Summarizing Social Media & News Streams for Crisis-related Events by Integrated Content-Graph Analysis: TREC-2023 CrisisFACTS Track

Hossein Salemi hsalemi@gmu.edu
Yasas Senarath ywijesu@gmu.edu
Tarin Sultana Sharika tsharika@gmu.edu
Anuridhi Gupta agupta29@gmu.edu
Hemant Purohit hpurohit@gmu.edu
George Mason University

Abstract

Extracting informative content from different sources of data like social media and news websites and summarizing it is critical for disaster response agencies during crises. This paper describes our proposed system to extract and rank facts from online data sources for summarizing crisis-related events in the TREC 2023 CrisisFACTS track. First, our system leverages established methods such as REBEL or ClausIE to extract relevant facts from the input data stream. Then, since the summary should reflect the information needed by the response agencies, our system filters the extracted facts using an extended set of indicative terms used by those agencies. We then employ an integrated content-graph analysis to capture the similarity of facts to each other, facts to queries, and facts to indicative terms to score the importance of extracted facts. We evaluate and compare the performance of our proposed system by utilizing two extractive methods to extract facts from the multi-stream data and score them for summarizing the crisis-related events.

Keywords— Fact Extraction, Stream Processing, Disaster Management, Disaster Response

1 Introduction

Finding the relevant information to support disaster response efforts is a crucial challenge during times of crises [12]. Conventional information retrieval methods can be employed but sifting through vast amounts of unstructured data from various sources, including social media streams, remains a significant challenge. Recent works on summarization can help analyze textual content from articles of a given domain such as news, but they do not work well with social media text, which often has noisy content [11]. Another major challenge in information retrieval is to identify the important facts from the given stream of social media data. Matching the posts related to a user query is important; however, the resulting facts/summaries may not be that important in the context of disaster management. It is meaningful to identify heuristics that make a given summary/fact statement important or not in order to cater to the needs of the disaster management experts.

This work focuses on the extraction and retrieval of facts that serve the information needs of disaster response agencies, by processing a stream of documents from multiple online sources (e.g., Twitter, Reddit, Facebook, and online news sources). Our proposed system based on an integrated content-graph analysis has the ability to extract, rank, and query critical facts from a stream of crisis-related data. To accomplish this, first, we employ two methods from recent literature [3, 4]: pretrained seq2seq networks and dependency parsing. These methods extract facts from the input stream of text, which are then filtered using specific keywords derived from example queries. Finally, we use a graph-based representation of the facts and predefined queries and indicative terms to measure centrality and determine the importance of each fact in the stream.

In this paper, we will describe the details of our proposed system, the method of integrated content-graph analysis, and the results obtained by those methods on the datasets provided by the CrisisFACTS challenge.

2 Challenge Description

The CrisisFACTS challenge aims to address the requirement, for real-time fact-finding and summarization during disaster response situations [1]. It is an effort with TREC aiming to advance research and enhance

disaster management capabilities by utilizing state-of-the-art machine learning and natural language processing techniques. In this challenge participants receive datasets from sources such as Twitter, Reddit, Facebook and online news platforms gathered through the NELA News Collection. These datasets are complemented with queries (user profiles) that outline the information needs of stakeholders involved in disaster response extracted from FEMA ICS 209 forms. Participants are assigned with the task of developing systems that can effectively integrate these data sources into fact lists. These facts can then be compiled into summaries. The objective of this challenge is to push the boundaries of real-time fact-finding and summarization. The evaluation of participant runs involves comparing the comprehensiveness and redundancy ratio of their submitted facts against gold-standard facts constructed by NIST (National Institute of Standards and Technology) assessors. The assessment of comprehensiveness and redundancy ratio will be made using a fact-matching function and evaluated through various metrics, such as ROUGE-x and BERTScore. The winning solution is selected based on its ability to accurately predict future crisis events, consistently generate accurate and relevant fact lists and summaries, and demonstrate its ability to adapt to new information sources and data distributions. Additionally, the solution needs to be scalable, secure, and user-friendly. In both approaches, participant systems' lists of facts are truncated to a private k value based on NIST assessors' results.

