



Article

Designing a Prototype Platform for Real-Time Event Extraction: A Scalable Natural Language Processing and Data Mining Approach

Mihai-Constantin Avornicului^{1,2}, Vasile Paul Bresfelean^{1,*} , Silviu-Claudiu Popa¹, Norbert Forman² and Calin-Adrian Comes^{3,4} 

¹ Faculty of Economics and Business Administration, Babeş-Bolyai University, 400084 Cluj-Napoca, Romania; mihai.avornicului@econ.ubbcluj.ro (M.-C.A.); silviu.popa@econ.ubbcluj.ro (S.-C.P.)

² Faculty of Finance and Accountancy, Budapest Business University, 1149 Budapest, Hungary; forman.norbert@uni-bge.hu

³ Faculty of Economics and Law, “George Emil Palade” University of Medicine, Pharmacy, Sciences and Technology, 540139 Targu Mures, Romania; calin.comes@umfst.ro

⁴ Institute of National Economy, Romanian Academy, 050711 Bucharest, Romania

* Correspondence: paul.bresfelean@econ.ubbcluj.ro

Abstract: In this paper, we present a modular, high-performance prototype platform for real-time event extraction, designed to address key challenges in processing large volumes of unstructured data across applications like crisis management, social media monitoring and news aggregation. The prototype integrates advanced natural language processing (NLP) techniques (Term Frequency–Inverse Document Frequency (TF-IDF), Latent Semantic Indexing (LSI), Named Entity Recognition (NER)) with data mining strategies to improve precision in relevance scoring, clustering and entity extraction. The platform is designed to handle real-time constraints in an efficient manner, by combining TF-IDF, LSI and NER into a hybrid pipeline. Unlike the transformer-based architectures that often struggle with latency, our prototype is scalable and flexible enough to support various domains like disaster management and social media monitoring. The initial quantitative and qualitative evaluations demonstrate the platform’s efficiency, accuracy, scalability, and are validated by metrics like F1-score, response time, and user satisfaction. Its design has a balance between fast computation and precise semantic analysis, and this can make it effective for applications that necessitate rapid processing. This prototype offers a robust foundation for high-frequency data processing, adaptable and scalable for real-time scenarios. In our future work, we will further explore contextual understanding, scalability through microservices and cross-platform data fusion for expanded event coverage.

Keywords: real-time event extraction; data mining; natural language processing (NLP); term frequency–inverse document frequency (TF-IDF); latent semantic indexing (LSI); named entity recognition (NER)



Citation: Avornicului, M.-C.; Bresfelean, V.P.; Popa, S.-C.; Forman, N.; Comes, C.-A. Designing a Prototype Platform for Real-Time Event Extraction: A Scalable Natural Language Processing and Data Mining Approach. *Electronics* **2024**, *13*, 4938. <https://doi.org/10.3390/electronics13244938>

Academic Editor: Arkaitz Zubiaga

Received: 18 November 2024

Revised: 10 December 2024

Accepted: 12 December 2024

Published: 14 December 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Due to the advent of the digital revolution, there has been an unprecedented surge in the volume, speed and diversity of online information. Many sectors now rely on real-time event extraction systems; these include sports analytics, social media monitoring, crisis management and news. In order to identify critical events as they happen, these systems are designed to gather continuously, filter and then interpret vast streams of unstructured data. Platforms that can process incoming data with minimal latency are necessary for real-time event extraction, as opposed to traditional data mining methods that examine historical datasets for trends and patterns [1,2]. This can ensure that users receive immediate and actionable insights. When taking into consideration this prototype, real time refers to the conceptual ability to process and provide output with a minimal delay. This is categorized into soft real-time (for example, social media monitoring where minor delays are acceptable) and hard real-time (emergency management where immediate responses are important)

cases. The current design relies on soft real-time use cases, but the future iterations in the next development stages should explore hard real-time scalability.

Despite the large amount of research on event extraction, current systems frequently fail in real-time scenarios due to various major challenges. A few illustrations are as follows:

Terminology consistency—Variations in terminology (such as “digital platform”, “on-line platform”, or “internet platform”) often result in ambiguity and limit comparative analyses across studies. The development of unified frameworks for event extraction is complicated by the possibility that each term may refer to various architectures or functionalities [3,4].

Relevance scoring and adaptability—Many platforms lack robust mechanisms for dynamically scoring and adjusting relevance when events unfold in real time. An accurate relevance scoring is important for ensuring that extracted events are timely and contextually significant in order to support critical applications, like emergency responses or market monitoring [5–8].

Scalability and efficiency—Processing high volumes of data in real time has some computational demands. More sophisticated natural language processing (NLP) and machine learning models could struggle with scalability under high-frequency data inflows, while systems based on keyword-matching techniques frequently suffer due to poor precision and high noise [7,9–11]. In order to better capture complex semantics, for example, transformer-based models need additional adaptation to account for real-time constraints [12,13].

1.1. Objectives of the Study

In response to these challenges, in this study, we introduce a prototype event extraction platform, designed to address challenges in real-time data processing across diverse applications. The platform opens the path for future iterations and emphasizes showing foundational capabilities. The primary objectives of this prototype are as follows:

Dynamic relevance scoring—By integrating TF-IDF and LSI, the platform provides adaptive, content-based relevance scoring. Basic Web Structure Mining, including hyperlink analysis, continuously assesses source credibility; these features are streamlined to fit with the experimental character of the platform.

Scalable real-time processing—Through efficient caching, robust indexing and modular architecture, the platform manages varying data volumes while remaining responsive. These elements enable responsiveness even in high-frequency scenarios and show the scalability potential of the system in controlled environments.

A straightforward user-centric query interface is developed, is functional and intuitive and allows users to filter event data by parameters like date, location and event type, supporting a user-centered approach to event retrieval for both general and specialized users.

1.2. Main Contributions

First, we introduce an integrated data processing framework to combine Web Content and Web Structure Mining with NLP-based entity recognition. Even for some high-frequency and high-volume environments, this establishes itself as a comprehensive approach that enhances the accuracy of event detection and relevance ranking.

Second, we conduct a systematic quality evaluation based on the ISO 9126 standard. [14]. This evaluation focuses on attributes such as cohesion, modifiability and reliability, thus ensuring that the prototype meets high standards for operational robustness.

Third, we provide a quantitative performance analysis based on metrics like precision, recall, F1-score, response time and user satisfaction. All of these confirm the platform’s accuracy and scalability and position it as a promising solution for real-time event extraction and information retrieval.

This platform was developed as an internal prototype to explore real-time event extraction capabilities. Currently deployed within a controlled-access environment, it has been tested with real and synthetic datasets to ensure robustness and scalability. For future

enhanced versions, we desire to make portions of the code available on GitHub, so as to foster research collaboration.

By addressing the limitations of current event extraction systems, this prototype advances real-time data mining and offers a structured, scalable and efficient foundation for decision-making in time-sensitive domains.

