corpus of data for the topic modeling and retrieval part. The figure (Fig: 4.1) illustrates the effect of this filtering across different crisis events, comparing the total number of documents before and after reduction in a logarithmic scale. While the reduction varies across events, the overall trend shows a decrease in the total corpus of data, with many

events retaining a substantial portion of the original data. This step helps improve retrieval efficiency by focusing on more relevant documents.

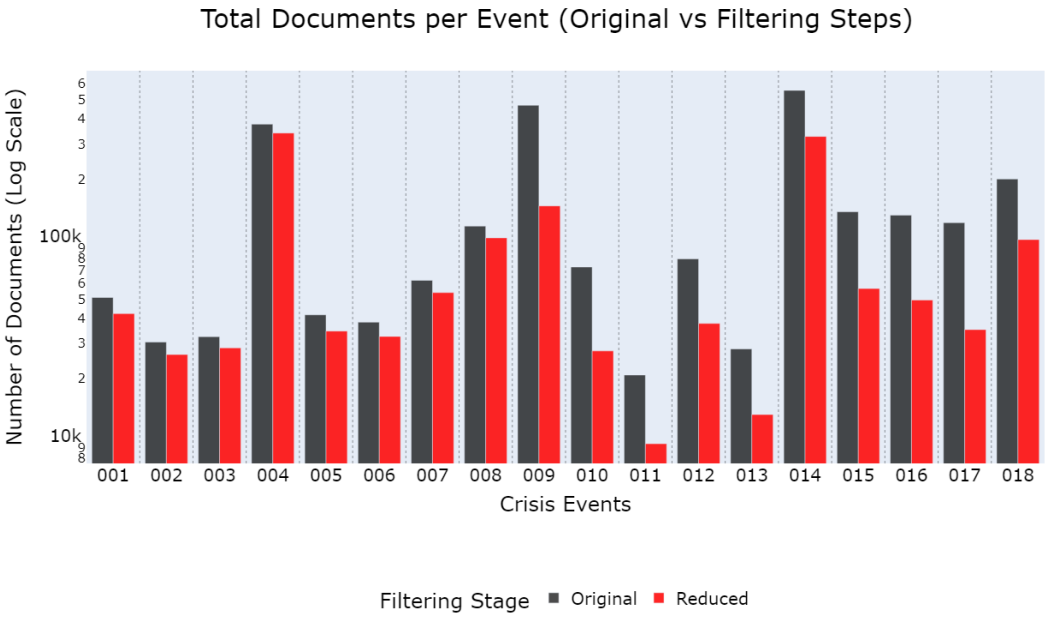

Figure 4.1: Data distribution before and after reduction in a logarithmic scale.

Metric	Original Dataset	After Filtering
F1-score	0.0132	0.0192
Precision	0.0067	0.0098
Recall	0.9321	0.9215

Table 4.2: Mean ROUGE-2 scores across all events before and after filtering

To further evaluate the initial filtering, we computed the ROUGE-2 score [15], we collected all the sentences present in the initial dataset and computed the score, we expected a recall value, which indicate the true positive sentences selected that are present in the target summary, of 1.0, since we obtained a lower value we supposed that the target summary does not contain all the bi-gram words present in the entire dataset. Table (Tab: 4.2) presents the ROUGE-2 scores before and after filtering. While recall remains high (from 0.9321 to 0.9215), indicating that most of the important information is preserved, both f1 and precision scores improve. This suggest that the filtering process successfully removed some non relevant information, making the next step in the pipeline, the topic modeling more efficient by having to deal with less data.

4.3 BERTopic settings

As we previously mentioned we employed BERTopic [25] in order to extract meaningful topics from our dataset, with a combination of UMAP [34] for dimensionality reduction, HDBSCAN [35] for clustering, and c-TF-IDF representation for topic extraction. This method provides interpretable topics while maintaining flexibility in cluster formation. Our implementation is structured around three key components:

1. Sentence Embedding Model: CrisisTransformers CT-mBERT-SE model [43] was used to create embeddings for its optimization for crisis related events.
2. Dimensionality Reduction: UMAP was applied in order to project the high dimensional data into a lower dimensional space, preserving key structures while reducing noise
3. Density-Based Clustering: HDBSCAN was used in order to create cluster by detecting automatically the optimal number of topic based on the density patterns.

UMAP

We made use of UMAP [34] in order to reduce the high dimensionality embedddings into a lower dimensional space. Our configuration is as follows:

- number of neighbor = 40: This parameter controls local vs. global structure preservation. We selected a high value due to the composition of our dataset in order to capture broader topic structure.
- number of components = 20: This is the target dimensionality reduction. By selecting a high value we retained more information, however we increase computational costs.
- minimum distance = 0.1: Controls how UMAP stack points. By selecting a lower value we aimed to maintain similar points closer.

In order to demonstrate the effectiveness of UMAP, we can compare PCA [51], which is a linear dimensionality reduction technique, with UMAP on the same embeddings extracted from a day-event dataset for event 009 and 017. As shown in (Tab: 4.3 and Tab: 4.4), PCA is unable to provide sufficient separation in order to obtain effective clustering due to the inability of capturing important details when dealing with a complex dataset. UMAP, on the other hand, preserves more local structure and captures deeper details while reducing the data dimensionality. Furthermore, PCA reduction produces similar distribution for different and uncorrelated events, making it less suitable for distinguishing clusters when dealing with a large corpus of data.