3 Proposed System using Integrated Content-Graph Analysis

In this section, we present the architecture of our proposed system for extracting and prioritizing facts from the stream of data. Figure 1 depicts the overall architecture of our system and its different modules. In this design, for each day of an event, the system receives the stream of data from different sources including, Twitter, Reddit, Facebook, and news, in addition to the list of queries, and generates a list of facts and their importance scores.

Our system extracts the facts through four modules: In the first step, the streams are analyzed by **Text to Fact** module and for each stream record a fact is extracted. At the same time, the **Query Extender** module enlarges the list of queries by extending the indicative terms for each query. This helps the system to filter the extracted facts based on a broader list of indicative terms that leads to finding more relevant facts. After that, the **Filtering** module analyzes the extracted facts and filters them by using the extended indicative terms. This module outputs a filtered list of facts and their relevancy score to each indicative term. Finally, these filtered facts are passed to the **Scoring** module to be ranked based on their importance for generating the summary of the event-day report. In this module, a graph containing the extracted facts, extended indicative terms, and queries are created and a graph analysis technique is used to calculate the importance of each fact based on its similarity to other facts and its relevancy to queries and indicative terms. The implementation details of the modules are presented in the following sections.

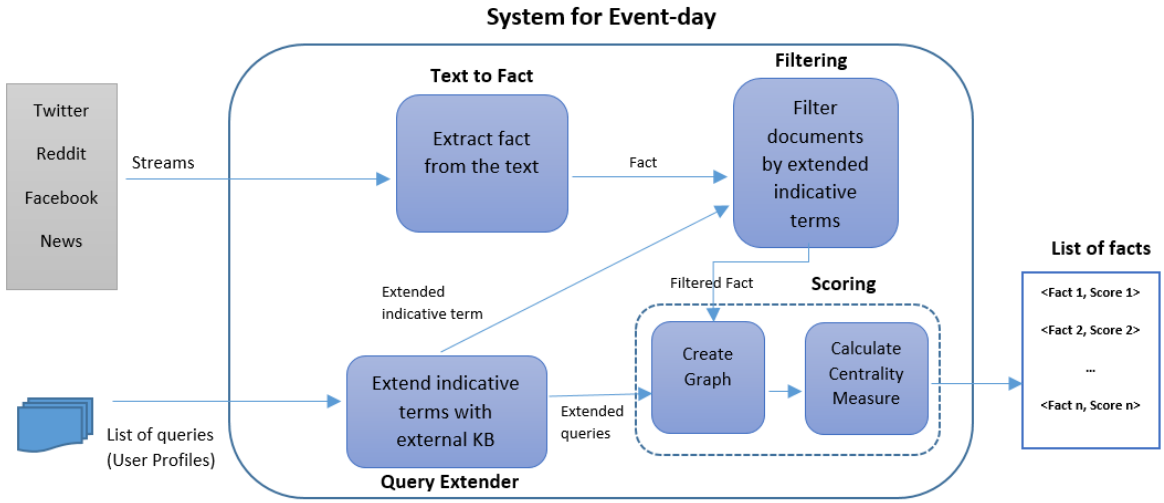

Figure 1: System Design

3.1 Text to Fact

In this section, we will discuss about the approach used to convert the input text from social media into facts. We used two methods to convert the input sentences to fact format. The input for this module will be a text

Query	Indicative Terms	Extended Indicative Terms
What regions have announced a state of emergency	state of emergency	announced state emergency
Where are emergency services deployed	service teams emergency services	emergency services deployed
What hazardous chemicals or materials are involved	fuel hazard waste infectious chemical	hazardous chemicals materials

Table 1: Query extender using KeyBERT.

document and the output will be a set of free-text facts.

Method A: The first method we utilized to convert the input text to facts is called REBEL [1] [3]. This method uses an autoregressive seq2seq model to generate triplets as a sequence of text. The REBEL method is based on Encoder-Decoder Transformer (BART) and performs end-to-end relation extraction for more than 200 different relation types.