2. Theoretical Background

For the internal research prototype for event extraction, we employed established methods in natural language processing, data mining and information retrieval, in order to meet the stringent demands of real-time event extraction applications. In the next subsections, we outline the theoretical techniques supporting the platform's accuracy, scalability and responsiveness, like TF-IDF, LSI and NER. Each of these components has its importance to the platform's design and helps with the accurate processing, filtering and ranking of data pertaining to events. In the context of the current prototype, all of these methods form a modular pipeline. TF-IDF serves as an initial filter used to prioritize event-relevant data, LSI enables semantic clustering for improved relevance scoring, and NER identifies the key entities for detailed event categorization. All of these address, together, the challenges of real-time event extraction in a conceptual framework, and future iterations would further refine them for practical applications.

2.1. Term Frequency–Inverse Document Frequency (TF-IDF)

In order to assign weights to terms in a document based on their importance relative to a larger corpus, TF-IDF represents a foundational information retrieval technique [15,16]. In real-time applications, TF-IDF is instrumental in filtering large datasets and allows the platform to prioritize documents containing terms that are highly relevant to specific events, reducing noise from common, non-specific words. In this current prototype, TF-IDF represents the first step in the pipeline which reduces irrelevant data efficiently and ensures that only documents with high event relevance proceed to deeper processing stages like the LSI and NER.

The TF-IDF weight for a term t in a document d within a corpus of N documents is calculated as follows [17]:

$$\text{TF-IDF}(t, d) = \text{tf}(t, d) \times \log\left(\frac{N}{\text{df}(t)}\right) \quad (1)$$

where

$\text{tf}(t, d)$ is the term frequency, i.e., the number of times t appears in d ;

$\text{df}(t)$ represents the document frequency or the number of documents in which t appears;

N is the total number of documents in the corpus.

As for the application in event extraction, in our platform, TF-IDF serves as an initial filtering mechanism to facilitate rapid, high-precision sorting of documents by their relevance to particular events. This technique is efficient in managing vast, unstructured data streams and ensures that high-relevance data advance to further processing stages [1,15].

2.2. Latent Semantic Indexing (LSI) and Dimensionality Reduction

LSI captures semantic relationships between terms, and it may effectively address synonymy and polysemy, which complicates text search [18,19]. With LSI, the platform is able to identify conceptually related terms regardless of source-to-source terminology variation by utilizing Singular Value Decomposition (SVD) for dimensionality reduction. For this prototype, LSI operates as a semantic layer that groups related documents in order to support accurate event clustering and retrieval. It has the ability to handle varied terminologies, and this makes it important in unifying data across diverse sources.

LSI employs SVD to reduce the high-dimensional term–document matrix A into a lower-dimensional space:

$$A = U\Sigma V^T \quad (2)$$

where

U and V are orthogonal matrices;

Σ is a diagonal matrix containing singular values.

By retaining only the top k singular values, LSI captures central semantic structures and balances computational efficiency with relevance accuracy [18,19].

In our internal prototype, LSI is applied following TF-IDF to semantically group similar documents, thus enabling effective clustering and retrieval of conceptually related information. This layer of analysis is particularly valuable in contexts such as crisis monitoring, where different sources might describe the same event using varied language (e.g., “flood” vs. “inundation”) [18].

2.3. Named Entity Recognition (NER)

NER is an NLP method used to detect and classify entities within text, like names, dates and locations. NER enhances the platform’s capacity to isolate specific event-related entities and provides a finer level of detail for accurate event categorization [20,21]. Some recent studies in NER emphasize the use of transformer-based architectures that achieve state-of-the-art performance to identify complex entities from diverse datasets [22,23]. In this prototype, NER represents the final stage, and extracts and categorizes entities like names, dates and locations that are essential for actionable event insights. Its integration is important because the data passed through TF-IDF and LSI are refined into specific and event-critical outputs.

The NER model that is used in our prototype is machine learning-based and integrates Conditional Random Fields (CRFs) and Bidirectional Long Short-Term Memory (BiLSTM) networks:

- CRFs label word sequences and assign tags to each token based on its context within a sentence, making them highly effective for entity boundary detection [21].
- BiLSTMs process text in both forward and backward directions, allowing the model to capture dependencies across entire sentences and enhancing accuracy in complex structures [21].

Because it combines CRF and BiLSTM models, the prototype achieves high accuracy in identifying entities relevant to specific events (like names, dates and geographic locations), but also significantly enhances the relevance and contextual accuracy of extracted data [20,21]. For future enhanced versions, the integration with recent techniques like Flair embeddings and transformer-based NER models could further improve entity extraction by leveraging contextualized word embeddings [24].

2.4. Alternative Techniques

Word embedding (Word2Vec, GloVe) and transformer-based models (BERT) provide a few distinct alternatives to the methods used in our prototype. Taking into consideration the requirements for immediate use, they have benefits and limitations:

- Word embedding models like Word2Vec and GloVe generate vector representations that capture word semantics based on co-occurrence patterns [25,26]. Their lack of interpretability and higher computational demands make them somehow less ideal for high-speed, real-time processing needed in our platform. Other more recent research studies have explored contextual embeddings derived from static word vectors to improve their utility in domain-specific applications [27,28].
- Transformer-based architectures, like BERT, seem to offer superior contextual embeddings by considering bidirectional context [12]. But their significant computational load is a limitation, mainly for high-frequency, real-time environments. Other advanced transformer variants like RoBERTa [29] and DistilBERT [30] focus on optimiz-

ing computational efficiency and maintaining high accuracy; this makes them more suitable for limited-resource scenarios. Unified frameworks (like T5) extend these and cover a much larger range of NLP tasks, but they remain computationally expensive in real-time environments [31].

Due to the platform's operational constraints, TF-IDF, LSI and machine learning-based NER were chosen for their balance of accuracy, scalability and computational efficiency.

2.5. Comparison with Existing Works

Real-time event extraction has come a long way, and there are a few systems tackling issues such as scalability, adaptability and latency. The current prototype builds on this and presents some features that integrate multiple data sources and flexibility for different scenarios.

Systems such as DEES [32,33] and TwitterNews+ [33] show modular designs that can work well for specific applications such as, for example, disaster management or single-source social media streams. These systems perform well in their areas, but they often stick to the predefined data types. Our prototype takes one step further and uses LSI and NER to work across multiple domains and to handle various input types.

Other researchers combine spatial and temporal data for event detection. For example, Deep-Eware [34] and the framework developed by [35] include these dimensions but mostly rely on geotagged data. Our prototype can process non-geotagged inputs, and it is more flexible in dealing with real-time data from diverse sources.

Transformer-based models are known for their precision in event extraction, as they are presented by [36,37]. But these models often require important computational resources, and this would make them unsuitable for real-time use. Our prototype can avoid this bottleneck based on TF-IDF and LSI, with a practical balance between speed and accuracy, in particular for time-critical applications.

There are also other platforms that show the importance of flexibility and scalability, like the ones presented by [38] and implemented by [39]. We share these goals, but our design also focuses on the challenges of real-time multi-source event extraction. Other researchers [9] explore advanced techniques like graph convolution and attention mechanisms to process noisy or unstructured data. Our platform uses lightweight methods to handle complex input streams while also keeping the processing efficient.

3. Platform Design and Architecture

Prototype preliminary design emphasizes efficiency, accuracy and scalability, which are achieved through a modular architecture based on its three components: data retrieval, document processing and query processing (Figure 1). NLP techniques and a microservice-based deployment approach confirm the platform's capability to operate reliably under high-demand conditions. Other specific details are presented in Appendices A.1–A.3.