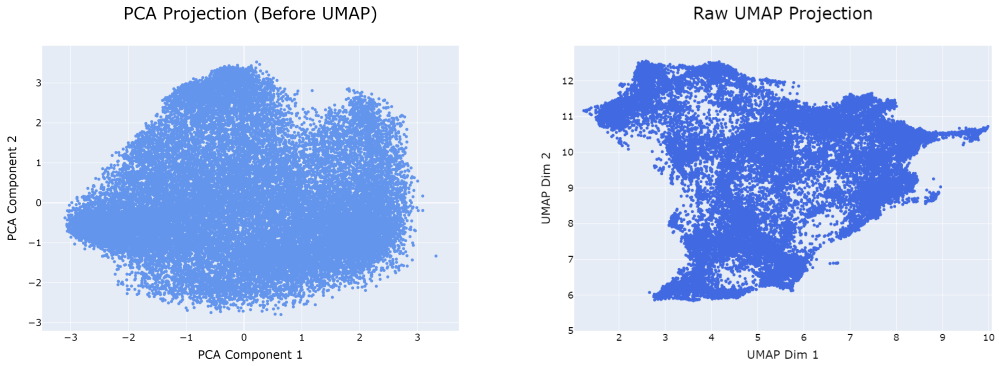

Table 4.3: PCA vs UMAP event 009 - Beirut Explosion

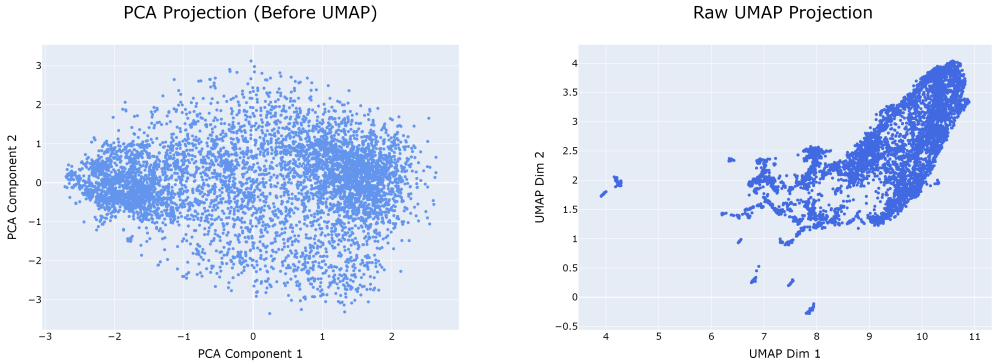

Table 4.4: PCA vs UMAP event 017 - Tornado Outbreak

HDBSCAN

We made use of HDBSCAN [35] in order to create clustering from our corpus of data, so that we could select only the inherent topic to the crisis event. Our configuration is as follows:

- minimum cluster size = 40: This parameter indicated the minimum points to select in order to form a cluster. By selecting a high value we aim to prevent over fragmentation.
- minimum samples = 20: This values controls the sensitivity noise. Due to the composition of our dataset we selected this value in order to ensure that outliers are not classified as topics.
- allow single cluster = False: This parameters ensures the formation of multiple clusters.

With our UMAP and HDBSCAN configuration we were able to separate data into clusters of more manageable size (Fig: 4.2) allowing us to apply the retrieval step. Moreover, this structured clustering improves the alignment between queries and relevant document groups, reducing noise and enhancing retrieval performance. By filtering out non relevant topics, we ensured that the retrieval in focused on well-defined topics, minimizing irrelevant results.

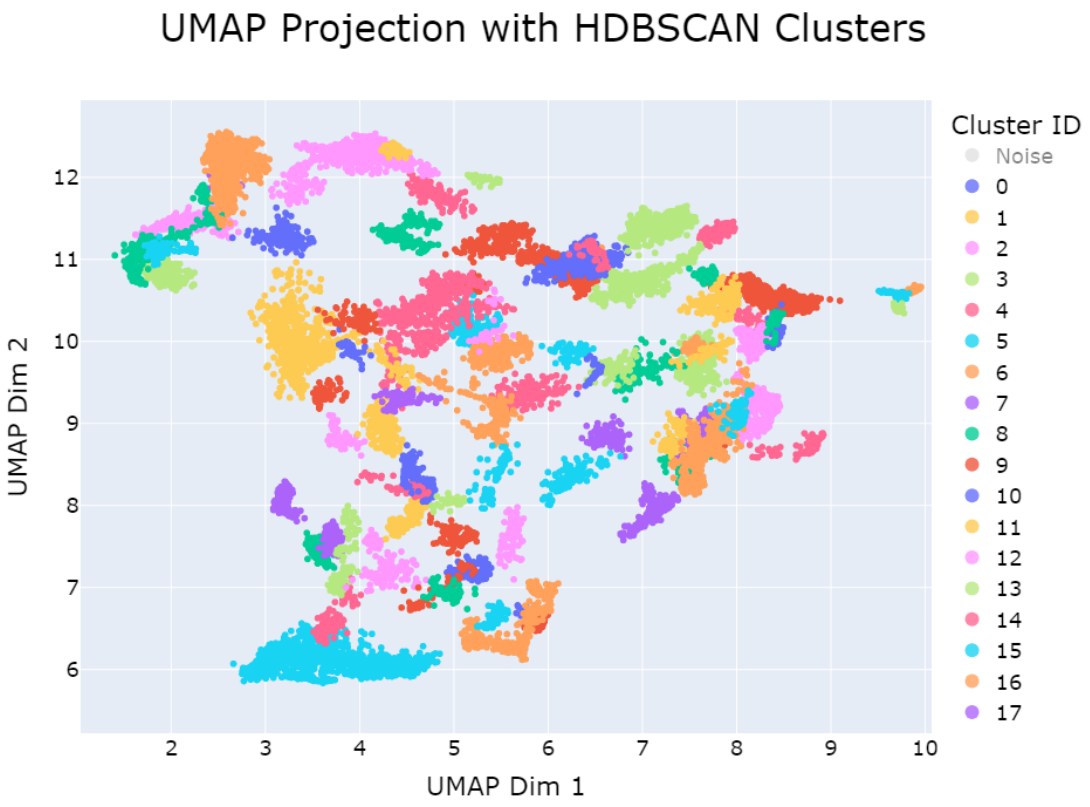

Figure 4.2: HDBSCAN clusters event 009 day 2020-08-24 without noise

4.3.1 Cluster Selection

In this section we will illustrate how cluster was selected in order to perform the retrieval step. When performing topic modeling with BERTopic, documents that can not be fitted in one clustered are labeled as outliers and are grouped in a single clustered. However, since we had to create one topic model for all the events in analysis, and due to the different nature of information arriving from multiple sources, we could not setup the model in order to create cluster of only relevant

data, and useful information was cast into the outliers cluster. For this reason while selecting the clusters we included also the outliers.

As previously stated in the methodologies chapter (3) for each topic, BERTopic produces a series of keywords to describe each cluster, from there we employed the event description, present in the dataset, to compute the semantic similarity between the topic (Fig: 4.3) and the event description selecting only the topics with similarity higher than a defined threshold (th > 0.5) and higher than the median value based on all the scores computed for the event day dataset in analysis.