Method B: In our second method we utilized an implementation [4] of ClausIE (Clause-Based Open Information Extraction) [5]. ClausIE extracts clauses from an input sentence by relying on dependency parsing and a small collection of domain-independent lexica.

3.2 Query Extender

The *Query Extender* module of the proposed system is based on keyword extraction using the KeyBERT method [8]. Keyword extraction is a crucial task in document analysis in terms of information retrieval. Indicative terms can be considered a set of keywords or phrases that provide valuable insight about the event. The extension of indicative terms provides a broader area with more relevant information. There are several methods to extract keywords or phrases from documents but KeyBERT shows better results compared to traditional methods [9]. Table 1 shows a few examples of queries, indicative terms, and extended indicative terms using keyBERT. These examples demonstrate how the query extender method targets more relevant facts based on the extension.

3.3 Filtering

The stream of data can be noisy, so all facts that are extracted by *Text to Fact* module would not be relevant to the event. Therefore, once the facts have been extracted, they are filtered by *Filtering* module to generate the list of facts which are informative for generating the summary of the event. Since the filtered facts should represent the information needs of disaster response agencies, we exploit the indicative terms from the user profiles list extended by *Query Extender* module.

To achieve this goal, in the first step, the *Filtering* module creates an indexer to index the streams of data for an event-day based on the extracted facts. Then, the module leverages the indexer to retrieve the facts which are relevant to the extended indicative terms. It should be considered that each fact can be relevant to more than one indicative term. Therefore, we query the indexer with every indicative term and the indexer returns a list of facts and their relevancy scores to that indicative term. For each indicative term, we select the top K more relevant facts and concatenate them to generate the list of filtered facts for the event-day.

3.4 Scoring

The list of facts filtered by *Filtering* module is relevant to the event, but we still need to rank them based on their importance for summarizing the event-day. Although the *Filtering* module provides the relevancy scores of each fact to the indicative terms, these scores are not the only indicators of the fact's importance. Thus, in our system, the *Scoring* module is responsible for calculating the importance of each fact based on a graph analysis method by taking diverse importance factors into account. This module calculates the importance score of the facts in two steps as described next: *Create Graph* and *Calculate Centrality Measure*.

3.4.1 Graph Construction

The first step in analyzing the importance of the facts is specifying the factors that can represent the importance of the fact. Here, we use the following three factors:

[1] https://huggingface.co/Babelscape/rebel-large

- **The similarity of facts to each other:** In our system, facts are extracted independently from documents from different sources of data such as news, Twitter, Facebook, or Reddit. During crises, when an important event happens, it is most likely to be discussed on different platforms. Additionally, it is probable that some topics such as requesting for help or reporting rescue efforts are expressed several times. Therefore, if a fact is more similar to other facts (by considering that the facts have been filtered to be relevant to the event), it can indicate that this fact is likely conveying an important fact that has been discussed several times.
- **Relevancy of facts to indicate terms:** As we discussed in the previous section, the *Filtering* module outputs a list of relevant facts as well as their relevancy scores to the indicative terms. Since the indicative terms are extracted and extended from the information needs of disaster response agencies, if a fact has a higher relevancy score to an indicative term, that fact is likely a more important fact to be reported. Additionally, if a fact is relevant to more than one indicative term, this can also reveal the importance of that fact. Therefore, the relevancy score reported by the *Filtering* module in addition to the number of times a fact retrieved by the *Filtering* module (due to being relevant to several indicative terms) can be the factors that imply the importance of the fact.
- **The similarity of facts to user profile queries:** Although indicative terms have been extracted from user profile queries, they may not convey the whole context of the query. So, to find out how much a fact is aligned with the queries, we calculate the similarity of the fact with the queries and take it into account in the graph analysis method. This means that if a fact is more similar to the queries, it is likely more informative.