3.1. Data Retrieval Component

The data retrieval component regularly obtains information from a ranked list of sources based on retrieval schedules that change based on how often each source updates. Focusing more on high-priority sources more often shows that resources are being used effectively. This component was built on Python libraries, such as Scrapy and BeautifulSoup for web scraping, and Docker containers which provide isolated environments for each crawler to enhance modularity and scalability.

Each data source is assigned a crawl interval in accordance with its typical update frequency. For example, high-priority sources could be checked every few minutes, but lower-priority sources are revisited less frequently. The scheduling interval is adjusted dynamically to balance timeliness with resource efficiency, as illustrated in the data retrieval pipeline. Each crawler runs independently in a Docker container, allows an easy scaling based on the data load and minimizes resource contention.

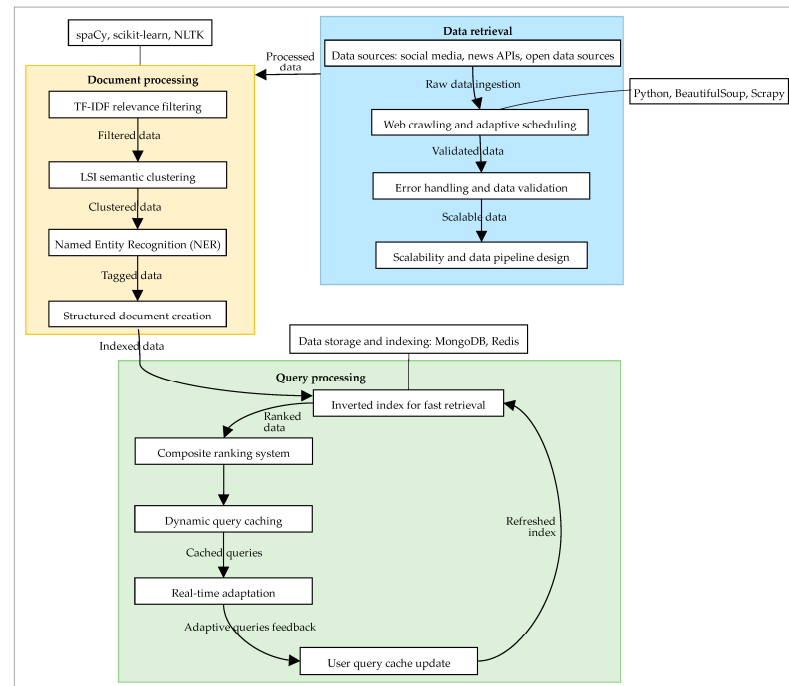


Figure 1. The architecture of the internal prototype for the real-time event extraction platform. Legend: blue—data retrieval; yellow—document processing; green—query processing; solid arrows—data flow; looped arrows—feedback.

In order to maintain data quality and reliability, the data retrieval component includes basic error-handling mechanisms. If a source is unavailable or returns empty data, the system logs the issue and skips to the next source, thus avoiding unnecessary retries that could strain resources. Data preprocessing involves standardizing formats, removing noise and ensuring consistency so as to create a validated flow of data for downstream processing. NLTK and spaCy libraries handle basic text processing tasks, like noise reduction and deduplication [16].

Although the platform is a prototype, the data retrieval component is deployed in containerized environments to ensure modularity and scalability. Each data retrieval task can operate independently, and scalability tests can be conducted in order to simulate load distribution during high-demand events [40].

In the next figure (Figure 2), we present the `data_retrieval_pipeline` function that retrieves data, adjusts crawl intervals and logs errors as necessary. Processed data are returned in a standardized format, ready for further analysis.

3.2. Document Processing Component

The document processing component is designed to transform unstructured data into a structured format that is optimized for relevance ranking and event-specific extraction. This component sequentially applies TF-IDF relevance filtering, NER and LSI, each with its own contribution to the selection, clustering and tagging of relevant content for efficient processing and retrieval [15,18].

TF-IDF scores are calculated by using Python's scikit-learn library. A global TF-IDF matrix is used to calculate relevance scores for each document, which prioritizes documents with high event-specific term weights. Only documents that exceed a predefined relevance threshold are passed on for further processing, thus improving efficiency and relevance [16].

For the documents that pass the relevance threshold, NER is applied by using spaCy and NLTK libraries. This step identifies and categorizes key entities (for example, names, dates and locations) within the document. NER extraction uses models that incorporate Conditional Random Fields (CRFs) and BiLSTM networks, which easily enhance accu-

racy in identifying entity boundaries and enriching the contextual tagging of document content [20,21].

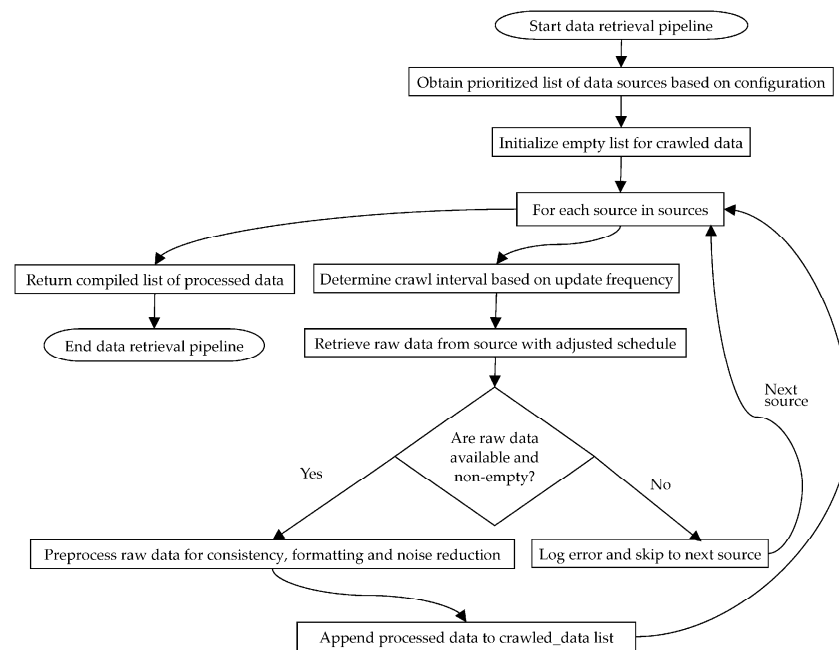


Figure 2. Data retrieval pipeline diagram.

After NER, LSI is used to give the document a semantic representation. An LSI model that has already been trained clusters documents based on conceptual shared content. This lets the system recognize and link semantically related terms across documents. This semantic representation enhances further clustering and retrieval relevance [19].

Each processed document is organized into a structured format to embed both LSI-generated content and identified entities. This structure supports efficient indexing by keywords, entities and topics, and makes documents ready for retrieval and relevance ranking in MongoDB.

In Figure 3, we present the document processing pipeline that integrates TF-IDF for initial filtering, NER for entity extraction and LSI for semantic clustering, thus ensuring that only relevant, contextually enriched documents are structured and stored for efficient ranking and retrieval.

3.3. Query Processing Component

The query processing component supports efficient user interaction within the prototype and handles user queries with basic caching, inverted indexing and simplified ranking.

The prototype utilizes MongoDB to create a simple inverted index that links query terms to relevant documents, thus enabling fast keyword-based document retrieval [1].