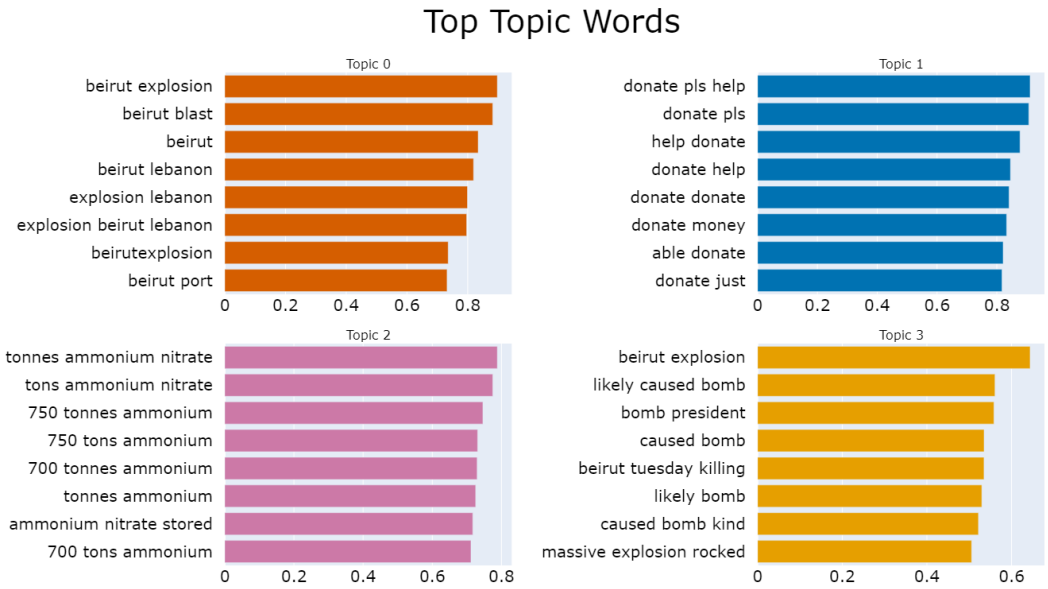

Figure 4.3: Barchart representing the top words for the top 6 topics for event 009 day 2020-08-04

We will now illustrate how clusters are selected for the event 009 on the day 2020-08-04. The first step consists in retrieving keywords from the event description which is provided with the dataset.

For the event 009 the description of the event is: *On 4 August 2020, a large amount of ammonium nitrate stored at the port of the city of Beirut, the capital of Lebanon, accidentally exploded, causing at least 180 deaths, 6,000 injuries, US$10–15 billion in property damage, and leaving an estimated 300,000 people homeless*

From which the following topic words are extracted using KeyBert [31]: `lebanon accidentally exploded, capital lebanon accidentally, lebanon accidentally,|`

```
accidentally exploded causing, beirut capital lebanon, city beirut,
beirut capital, city beirut capital, beirut, accidentally exploded,
exploded causing, port city beirut, capital lebanon, exploded,
2020 large ammonium, large ammonium nitrate, ammonium nitrate stored,
ammonium nitrate, lebanon, exploded causing 180
```

Afterwards, the semantic similarity between the topic words and the extracted key terms is computed using cosine similarity. Based on a predefined threshold, topics are either retained or discarded. As shown in the following table (Tab: 4.5), our model effectively identifies topics relevant to the events by leveraging the BERTopic generated words for each cluster. This process ensures that only meaningful and inherent documents are preserved, enhancing the overall quality of topic selection. Another thing we can notice from the before mentioned table, the outlier topic, contains a lot of inherent words to the event in analysis, so discarding it would mean losing a lot of potential useful information, for this reason, if eligible the outliers topic is maintained in the selection process.

Topic	Score	N Doc	Selected	Top Topic Words
0	0.7500	13621	YES	beirut explosion, beirut blast, beirut, beirut lebanon
1	0.4398	425	NO	donate pls help, donate pls, help donate, donate help
2	0.6734	386	YES	tonnes ammonium nitrate, tons ammonium nitrate
3	0.6742	275	YES	beirut explosion, likely caused bomb, bomb president
4	0.4272	213	NO	news updates, page latest updates, latest updates
5	0.4371	206	NO	hope stay safe, hope okay safe, hope safe okay
6	0.6243	202	YES	cause explosion unknown, cause explosion unclear
7	0.5209	202	YES	explore lebanon cgmethod, explorepage explore lebanon
8	0.6633	176	YES	hezbollah denied, israel denies, explosion caused israeli
9	0.4464	173	NO	really scary, damn scary, terrifying scary
-1	0.7934	13839	YES	beirut explosion, beirut, explosion lebanon

Table 4.5: Topic Distribution with Scores and Number of Documents for the first 10 topics for event 009 day 2020-08-24

To evaluate the effectiveness of the topic modeling selection, we computed the ROUGE-2 score, for the entire dataset and compare it with all the data selected by the topic modeling, in order to verify the quantity of lost information. Table (Tab: 4.6) presents the ROUGE-2 scores for the original dataset and the topic model passage in the pipeline. While the recall values remains high (from 0.9321 to 0.9012), indicating that most important information is preserved, both f1 and precision improve. This suggest that this process is able to remove non inherent to the event data, further reducing the dataset, while maintaining key information.

Metric	Original Dataset	After Topic
F1-score	0.0132	0.0265
Precision	0.0067	0.0136
Recall	0.9321	0.9012

Table 4.6: Mean ROUGE-2 scores across all events before and after topic modeling

4.4 Double Step Retrieve and Rerank Evaluation

As stated in the methodologies chapter (3), the next step of our pipeline that we employed to enhance retrieval effectiveness was a double step retrieval and reranking strategy. For each step a dual procedure is used, the first consist in selecting the top k documents for each query, using either BM25 [6] or dense retrieval, followed by a re-ranker which works with the query-document pair directly in order to better assign score to the documents based on the query. By jointly combining the query and the text into a single sequence, the reranker is able to enable context aware matching, which allows the model to better detect the relation between the two parts, and assign a more precise score. Re-ranker [52] do not produce embeddings, and needs a full forward passage in order to output a score, making them computational expensive, for this reason an encoder passage is needed in order to reduce the number of candidate passed to the re-ranker.

Given a query q and a document d, the re-ranker computes a relevance score $s(q, d)$ as follows:

$$s(q, d) = f_\theta([q; d]) \tag{4.1}$$

- f_θ is the pre-trained transformers model used with learned parameters (θ), in our case we employed mxbai-rerank-xsmall-v1 [47].

- $[q; d]$ is the concatenation of the query and the document.

Initially, BM25 is used due to its efficiency in handling large documents collection, because even after the topic modeling we still had a lot of documents in our collection. However BM25 only performs lexical search, which does not consider semantic similarity, and high score may be assigned to non related to the query document (Tab: 4.7). Therefore, the second step of retrieval in our pipeline consisted in using a dense retriever in order to refine the ranking by considering the semantic meaning of documents (Tab: 4.8).