Based on these three factors, the *Scoring* module creates a graph containing the facts, the indicative terms, and the queries. In this graph, we have three types of nodes: *fact node*, *term node*, and *query node*. Moreover, we have three types of edges that represent our mentioned importance factors: *fact-fact* edges which show the similarity of facts to each other, *fact-term* edges which indicate the relevancy of each fact to indicative terms, and *fact-query* edges which specify the similarity of facts to the queries. Figure 2 depicts the schematic of the graph used in our scoring method.

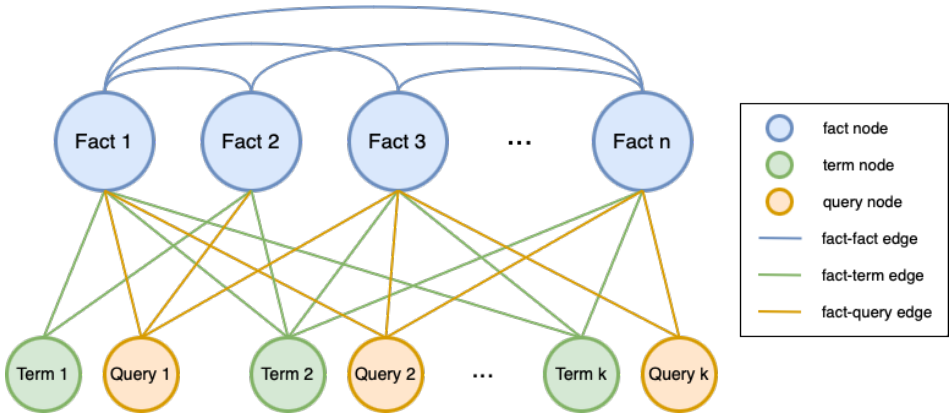

Figure 2: The graph for scoring the facts

To calculate the weights of *fact-fact* edges in the graph, we use a text similarity method by using *Sentence-BERT* model [13]. We calculate the embeddings of each pair of facts using *Sentence-BERT* model and then compute the cosine of their embeddings as their similarity measure. We use this similarity measure as the weight of the *fact-fact* edge between two *fact nodes*, so these weights are in the range of [0,1]. However, we just add the edges for *fact nodes* whose similarities are more than the threshold Sim_{min}. We use the same method for calculating the weights of *fact-query* edges by calculating the cosine value of the embeddings for each pair of (fact, query). In order to calculate the weight of *fact-term* edges, we use the relevancy score reported by *Filtering* module. However, since we need all weights of the graph to be normalized, we leverage the min-max normalization method to normalize the *fact-term* edges. For each indicative term, if the *Filtering* module returns a list of results as $R = \{(f_1, s_1), (f_2, s_2), ..., (f_n, s_n)\}$, where f_i is a fact and s_i is the relevancy score of that fact to the indicative term, we normalize the relevancy score s_i as:

$$s' = \frac{s_i - s_{min}}{s_{max} - s_{min}} \tag{1}$$

in that, s_{min} and s_{max} are the minimum and maximum relevancy scores reported respectively. In this way, the normalized relevancy scores will be in the range of [0,1], and we use this value as the weight of each *fact-term* edge in the graph.

3.4.2 Centrality Measurement

After constructing the graph, we analyze the graph to find the most important facts. To achieve this goal, we use *Closeness Centrality* [6] measure which is the average shortest-path distance to a node from all reachable nodes in the graph. When this measure for a fact is higher, it means that the fact is more relevant and similar to other facts, indicative terms, and queries, and so it implies that the fact is likely more important. We calculate the *Closeness Centrality* measure for the *fact node* u, as:

$$C_u = \frac{n-1}{\sum_{v=1}^{n-1} d(u,v)} \tag{2}$$

where n is the number of nodes have path to node u in the graph, and $d(u,v)$ is the shortest-path distance between node u and v in the graph. Additionally, when the algorithm figures out the shortest path between two nodes u and v, the distance between two adjacent nodes i and j in the graph is considered as $1 - w_{i,j}$, in that $w_{i,j}$ is the weight we assigned to the edge between two nodes in the graph generation step. It means that if two nodes are more relevant or similar, their distance is shorter. Finally, to calculate the importance score for each fact, we normalize the centrality measure C_u by using min-max normalization method as:

$$Importance(u) = \frac{C_u - C_{min}}{C_{max} - C_{min}} \tag{3}$$

where C_{min} and C_{max} are the minimum and maximum of centrality measures in the fact set. Therefore, the calculated importance for each fact is in the range of [0,1].