For the document ranking, the prototype combines relevance scores (based on TF-IDF and LSI) with basic authority measures. The ranking system uses fixed weights for each criterion and offers a straightforward evaluation of relevance.

The most frequently used queries are cached temporarily to improve response times for recurring requests. The cache is updated periodically but does not adapt in real time, due to the prototype's experimental nature [25].

This component (Figure 4) processes user queries by checking the cache, retrieving relevant documents through the inverted index and applying a simplified ranking method. It shows a responsive experience in our prototype version, then it sets the stage for more advanced features in future versions.

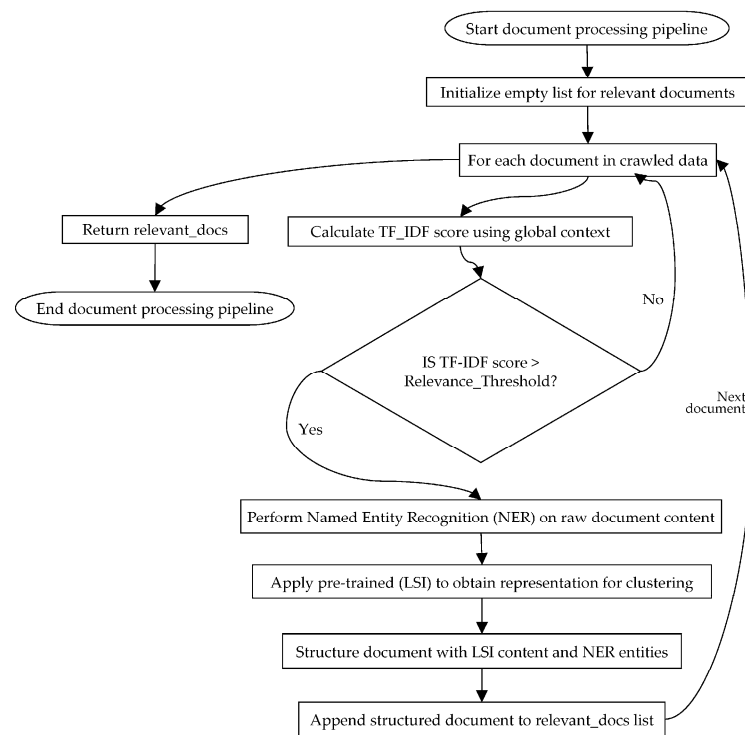


Figure 3. Document processing pipeline diagram.

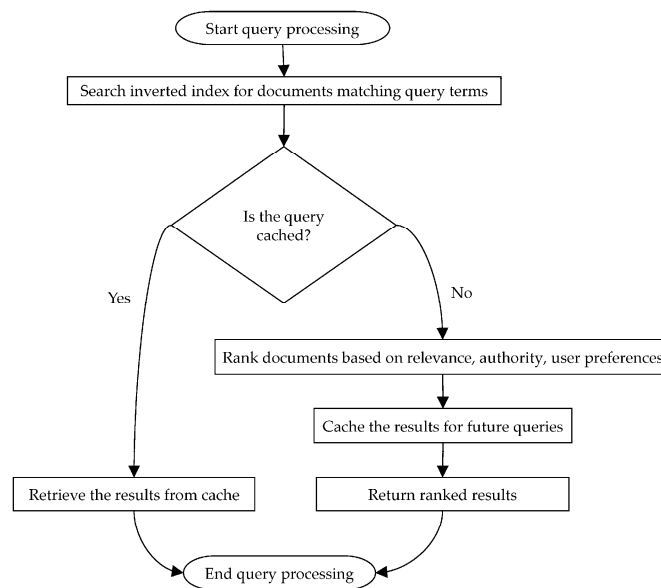


Figure 4. Query processing decision pipeline diagram.

3.4. Technologies and Software Stack

The platform is built on a software stack that is optimized for large data volumes and real-time performance, based on Python and specialized libraries for NLP and data processing. We highlight the main technologies utilized for our prototype:

- Programming language: Python 3.8 or higher used for data processing and NLP;
- Libraries:
 - NLTK and spaCy for NLP, to handle TF-IDF, LSI and NER operations;
 - scikit-learn for machine learning and clustering;
- Database: MongoDB for document storage, indexing and querying;

- Containerization: Docker for containerized deployment, with optional Kubernetes and cloud options for scalability, though not necessary for the initial prototyping stages;
- Caching: Redis for caching high-demand queries in distributed setups, optional at this stage;
- Other technical specifications and requirements are presented in Appendix A.1.

3.5. Error Handling and Redundancy

Even during high-demand situations, the prototype can efficiently operate and remain also fault-tolerant because of its modular design. Every one of the components works independently but also connects based on simple API calls, which makes the system resilient to failures in one part. We present some of the main features as follows:

- Independent module operation—This means that each module performs its tasks without relying on the others. For example, when the document processing module encounters an issue, the query processing module still continues to provide users with the previously indexed data, which ensures continuous system responsiveness.
- Retry and fallback mechanisms—This means that when the system cannot access a primary source, it retries with increasing delays so as to avoid overloading the source. If the source remains inaccessible, then the platform switches to cached data or secondary sources. For example, if a news feed becomes unavailable, then the system utilizes recent cached updates or alternative feeds.
- Real-time error tracking—This means that failures are logged with details such as source ID, failure type and timestamp, and the administrators can quickly diagnose and address these issues by using these records. The alerts for critical failures ensure that every high-priority problem can be resolved promptly.
- Redundant data flow—This helps ensure the system's reliability. For example, when a data source becomes unavailable, then cached data or secondary sources fill the gap, a design that minimizes disruptions and maintains operational consistency.

With the purpose of ensuring the reliability of user behavior metrics like CTR and engagement, our current prototype validates these metrics to detect anomalies or manipulative patterns such as, for example, fake clicks or artificially inflated engagement (see Appendix A.10 for implementation details). These mechanisms are presently tested in controlled environments, and in our future work we will refine their robustness in real-world scenarios.

3.6. Testing Environment and Datasets

The evaluation of the capabilities of the platform was based on a combination of real-world datasets and simulated datasets which were designed to simulate high-traffic scenarios and a variety of use cases. Publicly available datasets such as, for example, some crisis-related news and social media streams from sources like Kaggle [41], GDELT event database [42], etc., were the foundation for testing event extraction functionality. In order to explore the platform's adaptability, real-time data were also gathered using APIs, like the Twitter API [43] and Reddit's Pushshift API [44], to assess its ability to handle live inputs. Custom datasets were generated in controlled environments to simulate the high-volume traffic events for scalability and response time evaluations under stress conditions. Details on the hyperparameters used during the prototyping phase are in Appendix A.7. The chosen datasets were selected to represent some varied domains, like disaster response, breaking news and real-time sports updates. It looks like the results are favorable, but more testing needs to be carried out in a real-world setting that is not controlled. This will help determine if the prototype is ready to be deployed.