Field	Value
Query	"How long did the tornado stay on the ground during 2020 Tornado Outbreak of April?"
Text	"Another strong and deadly long tracked #tornado in southern #Mississippi last night."
BM25 Score	12.4348
Cross Score	0.1467

Table 4.7: Sample of retrieved data of a non inherent fact for the query on the first day of event 017

Field	Value
Query	"Where have tornadoes touched down during Multiple tornadoes, storm warnings, emergency response, injuries, evacuations?"
Text	"An EF-0 tornado with estimated peak winds of 80 mph touched down in Homosassa (Citrus County, FL) Monday morning."
BM25 Score	9.7658
Dense Score	0.5961
Cross Score	1.0

Table 4.8: Sample of retrieved data from day one of event 017

Another reason for us to employed the double step retriever was to further reduce the number of documents, for this reason BM25 the top-k was set to 200 and for the dense encoder was set to the minimum number between 200 and the 10% of the data present passed to the retriever. Another parameter used to reduce the amount of selected data was the dynamic score, a parameter set based on the top percentile score assigned by the cross encoder, documents with a score under that value were not event considered because to irrelevant to the query.

Type	Top K	Dynamic Score
BM25	200	70
Dense	min(200, len(data)*0.1)	90

Table 4.9: Chosen parameter for the BM25 and the Dense retrieval

To evaluate the effectiveness of our retrieval step, we analyze how the number of documents is reduced per event while maintaining a high recall. The graph (Fig: 4.4) illustrate the document distribution before and after retrieval. The results indicate that our pipeline effectively reduces the number of documents per event, making the process more efficient (i.e. in event 001, the initial 51k documents were reduced to just 5k, representing a tenfold reduction in document size). This reduction is essential in order to minimize the noise in the dataset

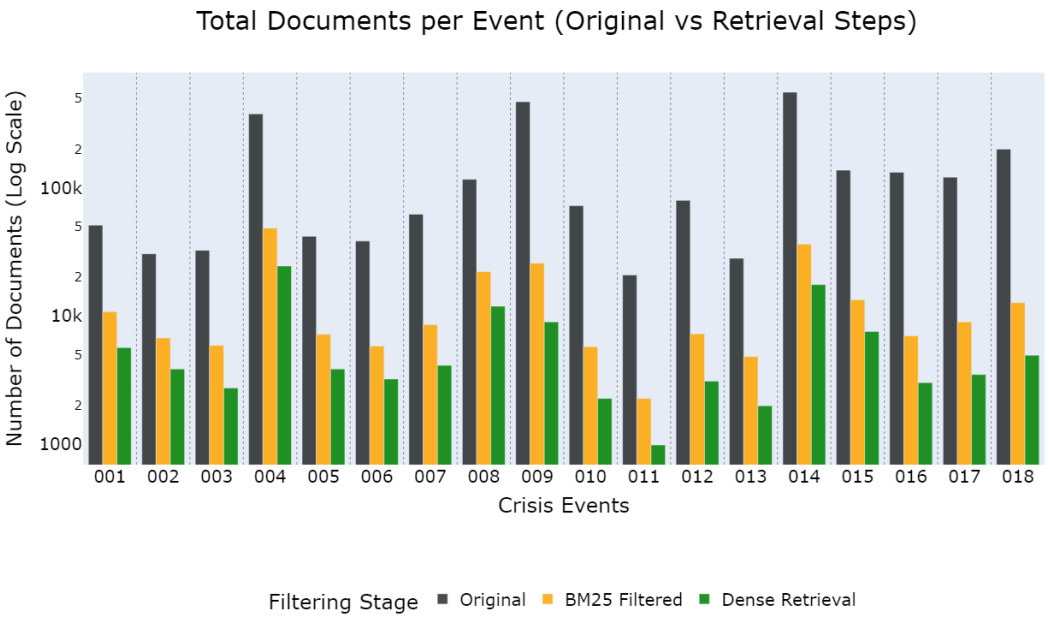

Figure 4.4: Data distribution before and after retrieval

while preserving relevant information for the last step of our pipeline, where the summary is composed.

The results (Tab: 4.10) confirms that BM25 retrieval significantly reduces the document set size (e.g. for event 001 the set is reduced from 51k to 10k documents) while maintaining a high recall, ensuring that relevant information is preserved. However the precision while improved, due to the reduction still remains to a low value, indicating that the dataset needs to be further reduced and that using only BM25 may still retrieve irrelevant and redundant information.

Metric	Original Dataset	After BM25 retrieval
F1-score	0.0132	0.0695
Precision	0.0067	0.0366
Recall	0.9321	0.8348

Table 4.10: Mean ROUGE-2 scores across all events before and after BM25 retrieval

By incorporating dense retrieval (Tab: 4.11), we further reduce the dataset size, leading to an improvement in the precision (from 0.0067 to 0.0718) and f1 score (from 0.0132 to 0.1280). However this selection, which highly reduces the documents set size comes at a slight recall trade-off which is decrease from 0.9321

to 0.7536. While this recall score does not determines the semantic correctness of the retrieved data, since only the bi-grams are evaluated, it suggest that the majority of useful information is retained by the retrieval pipeline.

Metric	Original Dataset	After Dense retrieval
F1-score	0.0132	0.1280
Precision	0.0067	0.0718
Recall	0.9321	0.7536

Table 4.11: Mean ROUGE-2 scores across all events before and after dense retrieval

The combination of BM25 and dense retrieval provides a balance between efficiency and effectiveness. BM25 acts as a first-stage retrieving, ensuring broad coverage, while the dense retriever refines relevance, enhancing precision and f1 score. This double retriever and rerank step significantly reduces the number of documents maintaining a high ROUGE2 recall, and giving us a solid base for the last part of the pipeline where the top documents are selected and the summary for each event is composed.

4.5 Summary Composure and Evaluation

After the retrieval system selected the most promising facts to include in the submission file, the next part of our pipeline consisted in aggregating the selected facts in order to produce a temporal summary that a crisis event manager could use in order to asses damage during a crisis emergencies. As stated in the methodologies chapter (3) each event submission should have a defined format (Tab: 3.2), for every fact included the information from which it was composed should be stated. Since the amount of data was still high we firstly reduce further the dataset by selecting the top 50% of data for each event, afterwards we applied a filter based on semantic similarity in order to reduce redundancy. This filter uses bge-m3 [45] model in order to compute the embeddings, afterwards a similarity matrix between all the candidates text is computed using cosine similarity. A facts is marked to be dropped if their score is higher than 0.95, this helped us reduce the redundancy of facts in the final selected dataset (Algorithm: 4).