4 Experiments and Results

4.1 Experimental Setup

We executed our experiments on the dataset provided in CrisisFACTS track of TREC 2023 [1]. To implement our system we have used the following configurations:

- **Text to Fact**: We run our experiments in two fact generation settings: Method A and Method B discussed in section 3.1.
- **Query Extender**: We leveraged KeyBERT model [8] to extend the indicative terms.
- **Filtering**: We used *pyTerrier* [10] library to index the generated facts. Additionally, we configured the indexer to utilize *DFReeKLIM* weighting model [2] to score the retrieved facts.
- **Scoring**: In this module, to generate the embedding of each fact we used sentence transformer model *bert-base-nli-mean-tokens* [13]. Then we calculate the cosine similarity of embeddings to figure out the facts' similarity. Moreover, we utilized *Networkx* library [7] for generating the graph and calculating the closeness centrality of fact nodes. Also, we set $Sim_{min} = 0.5$ to prune the *fact-fact* and *fact-query* edges with similarities less than this value.

4.2 Evaluation Metrics

To evaluate the list of facts two types of evaluation metrics are used:

- **Metric Set 1- ROUGE-based Summarization Against Daily Summaries**: In this metric, the "summary" of the day's events are generated by combining the top-K important generated facts from each event-day pair. Then, using ROUGE-x and BERTScore, this summary is compared to existing summaries such as NIST-based assessments and Wikipedia summary.
- **Metric Set 2- Individual Fact-Matching Between Runs and Manual Fact Lists**: The second metric of evaluation is the individual fact matching of the generated facts during the runs with a manually curated fact list. Using a pooled collection of facts from all the participant runs, NIST assessors create a set of facts for each event-day pair. Assessors then match participant runs' output to each of these facts to evaluate comprehensiveness and redundancy ratio, which serve as alternates for recall and precision, respectively. In other words, for a given event day pair, there are the set of "gold standard" facts S obtained by the assessors, the set of facts F generated by the system, and the matching function $M(F,S)$ to measure the overlap between these two set of facts.

  The formula for calculating comprehensiveness (recall based) is:

$$C(F,S) = \frac{M(F,S)}{|S|} \tag{4}$$

The formula for calculating redundancy (precision based) is:

$$R(F, S) = \frac{M(F, S)}{|F|} \tag{5}$$

For each event, the average redundancy and the average comprehensiveness is calculated for all the days of the event.

Method	nist			wiki		
	Recall	Precision	F1 Score	Recall	Precision	F1 Score
A	0.594	0.600	0.597	0.471	0.422	0.445
B	**0.615**	**0.613**	**0.613**	**0.506**	**0.458**	**0.480**

Table 2: The mean performance of methods A and B using BERTScore metric.

Method	nist			wiki		
	Recall	Precision	F1 Score	Recall	Precision	F1 Score
A	0.173	**0.299**	0.211	0.169	**0.024**	**0.039**
B	**0.379**	0.293	**0.319**	**0.253**	0.016	0.028

Table 3: The mean performance of methods A and B using ROUGE metric.

Method	Redundancy	Comprehensiveness
A	0.305	0.058
B	**0.507**	**0.136**

Table 4: Average manual score for all runs of methods A and B.

4.3 Results

In this section, we will present and discuss the performance of the two methods (Method A and Method B) described earlier.

We have computed the mean BERTScore values (precision, recall, and F1 score) for all event days and displayed them in Table 2. As highlighted in the table, our method B outperforms method A by 2.68% and 7.87% for NIST-based evaluations and Wikipedia summaries, respectively, in terms of F1-score. Additionally, we have presented the ROUGE scores for methods A and B in Table 3, which also exhibits a similar trend in scores for NIST-based assessments (with a 51.18% increase in F1 score of method B compared to method A). However, our system's ROUGE-based performance against the Wikipedia summaries is comparatively low overall. Table 4 demonstrates the performance of our systems in terms of redundancy and comprehensiveness, as detailed in Section 4.2. We can observe a better performance of method B compared to method A in redundancy and comprehensiveness metrics as well.