3.7. Security and Data Privacy

In order to comply with the industry standards and the regulatory frameworks, security and privacy measures have been integrated into the platform [3]. All of these measures were implemented due to the sensitive nature of real-time data processing. For

example, in the European Union, there is the General Data Protection Regulation (GDPR) that requires that sensitive information, such as the one obtained from social media or other public platforms, should be anonymized. In order to ensure that data usage is both ethical and compliant, there are mechanisms in place to respect restrictions made by third-party APIs, such as rate limits. Caching practices have been developed in order to protect users' privacy and to prevent the storage of personal or identifiable information beyond what is required for active sessions. The publicly available datasets and the APIs that were used for testing offer limited access to sensitive user information, in accordance with their platforms' terms of service. Our prototype does not store nor analyze any personally identifiable information, due to the fact that the datasets are pre-anonymized or do not include user-level metadata. Future iterations of the platform will implement additional safeguards such as, for example, anonymization during preprocessing and automated monitoring to comply precisely with data privacy standards such as GDPR.

Through the combination of all of these factors, secure interactions with a wide variety of data sources are made possible, particularly in situations that involve potentially sensitive information, such as crisis management. Additionally, as the prototype develops, these important features will be refined further.

3.8. Scaling Architecture and Load Balancing

The platform's architecture has been designed with scalability in mind, so that it can handle high data volumes and peak user demand when intensive events happen. For the present prototyping stage, scaling is supported through Docker containers, which allow modular, independent operation of system components. In larger testing environments, Kubernetes could be optionally employed in order to orchestrate these containers across multiple servers, for efficient distribution of processing tasks and to facilitate horizontal scaling when required. Scalability experiments with Kubernetes-based orchestration show interesting potential for dynamic scaling under high-demand scenarios (configuration sample in Appendix A.5).

Load balancing, a core feature even in this prototype phase, is implemented via Nginx that routes incoming requests across multiple service instances. This configuration optimizes resource utilization and maintains low latency during peak demand. The system's autoscaling capabilities have been tested under controlled conditions, and further real-world evaluations are needed so as to refine its ability to adapt dynamically to fluctuating loads like those triggered by major news events or crises.

4. Ranking and Scoring

Finding relevant data and putting them in a meaningful order are both important for an event extraction platform to work well. Our platform uses a composite scoring model to accurately put information in order of importance for users. This model ranks documents based on three main criteria:

- Content relevance scoring that determines alignment between document content and event-related keywords;
- Authority-based ranking that evaluates the credibility of a document based on the reliability of its source;
- User behavior analysis that adjusts rankings dynamically and analyzes user interactions, like click-through rates and engagement duration.

4.1. Content Relevance Scoring

The presence and significance of event-related keywords and thematic structures in a document are analyzed to determine its content relevance. The platform employs LSI to discover thematic relationships between terms and TF-IDF to weigh the importance of keywords in context.

TF-IDF assigns higher weights to terms that appear frequently in a document but are uncommon across the dataset, and it also highlights the terms that are particularly

relevant to particular events. This is complemented by LSI which clusters documents based on semantic coherence. This enables the platform to detect similar themes even when terminology varies (for example, “earthquake” vs. “seismic activity”). Transformer-based models provide precise results but depend on complex computations that create delays. Our approach reduces processing time with TF-IDF and LSI while also maintaining reliable accuracy, which is a better way for applications that require quick decisions, like monitoring social media or responding to crises.

For the nuanced requirements of real-time event extraction, this combined approach ensures that relevance scoring accounts for both lexical frequency and deeper thematic alignment. In order to significantly enhance ranking accuracy for complex queries, and for a deeper contextual understanding, future implementations should use transformer-based embeddings to improve thematic coherence.

4.2. Authority-Based Ranking

The prototype platform includes authority-based ranking to assess the reliability of documents based on their sources to reduce disinformation and highlight trustworthy data. During this process, the following factors are considered:

- **Hyperlink analysis**—This is based on the ideas behind the PageRank algorithm and gives documents that are referenced or linked by credible sources more authority. Higher authority scores are awarded to documents that are frequently linked to or cited by trusted entities.
- **Source reputation scoring**—Sources are rated dynamically based on metrics for reliability, consistency and user engagement. For example, more reputable sources, like scientific journals and well-established news outlets, tend to receive higher ratings than less reputable sources. In order to allow the sources to adapt to the digital landscape, the reputation of each source is updated dynamically based on recent performance metrics.

Enhanced versions will incorporate sentiment analysis and social media credibility signals which will complement traditional authority-based methods, mainly for semi-structured data sources like social media (Appendix A.5). This is aimed at improving the credibility assessment of data from such platforms where the traditional metrics are often less consistent.

For example, sentiment analysis can prioritize reliable sources in real time by identifying patterns of extreme sentiment or potential biases. This could help filter and rank trustworthy content from semi-structured sources like social media during a crisis, and aid decision-making when timely information is needed. Aspects of sentiment analysis are in Appendix A.5.

4.3. User Behavior Analysis

User interactions present some important valuable insights into the perceived relevance and the quality of a document. In order to adjust document rankings dynamically and to align the results with user preferences and trends, our platform analyzes user behavior metrics.

- **Click-through rate (CTR)**—Documents with higher CTRs are often the ones more relevant to users. Those that have demonstrated user interest by tracking how frequently users click on certain documents can be prioritized by our current prototype.
- **Engagement duration**—Longer engagement times suggest that a document is informative and valuable. Therefore, documents with higher engagement durations receive higher rankings.

This feedback loop makes sure that the platform remains responsive to user preferences and trends. Also, it dynamically adjusts rankings to maintain alignment with real-time interests and trending events.

4.4. Composite Scoring and Normalization

A composite score combining content relevance, authority and user behavior dictates the final document ranking [16,45]. The platform uses a weighted sum model and in the context of the query or application, each criterion is assigned a weight based on its importance.

4.4.1. Composite Scoring Formula

$$Rd = \alpha \cdot S_{\text{content}}(d) + \beta \cdot S_{\text{authority}}(d) + \gamma \cdot S_{\text{behavior}}(d) \quad (3)$$

where

$S_{\text{content}}(d)$ is the content relevance score;

$S_{\text{authority}}(d)$ is the authority-based score;

$S_{\text{behavior}}(d)$ is the user behavior score;

α , β and γ are weights tuned based on platform requirements.

The platform adjusts these weights dynamically based on user preferences and query context. In a crisis situation, authority-based ranking could have a higher weight to ensure that accurate information is prioritized, but in entertainment news, user behavior could have a greater influence.

4.4.2. Normalization of Component Scores

With the purpose that every component ensures consistent contributions, the platform normalizes every score to a range between 0 and 1. Normalization allows the platform to weigh each score proportionately and also prevents any criterion from dominating the composite score [16]. The normalization formula for a score S is as follows:

$$S' = \frac{S - S_{\min}}{S_{\max} - S_{\min}} \quad (4)$$

where

S' is the normalized score;

S_{\min} and S_{\max} represent the minimum and maximum values of S , respectively.

4.5. Ranking Workflow and Adaptability

All these three scoring criteria are combined in the ranking process into one simple streamlined pipeline. Figure 5 shows a graphic depiction of the ranking process concerning the way every component adds to the overall composite score. It presents a linear flow from the initial document set to the final ranking. Content relevance, authority and user behavior scores are calculated independently before they are integrated into a composite score, which determines the document's rank.