In order to compose the submission file for each event the remaining data was grouped by `request_id`, which indicates the of the event in which the information was produced, and by `query_id`, which indicates the query for which the data was selected by the retrieval system. Afterwards we produced a collection for each event-day-query from which we selected the top information, based on the score assigned by the system, in order to not surpass the 500 character, and with

Algorithm 4 Filtering Redundant Texts Using Semantic Similarity

Require: Document set D, threshold τ, embedding model `bge-m3`
Ensure: Filtered document set D'
Compute embeddings E for all texts in D using `bge-m3`
Normalize and compute cosine similarity matrix S
Initialize mask $M \leftarrow$ `True` for all texts
for $i = 1$ to $|D| - 1$ **do**
    if $M[i]$ is `True` **then**
        Find indices J where $S[i, j] \geq \tau, \forall j > i$
        Set $M[J] \leftarrow$ `False`
    end if
end for
Return $D' \leftarrow D[M]$

a maximum of 7 entry for each included fact (Tab: 4.12), the importance score assigned was based on the mean score of the selected data.

Top percentile	Characters limit	Top entry limit
50%	500	7

Table 4.12: Chosen parameters for creating submission file

4.5.1 Evaluation

To evaluate our submission, we utilized the algorithm developed by the TREC [53], where the submitted top 32 facts, based on the importance score, from the submission were aggregated in order to compose the summary for each event. Afterwards the ROUGE2 and BERTScore algorithm were applied in order to evaluate the quality of the submitted summary that was produce by our system. The target summaries were composed by NIST assessors which evaluated the documents present in each day-event dataset, by labeling then as USEFUL, POOR, REDUNDANT and LAGGED. The target summaries were composed of only USEFUL data, which resemble the useful information that NIST assessor determinate to be effective in assessing the status of the crisis event in analysis.

In the following table (Tab: 4.15 and Tab: 4.14) we can see the obtained ROUGE-2 and BERTScore evaluation score for both the NIST summary, and for the Wikipedia summary, that was obtained from the lead part of the page for the event, which contains a quick summary of the most important part stated in the article. We also included the lenght of the summary in order to better visualize

the magnitude order of the submitted summaries.

Event	length	nist.f1	nist.precision	nist.recall	wiki.f1	wiki.precision	wiki.recall
001	83129	0.2565	0.1821	0.4339	0.0097	0.0049	0.4514
002	51633	0.3333	0.2706	0.4336	0.0136	0.0069	0.5327
003	53415	0.4061	0.3954	0.4174	0.0103	0.0052	0.4245
004	144427	0.3526	0.3258	0.3843	0.0243	0.0125	0.4719
005	45438	0.2850	0.2169	0.4154	0.0144	0.0073	0.3926
006	34894	0.3215	0.2924	0.3570	0.0138	0.0071	0.2766
007	26602	0.2548	0.2796	0.2340	0.0521	0.0290	0.2552
008	77658	0.3601	0.4485	0.3008	0.0242	0.0125	0.3547
009	72980	0.3353	0.2918	0.3941	0.0205	0.0105	0.3868
010	38310	0.3010	0.2283	0.4419	0.0247	0.0128	0.3738
011	26752	0.2549	0.1854	0.4079	–	–	–
012	38297	0.3523	0.3810	0.3277	–	–	–
013	40408	0.3824	0.4593	0.3275	0.0144	0.0073	0.3406
014	88700	0.3690	0.4269	0.3250	0.0365	0.0191	0.4274
015	80034	0.3651	0.3072	0.4498	0.0105	0.0053	0.4857
016	31883	0.2948	0.3150	0.2771	0.0269	0.0141	0.2829
017	50915	0.4037	0.4136	0.3943	0.0132	0.0067	0.4825
018	41956	0.3323	0.4681	0.2575	0.0161	0.0082	0.4275

Table 4.13: Comparison of NIST and Wiki ROUGE-2 scores for different events

Event	length	nist.f1	nist.precision	nist.recall	wiki.f1	wiki.precision	wiki.recall
001	83129	0.7575	0.7481	0.7671	0.7140	0.6695	0.7649
002	51633	0.7776	0.7499	0.8074	0.7092	0.6638	0.7612
003	53415	0.8280	0.8229	0.8331	0.7585	0.7153	0.8073
004	144427	0.7368	0.7305	0.7433	0.7369	0.7338	0.7399
005	45438	0.7922	0.7736	0.8118	0.7233	0.6911	0.7586
006	34894	0.7350	0.7029	0.7701	0.6725	0.6302	0.7209
007	26602	0.7293	0.7081	0.7518	0.7249	0.6923	0.7607
008	77658	0.7899	0.7568	0.8259	0.7318	0.7069	0.7585
009	72980	0.7680	0.7333	0.8061	0.7201	0.6973	0.7443
010	38310	0.7732	0.7612	0.7855	0.7348	0.7202	0.7499
011	26752	0.7905	0.7989	0.7822	–	–	–
012	38297	0.7934	0.7865	0.8003	–	–	–
013	40408	0.8315	0.8173	0.8462	0.7201	0.6995	0.7420
014	88700	0.7621	0.7862	0.7394	0.7558	0.7544	0.7572
015	80034	0.7843	0.7768	0.7919	0.7382	0.6997	0.7810
016	31883	0.7666	0.7449	0.7896	0.7313	0.7147	0.7486
017	50915	0.8062	0.7913	0.8216	0.6953	0.6425	0.7576
018	41956	0.8135	0.8048	0.8224	0.7463	0.7123	0.7837

Table 4.14: Comparison of NIST and Wiki BERTScore scores for different events.

Due to the length of the submitted summaries, BERTScore [16] could not be computed using the more accurate deberta-xlarge-mnli model [49], because it exceeded our available memory constraint. This model would have provide a more

reliable and comparable to those computed by CrisisFACTS participants results. To address this limitation, we employed distilbert-base-uncased [50], a lighter model that still yielded results. However, to validate the accuracy of our approach, we re-ran the BERTScore using deberta-xlarge-mnli on a more powerful GPU (A100 with 40 GB) for two events:

- Event 007, achieving an f1 score of 0.8818
- Event 011, achieving an f1 score of 0.8212

These scores were consistently higher than those obtained with the lighter model, confirming that the score achieved with distilbert-base-uncased provides a reliable approximation while allowing us to compute the scores across all events.