Based on our analysis, it can be inferred that the results obtained using method B are comparatively better than those obtained using method A. This might be due to the fact that the scoring module handles the facts generated by method B better than those generated by method A.

5 Conclusion

In this work, we proposed a system for extracting and scoring critical information to generate informative summaries for the disaster response agencies during crises. Our system analyzes multiple streams of data from different sources of online data to extract crisis-related facts. To achieve this goal, our system expands the list of indicative terms provided by the disaster response agencies and leverages it to filter the facts extracted by existing fact extraction methods. Although we have used two extractive methods for generating the facts, our system does not rely on these fact extraction methods only, so it can be used to work with any extractive and abstractive fact generation methods. Additionally, since the summary of the crisis-related events is generated by truncating top-K most important facts, ranking the extracted facts is one of the critical parts of summary generation task. In our system, we introduce an integrated content-graph analysis that analyses a graph containing the relevancy of facts to each other, facts to queries, and facts to indicative terms to score facts. This enables our system to prioritize the most important facts. Furthermore, our experiments demonstrate a good performance of the proposed system in analyzing the dataset of the TREC 2023 CrisisFACTS track [1].

References

[1] 2023 TREC CrisisFACTS Track, 2023. https://crisisfacts.github.io/.

[2] Amati, G., Amodeo, G., Bianchi, M., Celi, A., De Nicola, C., Flammini, M., Gaibisso, C., Gambosi, G., Marcone, G., et al. Fub, iasi-cnr, univaq at trec 2011 microblog track.

[3] Cabot, P.-L. H., and Navigli, R. Rebel: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021* (2021), pp. 2370–2381.

[4] Chourdakis, E. T. Claucy. https://github.com/mmxgn/spacy-clausie, 2018.

[5] Del Corro, L., and Gemulla, R. Clausie: clause-based open information extraction. In *Proceedings of the 22nd international conference on World Wide Web* (2013), pp. 355–366.

[6] Freeman, L. C. Centrality in social networks conceptual clarification. *Social Networks 1*, 3 (1978), 215–239.

[7] Hagberg, A., and Conway, D. Networkx: Network analysis with python. *URL: https://networkx. github. io* (2020).

[8] Hernandez Barrera, A. O., Montero Valverde, J. A., Hernández Hernández, J. L., Martínez-Arroyo, M., and De la Cruz Gámez, E. Automated creation of a repository for learning words in the area of computer science by keyword extraction methods and text classification. In *International Conference on Technologies and Innovation* (2023), Springer, pp. 186–203.

[9] Khan, M. Q., Shahid, A., Uddin, M. I., Roman, M., Alharbi, A., Alosaimi, W., Almalki, J., and Alshahrani, S. M. Impact analysis of keyword extraction using contextual word embedding. *PeerJ Computer Science 8* (2022), e967.

[10] Macdonald, C., Tonellotto, N., MacAvaney, S., and Ounis, I. Pyterrier: Declarative experimentation in python from bm25 to dense retrieval. In *Proceedings of the 30th acm international conference on information & knowledge management* (2021), pp. 4526–4533.

[11] Miao, W., Zhang, G., Bai, Y., and Cai, D. Improving accuracy of key information acquisition for social media text summarization. In *2019 IEEE International Conferences on Ubiquitous Computing & Communications (IUCC) and Data Science and Computational Intelligence (DSCI) and Smart Computing, Networking and Services (SmartCNS)* (2019), IEEE, pp. 408–415.

[12] Purohit, H., Castillo, C., and Pandey, R. Ranking and grouping social media requests for emergency services using serviceability model. *Social Network Analysis and Mining 10* (2020), 1–17.

[13] Reimers, N., and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing* (11 2019), Association for Computational Linguistics.