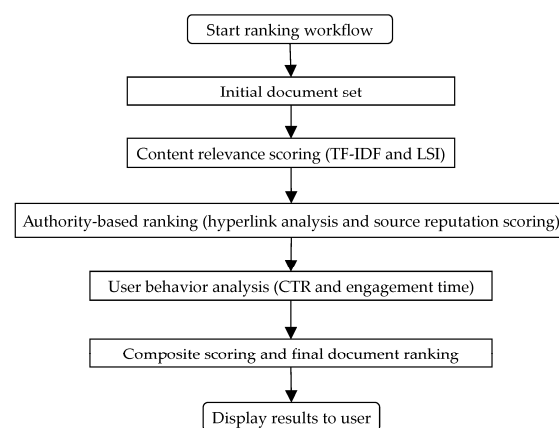


Figure 5. Ranking workflow.

4.6. Real-Time Adaptability and Caching Mechanism

The system maximizes responsiveness by including a cache for frequently accessed queries. This mechanism reduces processing delays and lets precomputed rankings be immediately retrieved, thus playing a significant role. The platform applies TF-IDF and LSI for filtering and relevance scoring so as to ensure minimal latency. These methods are able to simplify data processing and reduce the computational costs of transformer-based models. This approach ensures fast and reliable responses and keeps the system efficient during periods of high demand.

The periodic adjustment of the weights (α , β , γ) helps the platform fit changing data patterns and user interactions. The caching mechanism stores popular queries and their results; therefore, it eliminates the need to recalculate rankings for recurring queries. This is particularly effective during high-traffic events. The dynamic weight adjustment process ensures that content relevance, authority and behavior metrics remain balanced. The present testing shows these characteristics in controlled prototype environments; more iterations will be required to improve them to handle a wider spectrum of possibilities.

5. Quantitative Performance Evaluation

We conducted a preliminary assessment across key metrics: precision, recall, F1-score, response time and user satisfaction. Although it is in the prototyping phase, this quantitative analysis is able to show preliminary insights into the platform's effectiveness in retrieving pertinent event data under different conditions.

5.1. Accuracy Metrics: Precision, Recall, and F1-Score

Accuracy metrics are necessary to assess the platform's capacity to obtain pertinent information while minimizing irrelevant material [16]. Precision, recall and the F1-score provide, together, a comprehensive assessment of retrieval accuracy across various event categories (e.g., news, sports, crisis events).

Precision quantifies the proportion of relevant documents retrieved. The increased precision signifies a robust capacity to filter out irrelevant data.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (5)$$

Recall measures the percentage of relevant documents that were successfully retrieved, and it indicates the system's ability to gather all the necessary information.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (6)$$

F1-score assesses retrieval accuracy and is calculated as the harmonic mean of precision and recall. It guarantees high relevance and thorough coverage and helps especially in balancing precision and recall [16].

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (7)$$

In the evaluation, we conducted tests across a sample dataset of 1000 documents from three main categories: news, sports and crisis events. Table 1 shows the average results across categories and shows the platform's ability to maintain consistent accuracy across these diverse domains.

The high F1-scores across categories confirm the platform's accuracy and relevance in diverse, real-time scenarios, and also reinforce its utility in handling a broad range of high-stakes applications [6]. An ablation study was also conducted in order to understand the role of each module in the platform's performance and the results are presented in Appendix A.8.

Table 1. Average results across categories based on the tests.

Category	Precision	Recall	F1-Score
News	0.84	0.83	0.83
Sports	0.81	0.79	0.80
Crisis Events	0.87	0.85	0.86
Average	0.84	0.82	0.83

In order to better understand the platform’s performance, we compared it with two established systems: Deep-Eware [34] and TwitterNews+ [33]. We chose these systems because they are recognized benchmarks in event detection that could offer us some insights into several methodologies and outcomes. Deep-Eware [34] relied on 8137 annotated tweets, and it excelled in predefined well-structured topics like “incidents” (with an F1-score of 0.97) and “economy” (an F1-score of 0.92), which show a strong capacity to categorize data where topic boundaries are clear. However, the fact that it relies on fixed clusters could possibly make it somehow less effective for noisy or ambiguous data. TwitterNews+ [33] was evaluated on the Events2012 corpus and it demonstrated a balanced performance with an F1-score of 0.92. It seemed to perform well in identifying major and minor events, and this was due to the incremental clustering and noise-handling methods. For the comparison, we recalculated its F1-score based on the previous Formula (7) applied in our study.

In comparison, our prototype achieved an average F1-score of 0.83 across diverse topics of news, sports and crisis events (Table 1). It is able to maintain a balance between precision and recall across diverse datasets, although it encountered some challenges when it handled overlapping categories, and this requires further developments and refinements of the prototype. In Appendix A.9, we present a breakdown of the metrics and datasets used.

5.2. Response Time and Scalability

In real-time applications, where fast data processing is needed to meet user needs, response time is another important performance indicator that can be used [46]. We tested our prototype’s response time under three conditions: low (50 queries per second), medium (150 queries per second) and high (300 queries per second). These simulations are small, but they tell us a lot about how the platform can be used on a larger scale (Table 2). The response time simulations shown in Table 2 used synthetic datasets, but we also ran some initial tests with live data from the Twitter API. These tests were able to give a better view of how the platform handles real-time inputs. While we did not record detailed metrics during these tests, the results seemed consistent with the synthetic data. In our future work, we plan to run more structured tests in real-world scenarios in order to confirm scalability and reliability.

Table 2. Average response time by query volume.

Query Volume	Low Load (50 Q/s)	Medium Load (150 Q/s)	High Load (300 Q/s)
Response Time	112 ms	130 ms	155 ms

The prototype maintained response times below 200 milliseconds, even as the load increased. This aligns with real-time data processing standards and validates the platform’s ability to handle peak loads without compromising responsiveness. Caching frequently accessed queries further improved response times, which ensured minimal delays for users during higher-demand periods. More complex operational demands call for further validation in live environments in order to evaluate scalability.

5.3. User Satisfaction Evaluation

To measure user satisfaction, we conducted a survey with 50 participants (novice, intermediate and expert users). Based on a 5-point Likert scale, the participants rated the platform on several criteria: accuracy, speed, relevance and ease of use (Table 3).

Table 3. User satisfaction scores with different expertise levels, on a 5-point Likert scale.

Attribute	Novice Users	Intermediate Users	Expert Users	Overall Average
Accuracy	4.4	4.6	4.7	4.6
Speed	4.3	4.5	4.6	4.5
Relevance	4.2	4.4	4.5	4.4
Ease of Use	4.2	4.3	4.4	4.3
Overall Score	4.3	4.5	4.6	4.5

User feedback indicates a generally positive satisfaction across all criteria, with expert users rating accuracy and relevance particularly favorably. This is because of the platform's user-centered design which contributed to the usability scores and shows its adaptability to users with different levels of expertise.

5.4. Scalability Analysis and Caching Impact

With the purpose of optimizing for heavy-load scenarios, the platform integrates caching mechanisms that store results for frequently queried topics and reduce the need for repeated data processing. This is a caching strategy that improves response times during peak usage periods [10].

Cache hit rates and the resulting reduction in processing time were used to assess the efficiency of caching. Table 4 shows the percentage reduction in processing time that can be achieved by caching under medium- and high-load conditions.

Table 4. Caching efficiency and processing time reduction under medium- and high-load conditions.