Type	**f1 score**	**precision**	**recall**
NIST	0.3311	0.3271	0.3655
Wiki	0.0180	0.0094	0.3537

Table 4.15: Mean ROUGE-2 Score

Type	**f1 score**	**precision**	**recall**
NIST	0.7797	0.7663	0.7942
Wiki	0.6451	0.6191	0.6742

Table 4.16: Mean BERTScore

The results in Tables (Tab: 4.15) and (Tab: 4.16) shows the effectiveness of our retrieval pipeline, which combines a topic modeling and a double retrieve step with a re-ranking approach. Our system was evaluated across all eighteen crisis events contained in the TREC CrisisFACTS 2023 challenge, and we achieved strong performance across multiple metrics. Even though the ROUGE-2 scores could be improved, in particular by achieving higher precision while maintaining a the recall value high throughout a better final fact selection with the use of LLM models. The BERTScore results shows a strong semantic connection between the target and reference summary (F1=0.7797) demonstrating the effectiveness of our pipeline in capturing meaningful contextual relevance information. Although a complete comparison with the submitted results would have given a better representation of the potential of our pipeline, from the TREC proceedings for the challenge [54] we could compare only the events from 009 to 018. These events still form a substantial portion of the challenge dataset allowing for a meaningful assessments agains the mean value of all the participant submission for each event.

Event	nist.f1	nist.precision	nist.recall	wiki.f1	wiki.precision	wiki.recall
009	0.2133	0.2396	0.2458	0.0336	0.0196	0.2427
010	0.1491	0.1143	0.2860	0.0232	0.0128	0.1893
011	0.1452	0.1001	0.3590	–	–	–
012	0.1791	0.1797	0.2101	–	–	–
013	0.2297	0.2242	0.2861	0.0191	0.0102	0.2436
014	0.2343	0.2956	0.2208	0.0536	0.0322	0.2211
015	0.2337	0.2335	0.2872	0.0207	0.0180	0.3283
016	0.1821	0.1820	0.2081	0.0343	0.0196	0.1782
017	0.2242	0.2664	0.2198	0.0186	0.0099	0.2335
018	0.2281	0.2913	0.2116	0.0203	0.0108	0.2247

Table 4.17: Mean Baseline ROUGE-2 CrisiFACTS run

Event	nist.f1	nist.precision	nist.recall	wiki.f1	wiki.precision	wiki.recall
009	0.3353	0.2918	0.3941	0.0205	0.0105	0.3868
010	0.3010	0.2283	0.4419	0.0247	0.0128	0.3738
011	0.2549	0.1854	0.4079	–	–	–
012	0.3523	0.3810	0.3277	–	–	–
013	0.3824	0.4593	0.3275	0.0144	0.0073	0.3406
014	0.3690	0.4269	0.3250	0.0365	0.0191	0.4274
015	0.3651	0.3072	0.4498	0.0105	0.0053	0.4857
016	0.2948	0.3150	0.2771	0.0269	0.0141	0.2829
017	0.4037	0.4136	0.3943	0.0132	0.0067	0.4825
018	0.3323	0.4681	0.2575	0.0161	0.0082	0.4275

Table 4.18: Submitted ROUGE-2 Score by Event

Type	f1 score	precision	recall
Baseline	0.2011	0.2137	0.2445
Submitted	0.3390	0.3476	0.3602

Table 4.19: Mean ROUGE-2 Score Baseline vs Submitted comparison on NIST summary

During the tests conducted, we mainly relied on ROUGE-2 results in evaluating the effectiveness of our retrieval system, due to the fact that we focused on fact extraction and this algorithm measures the bi-gram overlap. Our pipeline demonstrated substantial improvements over the baseline NIST target summary, which was the main focus of this thesis. The f1 score increased by nearly 70%, from 0.2011 to 0.3390 (Tab: 4.18), showing that our approach achieves much stronger alignment with the target summaries. If we considering individual events, our system outperformed the baseline in across all the events, with the event 017

achieving the highest improving, from 0.2242 to 0.4037 (Tab: 4.17 and Tab: 4.18), demonstrating the effectiveness of our system in handling different crisis events. As for the wiki summaries since they were retrieved from wikipedia, and not compose from the dataset, the ROUGE-2 score is not indicative of the performance of the system.

Event	nist.f1	nist.precision	nist.recall	wiki.f1	wiki.precision	wiki.recall
009	0.5898	0.5795	0.6011	0.5213	0.4971	0.5490
010	0.5397	0.5372	0.5428	0.5075	0.4801	0.5384
011	0.5849	0.5842	0.5856	–	–	–
012	0.5417	0.5370	0.5467	–	–	–
013	0.6086	0.6094	0.6080	0.5076	0.4749	0.5454
014	0.6521	0.6584	0.6463	0.4970	0.4953	0.4988
015	0.5915	0.5902	0.5933	0.5357	0.4977	0.5804
016	0.5889	0.5870	0.5914	0.4982	0.4844	0.5131
017	0.5767	0.5717	0.5819	0.5001	0.4521	0.5597
018	0.6161	0.6190	0.6136	0.5189	0.4697	0.5800

Table 4.20: Mean Baseline BERTScore CrisiFACTS run

Event	nist.f1	nist.precision	nist.recall	wiki.f1	wiki.precision	wiki.recall
009	0.7680	0.7333	0.8061	0.7201	0.6973	0.7443
010	0.7732	0.7612	0.7855	0.7348	0.7202	0.7499
011	0.7905	0.7989	0.7822	–	–	–
012	0.7934	0.7865	0.8003	–	–	–
013	0.8315	0.8173	0.8462	0.7201	0.6995	0.7420
014	0.7621	0.7862	0.7394	0.7558	0.7544	0.7572
015	0.7843	0.7768	0.7919	0.7382	0.6997	0.7810
016	0.7666	0.7449	0.7896	0.7313	0.7147	0.7486
017	0.8062	0.7913	0.8216	0.6953	0.6425	0.7576
018	0.8135	0.8048	0.8224	0.7463	0.7123	0.7837

Table 4.21: Submitted BERTScore by Event

Type	f1 score	precision	recall
Baseline	0.5880	0.5875	0.5148
Submitted	0.7889	0.7801	0.7985

Table 4.22: Mean BERTScore Baseline vs Submitted comparison on NIST summary

For BERTScore our pipeline was able to achieve a significantly increment in the overall results, obtaining a nearly 40% boost in the f1 score, from an f1 score of

0.5880 to 0.7889 (Tab: 4.22), demonstrating the capacity of our system in retrieving semantically related documents to the crisis event in analysis. As for the each individuals events, our system outperformed the baseline across all of them, with event 012 achieving the highest improving, from 0.5417 to 0.7934 (Tab: 4.20 and Tab: 4.21), demonstrating the effectiveness of our models in different scenarios. Due to the nature of BERTScore, which relies on embedding to confront text, we can also take into the account the results achieved by our system against the wiki summary, where event in this scenarios we outperform the baseline results.