Query Volume	Cache Hit Rate	Processing Time Reduction
Medium Load	35%	20%
High Load	45%	27%

For processing times at high loads, the caching system achieved a 27% reduction, which shows its significant impact on improving platform efficiency. This is a big improvement that shows how important caching is to keep things running smoothly during times of high demand, especially for fast-running apps like real-time ones.

6. Qualitative Performance Analysis

We assessed the platform's qualitative attributes based on the ISO 9126 Software quality model [14], a widely recognized classic standard for software quality evaluation. Functionality, reliability, usability, efficiency, maintainability and portability are the six attributes used by ISO 9126 to categorize software quality. Every attribute provides a distinctive perspective on the way the platform aligns with user needs and maintains operational stability. This analysis complements the quantitative metrics presented previously in Section 5.

6.1. Functionality

Functionality assesses the platform's ability to meet specified requirements and includes characteristics such as suitability, accuracy and security.

The platform's modular design makes it functional and adaptable to a variety of event categories (for example, crisis monitoring and sports updates) and supports a broad range

of real-time extraction needs. The system maintains high contextual relevance and accuracy across multiple domains due to integrating TF-IDF, LSI and NER [16]. The preliminary tests indicate that the prototype is effective in retrieving domain-specific information, and the user satisfaction scores suggest functional suitability.

The platform's accuracy in retrieving relevant data was measured with F1-scores above 0.79 across diverse categories (see Section 5), which reflect a balanced performance in precision and recall. The hybrid ranking model factors in content relevance, source authority, and user behavior. This improves the accuracy of the results and makes the relevance scoring more precise based on context [6].

The platform includes important security protocols like user authentication and data encryption to ensure safe data handling due to the sensitivity of real-time data. Even though the platform is still in the prototyping phases, these safety measures are planned with benefits like public safety and financial monitoring [14].

6.2. Reliability

Reliability assesses, under operational conditions, platform's stability and performance. Features such as maturity, recoverability and fault tolerance are part of it.

The platform's core components are structured for continuous operation, and the initial simulations suggest that the platform can maintain a high uptime. Maturity has an important role for real-time applications and the architecture's robustness is designed to support uninterrupted service where needed [46].

The platform includes redundancy in data storage and implements an error-handling system to enhance reliability under high query volumes. The system can switch to secondary sources and allows data flow to continue without significant interruptions in case a primary data source fails [10].

Automated backup and recovery protocols make sure that data loss is minimized during unexpected failures. However, at an early stage, this design aligns with best practices in real-time applications so that users experience minimal disruptions [14].

6.3. Usability

For a wide range of users, from novices to experts, usability measures just how accessible and user-friendly the platform is. It includes learnability, operability and user error handling.

The platform's intuitive interface allows new users to learn basic functions rather quickly. Novice users could complete core tasks during preliminary tests, such as query filtering, within minutes. The user-centered design supports this ease of learning which includes clear navigation and help options [47].

In order to incorporate a customizable filtering system and color-coded indicators for document relevance, the user interface (UI) is being built and optimized for clarity and usability on an ongoing basis. Our further intention is to build in-platform resources like tooltips, user manuals and an interactive help center to assist users in solving issues on their own in future upgraded versions.

6.4. Efficiency

Efficiency in the ISO 9126 model addresses response time and resource utilization.

We presented in Section 5 that the platform's average response time remained below 200 milliseconds under low, medium and high loads. This was based on optimized indexing, caching mechanisms and efficient data processing strategies [46].

A dynamic distribution of processing tasks that optimizes CPU and memory usage is made possible by the modular design, which also enables horizontal scaling. During load simulations, the prototype's performance remained stable, and it did not consume excessive resources.

6.5. Maintainability

This refers to the platform's ability to evolve over time to meet new demands and technological advancements. It includes modularity, reusability and analyzability.

The architecture is designed in a modular manner, where individual components can be updated without affecting the entire system. For example, enhancements to the NER model can be made without the need to disrupt the data retrieval or query processing functions [10].

The code is structured to enable reusability, thus facilitating adjustments for different analytical needs. In future variants, the entity recognition part could be adapted for tasks such as sentiment analysis or trend detection.

For quick identification and resolution of issues, real-time diagnostic tools enable continuous monitoring of performance metrics. This helps minimize downtime during updates for consistent performance in real-time applications [14].

6.6. Portability

Portability addresses the platform's compatibility with different operational contexts, and it includes, for example, installability, configuration management, and interoperability.

The prototype is being further developed and is compatible with both on-premises and cloud deployment options, to give users flexibility in infrastructure choices; this compatibility is important for organizations with varied technical environments.

Users have the ability to adjust settings (for example, cache limits and query parameters), in order to optimize performance based on particular application needs. Due to this, the platform can adjust to different levels of demand and the preferences of individual users.

The design is already compatible with third-party databases and is being enhanced to incorporate APIs and analytical tools. This interoperability serves as essential for interdisciplinary applications that need data sharing across systems [4,48,49].

7. Limitations and Future Work

Designed as an internal prototype, the platform aims to showcase and investigate possible capabilities in a controlled-access environment for real-time event extraction. That was carried out using a mix of real and synthetic datasets following the initial testing, showing us that the model is robust and scalable, which is encouraging. There are some limitations, and addressing these challenges will serve as a roadmap for the development of the platform in subsequent iterations.

According to our findings, scalability under extreme load conditions is one of the challenges that we have identified. The platform effectively manages typical real-time usage, maintaining response times under 200 milliseconds, while sudden surges that exceed 1000 queries per second could potentially expose bottlenecks. Even though caching and distributed indexing techniques enhance performance, there is a need for further optimization to support unexpected demand peaks, for example, during global or high-stakes events.

Although proficient in standard terminology, another limitation involves the adaptability of the platform's NER model, which could struggle with rapidly evolving event-specific jargon. This could be relevant in dynamic fields like healthcare or technology, where new terms and phrases emerge quickly. The model is pre-trained on a static dataset, and it may overlook or misinterpret unfamiliar terminology that could impact extraction accuracy. The platform is to be tested for scientific purposes because the design of the platform suggests potential applications in this domain. For example, it could process repositories of domain-specific articles to cluster related works, and it could identify trends or extract key findings. As seen in real-time news aggregation contexts, the scientific literature often features stable jargon that could possibly reduce challenges posed by the evolving terminology. These are some of the potential applications that remain aspirational and will be subject to future developments and tests.

Our current prototype has demonstrated robust performance in controlled environments, but future iterations will prioritize deployment in natural, uncontrolled real-world settings. The platform will be tested during live social media events or disaster response scenarios, and this will give specific insights into its adaptability and precision. These scenarios will also help to refine error-handling protocols and responsiveness to the dynamic inputs. We will also run detailed tests during real-world events to measure latency and throughput and ensure the platform handles unexpected demand spikes.

Another limitation in terms of capturing nuanced contextual relationships could be the platform's reliance on traditional relevance scoring methods, such as TF-IDF and LSI. These techniques may not be able to completely interpret deeper contextual meanings often expressed in complex or indirect language, even if they have strong lexical and semantic relevance. Transformer-based models, such as BERT, which excel in contextual understanding, especially in situations where implicit references have a significant influence, seem to be some interesting directions for future research. To overcome the challenges with TF-IDF and LSI of not fully capturing deeper contextual relationships, the integration of transformer-based models like BERT will enable the platform to detect subtle thematic connections (Appendix A.6).