In synthesis the results obtained from ROUGE-2 and BERTScore demonstrated that our systems consistently outperformed the mean baseline results from Crisis-FACTS 2023 submitted solutions. The performance gains can be attributed to our topic modeling and double step retrieval and re-ranking strategy, which enhances semantic relevant reducing irrelevant documents without affecting the scalability.

Chapter 5

Conclusion

In this thesis, we developed a multiple step retrieval pipeline in order to enhance information extraction during crisis events following the TREC CrisisFACTS 2023 challenge requirements. Our approach integrates topic modeling technique to refine the dataset collection, ensuring that only semantically related documents are considered during the selection phase. To enhance retrieval effectiveness, we employed a dual retrieve and re-ranking strategy. By combining BM25, dense retrieval and cross-encoder re-ranking, we significantly improved the relevance and accuracy of the retrieved documents while maintaining computational efficiency.

The experimental results shows the effectiveness of our system in retrieving important facts in order to produce accurate and non redundant summary for crisis event managers. We achieve higer ROUGE-2 and BERTScore metrics compared to the mean results of CrisisFACTS participants across evaluated events, without compromising the scalability of the system.

Our findings highlight the effectiveness of our retrieval strategy in extracting useful information during crisis events. Future work could focus on further optimization, such as improving query selection in order to ensure comprehensive coverage of all aspects for different crisis scenarios, refining topic modeling to reduce noise during cluster selection, and fine-tuning re-ranking models to enhance recall while minimizing redundancy in retrieved content. Additionally, integrating Large Language Models (LLMs) could further improve the system by enhancing the final document selection, which is the weakest part of our pipeline and generate more accurate, and human-like summaries.

Bibliography

[1] Carlos Castillo. *Big Crisis Data: Social Media in Disasters and Time-Critical Situations*. July 2016. ISBN: 9781107135765. DOI: 10.1017/9781316476840 (cit. on p. 1).

[2] Leysia Palen and Sophia Liu. «Citizen Communications in Crisis: Anticipating a Future of ICT-Supported Public Participation». In: Apr. 2007, pp. 727–736. DOI: 10.1145/1240624.1240736 (cit. on p. 1).

[3] Muhammad Imran, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. *Processing Social Media Messages in Mass Emergency: A Survey*. 2015. arXiv: 1407.7071 [cs.SI]. URL: https://arxiv.org/abs/1407.7071 (cit. on p. 1).

[4] Sarah Vieweg, Amanda Hughes, Kate Starbird, and Leysia Palen. «Microblogging during two natural hazards events: What Twitter may contribute to situational awareness». In: vol. 2. Apr. 2010, pp. 1079–1088. DOI: 10.1145/1753326.1753486 (cit. on p. 1).

[5] Amanda Hughes, Lise St Denis, Leysia Palen, and Kenneth Anderson. «Online public communications by police & fire services during the 2012 Hurricane Sandy». In: *Conference on Human Factors in Computing Systems - Proceedings* (Apr. 2014). DOI: 10.1145/2556288.2557227 (cit. on p. 2).

[6] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. «Okapi at TREC-3». In: *Proceedings of the Third Text REtrieval Conference (TREC-3)*. 1994, pp. 109–126 (cit. on pp. 2, 10, 38).

[7] Stephen Robertson and Hugo Zaragoza. «The Probabilistic Relevance Framework: BM25 and Beyond». In: *Foundations and Trends in Information Retrieval* 3.4 (2009), pp. 333–389. DOI: 10.1561/1500000019 (cit. on pp. 2, 10).

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». In: *arXiv* abs/1810.04805 (2018). arXiv: 1810.04805 [cs.CL] (cit. on pp. 2, 7, 19, 30).

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. «Attention is All You Need». In: *arXiv* abs/1706.03762 (2017). arXiv: 1706.03762 [cs.CL] (cit. on pp. 2, 6, 8).

[10] Alan M Turing. *Computing machinery and intelligence*. Springer, 2009 (cit. on p. 5).

[11] Frank Rosenblatt. «The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain». In: *Psychological Review* 65.6 (1958), pp. 386–408. DOI: 10.1037/h0042519 (cit. on p. 6).

[12] Sepp Hochreiter and Jürgen Schmidhuber. «Long short-term memory». In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cit. on p. 6).

[13] *BERT Contextual Word Embeddings*. URL: https://dvgodoy.github.io/dl-visuals/BERT/ (cit. on p. 7).

[14] OpenAI. *GPT-4o: OpenAI's Most Advanced Model*. Accessed: 2024-06-10. 2024. URL: https://openai.com/index/hello-gpt-4o/ (cit. on pp. 8, 22).

[15] Chin-Yew Lin. «ROUGE: A Package for Automatic Evaluation of Summaries». In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://aclanthology.org/W04-1013/ (cit. on pp. 8, 32).

[16] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. «BERTScore: Evaluating Text Generation with BERT». In: *arXiv* abs/1904.09675 (2019). arXiv: 1904.09675 [cs.CL] (cit. on pp. 8, 9, 31, 43).

[17] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. «Bleu: a Method for Automatic Evaluation of Machine Translation». In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Ed. by Pierre Isabelle, Eugene Charniak, and Dekang Lin. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://aclanthology.org/P02-1040/ (cit. on p. 8).

[18] *Archie: The first internet search engine*. URL: https://www.linkedin.com/pulse/archie-first-internet-search-engine-mail-com (cit. on p. 10).

[19] Wikipedia contributors. *Probabilistic relevance model — Wikipedia, The Free Encyclopedia*. [Online; accessed 10-February-2025]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Probabilistic_relevance_model&oldid=1250196097 (cit. on p. 10).

[20] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. «Dense Passage Retrieval for Open-Domain Question Answering». In: *arXiv* abs/2004.04906 (2020). arXiv: 2004.04906 [cs.CL] (cit. on p. 13).

[21] National Institute of Standards and Technology (NIST). *TREC22 Temporal Summarization Track Overview*. 2013. URL: https://pages.nist.gov/trec-browser/trec22/tempsumm/overview/ (cit. on p. 16).

[22] National Institute of Standards and Technology (NIST). *TREC26 Temporal Summarization Track Overview*. 2017. URL: https://pages.nist.gov/trec-browser/trec26/rts/overview/ (cit. on p. 16).

[23] National Institute of Standards and Technology (NIST). *TREC29 Temporal Summarization Track Overview*. 2020. URL: https://pages.nist.gov/trec-browser/trec29/incident/overview/ (cit. on p. 16).

[24] National Institute of Standards and Technology (NIST). *TREC32 Temporal Summarization Track Overview*. 2023. URL: https://pages.nist.gov/trec-browser/trec32/crisis/overview/ (cit. on p. 16).

[25] Maarten Grootendorst. «BERTopic: Neural topic modeling with a class-based TF-IDF procedure». In: *arXiv* abs/2203.05794 (2022). arXiv: 2203.05794 [cs.CL] (cit. on pp. 18, 23, 30, 33).

[26] Dimo Angelov. «Top2Vec: Distributed Representations of Topics». In: *arXiv* abs/2008.09470 (2020). arXiv: 2008.09470 [cs.CL] (cit. on p. 18).

[27] *2023 TREC CrisisFACTS Track*. 2023. URL: https://crisisfacts.github.io/#tasks (cit. on p. 19).

[28] *INCIDENT STATUS SUMMARY (ICS 209)*. URL: https://training.fema.gov/emiweb/is/icsresource/assets/ics%20forms/ics%20form%20209,%20incident%20status%20summary%20(v3).pdf (cit. on pp. 19, 22).

[29] Karen Spärck Jones. «A Statistical Interpretation of Term Specificity and Its Application in Retrieval». In: *Journal of Documentation* 28.1 (1972), pp. 11–21. DOI: 10.1108/eb026526 (cit. on p. 23).

[30] *KeyBertInspired*. URL: https://maartengr.github.io/BERTopic/api/representation/keybert.html (cit. on p. 24).

[31] Maarten Grootendorst. *KeyBERT: Minimal keyword extraction with BERT*. Accessed: 2025-01-31. 2020. URL: https://github.com/MaartenGr/KeyBERT (cit. on pp. 24, 36).

[32] Eric Brill. «A simple rule-based part of speech tagger». In: *Proceedings of the Third Conference on Applied Natural Language Processing*. 1992, pp. 152–155. DOI: 10.3115/974499.974526 (cit. on p. 24).

[33] Jaime Carbonell and Jade Goldstein. «The use of MMR, diversity-based reranking for reordering documents and producing summaries». In: *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1998, pp. 335–336. DOI: 10.1145/290941.291025 (cit. on p. 24).

[34] Leland McInnes, John Healy, and James Melville. «UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction». In: *arXiv preprint* (2018). arXiv: 1802.03426 [stat.ML]. URL: https://arxiv.org/abs/1802.03426 (cit. on pp. 24, 33).

[35] Ricardo J. G. B. Campello, Davoud Moulavi, and Jörg Sander. «Density-Based Clustering Based on Hierarchical Density Estimates». In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. Springer, 2013, pp. 160–172. DOI: 10.1007/978-3-642-37456-2_14 (cit. on pp. 24, 33, 34).

[36] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. «A density-based algorithm for discovering clusters in large spatial databases with noise». In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. 1996, pp. 226–231 (cit. on p. 24).

[37] *CountVectorizer*. URL: https://maartengr.github.io/BERTopic/getting_started/vectorizers/vectorizers.html#countvectorizer (cit. on p. 24).

[38] *SentenceTransformers*. URL: https://sbert.net/ (cit. on p. 25).

[39] *Retrieve and Re-Rank*. URL: https://sbert.net/examples/applications/retrieve_rerank/README.html (cit. on p. 25).

[40] Jason Ansel, Edward Yang, Horace He, N. Gimelshein, Animesh Jain, et al. «PyTorch2: Faster Machine Learning Through Dynamic Python Byte code Transformation and Graph Compilation». In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. ASPLOS '24. Association for Computing Machinery, 2024, pp. 929–947. ISBN: 9798400703850. DOI: 10.1145/3620665.3640366. URL: https://pytorch.org/assets/pytorch2-2.pdf (cit. on p. 29).

[41] Nils Reimers and Iryna Gurevych. «Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks». In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2019. URL: http://arxiv.org/abs/1908.10084 (cit. on pp. 29, 30).

[42] Thomas Wolf, Lysandre Debut, Victor Sanh, et al. «HuggingFace's Transformers: State-of-the-art Natural Language Processing». In: *arXiv* abs/1910.03771 (2019). arXiv: 1910.03771 [cs.CL] (cit. on p. 29).

[43] Rabindra Lamsal, Maria Rodriguez Read, and Shanika Karunasekera. «Semantically Enriched Cross-Lingual Sentence Embeddings for Crisis-related Social Media Texts». In: (2024). arXiv: 2403.16614 [cs.CL] (cit. on pp. 30, 31, 33).

[44] Dorian Brown. *Rank-BM25: A Collection of BM25 Algorithms in Python*. 2020. DOI: 10.5281/zenodo.4520057. URL: https://doi.org/10.5281/zenodo.4520057 (cit. on pp. 30, 31).

[45] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. *BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation*. 2024. arXiv: 2402.03216 [cs.CL] (cit. on pp. 30, 31, 41).

[46] *Beijing Academy of Artificial Intelligence*. URL: https://huggingface.co/BAAI (cit. on p. 30).

[47] *mxbai-rerank-xsmall-v1*. URL: https://www.mixedbread.ai/docs/reranking/mxbai-rerank-xsmall-v1 (cit. on pp. 30, 31, 38).

[48] *MixedbreadAI*. URL: https://www.mixedbread.ai/ (cit. on p. 30).

[49] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. «DEBERTA: DECODING-ENHANCED BERT WITH DISENTANGLED ATTENTION». In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=XPZIaotutsD (cit. on pp. 31, 43).

[50] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. «DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter». In: *ArXiv* abs/1910.01108 (2019) (cit. on pp. 31, 44).

[51] Ian T. Jolliffe. *Principal Component Analysis*. 2nd. Springer, 2002 (cit. on p. 33).

[52] *Cross-Encoder*. URL: https://www.sbert.net/examples/applications/cross-encoder/README.html (cit. on p. 38).

[53] *TREC Github - utilities*. URL: https://github.com/crisisfacts/utilities (cit. on p. 42).

[54] Cody Buntain, Amanda Lee Hughes, Richard McCreadie, Benjamin D. Horne, Muhammad Imran, and Hemant Purohit. «CrisisFACTS 2023 - Overview Paper». In: *The Thirty-Second Text REtrieval Conference Proceedings (TREC 2023), Gaithersburg, MD, USA, November 14-17, 2023*. Ed. by Ian Soboroff and Angela Ellis. Vol. 1328. NIST Special Publication. National Institute of Standards and Technology (NIST), 2023. URL: https://trec.nist.gov/pubs/trec32/papers/Overview%5C_crisis.pdf (cit. on p. 44).