Credibility assessment is another challenge because the authority ranking of the platform relies on structured data signals such as hyperlinks and citation counts. This reliance could affect accuracy when assessing semi-structured sources like social media, where credibility indicators are less standardized but still important for real-time event detection. Including signals from less structured data into the authority ranking system will help to increase accuracy in high-impact fields like crisis response.

There are several enhancement strategies we intend to use to overcome these limitations. Using cutting-edge distributed architectures including microservices and serverless computing will help scalability to be strengthened and enable modular scaling and dynamic resource allocation. This approach should improve the resilience of the platform and preserve real-time performance even with strong demand spikes. In order to address potential bottlenecks during extreme load conditions (for example, exceeding 1000 queries per second), in future work, we plan to explore serverless computing architectures. Simulations based on Kubernetes for dynamic resource allocation hint at some significant latency reductions during peak loads (see Appendix A.4 for configuration aspects). To address the computational costs of transformer models, we will test for future iterations lightweight models like DistilBERT. These models use fewer resources while also maintaining good accuracy for contextual analysis. For multi-source fusion, we plan to use methods such as Kubernetes-based autoscaling (described in Appendix A.4) to make the infrastructure scalable under different workloads. This interesting approach can help the platform combine structured and semi-structured data while also handling scalability challenges. By fine-tuning the NER model with domain-specific data and integrating contextual embeddings from transformer models like BERT, the platform will better handle changing terminology and specialized entities in dynamic fields such as "mRNA technology" or "edge AI" (details on the implementation of these transformer-based embeddings are in Appendix A.6).

We plan to add transformer-based models to provide deeper contextual insights and enhance the platform's capacity to rank papers with implicit or complex phrasing more precisely, thus refining relevance scoring. Including social media and sentiment analysis in the authority ranking system will also help the platform to be flexible in assessing credibility over a wider spectrum of data sources. This increased capacity could be especially helpful in high-impact events when unverified, but timely social media data can supplement official channels.

For future work, we will concentrate on using multi-source data fusion, which synthesizes information from both structured sources (such as news items) and semi-structured or unstructured sources (such as social media), therefore supporting thorough event coverage. This will reduce reliance on any single data type, enhance resilience to gaps in particular sources, and offer a more complete view of real-time events.

For long-term plans, we want to include the integration of Explainable AI (XAI) to foster greater transparency, particularly through features such as attention visualization in transformer models. This could help build user trust in high-stakes applications like policy monitoring and crisis response, by explaining the basis for document relevance and credibility. For example, in dynamic fields like crisis response or misinformation detection, attention visualizations could give end-users an intuitive understanding of how key phrases or concepts influenced the platform's ranking decisions. During high-pressure scenarios, such visualizations could help crisis response teams quickly assess why certain documents or data points were prioritized and aid in faster decision-making.

As we further enhance the future versions, we plan to make portions of this prototype's codebase available for broader research collaboration. Adaptive learning from user interactions, multi-platform data aggregation, ethical AI practices for bias mitigation and other advanced AI techniques are some of the things that we plan to investigate in future versions of the platform.

8. Conclusions

In this paper, we presented a scalable prototype for real-time event extraction, which was designed to meet the challenges of high-frequency data monitoring. The platform retrieves accurate and contextually relevant information across vast datasets in real time by integrating advanced data processing, NLP and relevance-ranking methods. It has a modular design and adaptable framework which makes it suitable for a diverse range of applications, from crisis response and news aggregation to sports and financial events monitoring.

The core architecture is organized into three modules (data retrieval, document processing and query processing), each of which is optimized for efficient scaling. The hybrid ranking model combines multiple relevance criteria, such as content alignment, source credibility, and user interaction, thus dynamically prioritizing information based on user needs and evolving event contexts. Testing with real and synthetic datasets validated the prototype's robustness and showed its accuracy and reliability under high-demand conditions in a controlled-access environment.

In this paper, we introduced several contributions to real-time event extraction, from innovative hybrid ranking and relevance models to detailed validations of quantitative and qualitative performance. Despite being a prototype, it has demonstrated some promising initial results in critical areas, like addressing core demands for handling fast-evolving data. Some of the valuable contributions for researchers and practitioners in the fields of data-driven event extraction and monitoring are the platform's adaptability and performance validation.

Looking at the future, the next iterations of the platform aim to further enhance its capabilities. The platform has a lot of potential to become an important resource for companies needing accurate, timely information in dynamic data environments by addressing both immediate challenges and long-term goals.

Author Contributions: Conceptualization, M.-C.A.; methodology, M.-C.A., V.P.B. and C.-A.C.; software, M.-C.A., V.P.B. and C.-A.C.; validation, M.-C.A., V.P.B. and C.-A.C.; formal analysis, M.-C.A. and V.P.B.; investigation, M.-C.A.; resources, M.-C.A. and N.F.; data curation, M.-C.A.; writing—original draft preparation, M.-C.A., V.P.B., S.-C.P., N.F. and C.-A.C.; writing—review and editing, M.-C.A., V.P.B., S.-C.P. and C.-A.C.; visualization, M.-C.A., V.P.B. and S.-C.P.; supervision, M.-C.A.; project administration, M.-C.A. and V.P.B.; funding acquisition, M.-C.A., V.P.B., S.-C.P., N.F. and C.-A.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Technical Specifications and Workflow

Appendix A.1. Hardware and Software Requirements

Hardware

- CPU: Multi-core processor, 2.4 GHz or higher (desktop-level CPU should be sufficient for testing purposes);
- Memory: 8–16 GB RAM (sufficient for prototyping, additional memory may be required for larger-scale tests);
- Storage: SSD with at least 500 GB capacity (scalable based on data volume for caching and processing needs);
- Network: Standard internet connection (1 Gbps or higher for cloud setups).

Software

- Operating system: Linux (Ubuntu 20.04 LTS or higher recommended for ease of development and compatibility);
- Programming languages: Python 3.8 or higher;
- Libraries
 - NLP and Machine Learning: NLTK, scikit-learn and spaCy;
 - Data Processing and Storage: Pandas and MongoDB (for flexible data storage during development);
 - Web Scraping: BeautifulSoup and Scrapy;
- Deployment tools
 - Docker for containerized deployment (suitable for modular testing of individual components);
 - Kubernetes (optional) for orchestrating microservices in larger test environments, though not necessary for the initial prototyping stages;
- Caching: Redis for caching high-demand queries, mainly useful in distributed or higher-load tests, but optional at this stage.

System architecture

- Modular deployment—Each component (data retrieval, document processing, query processing) is designed to run independently within Docker containers; this allows for isolated testing and modular scaling.
- Prototype-ready cloud compatibility—While the prototype can be deployed on cloud platforms (e.g., AWS, Google Cloud), the architecture remains flexible for local or cloud testing. Full serverless deployment (like AWS Lambda) and advanced autoscaling are optional and would be explored if the prototype advances to higher loads.

Appendix A.2. Configuration and Implementation Choices

Containerization and orchestration

The platform components are each containerized to enable isolated updates and dynamic scaling.

Docker-compose.yml file to configure the data retrieval and document processing services within a shared network:

```
version: '3.7'
services:
  data_retrieval:
    image: platform_data_retrieval:latest
    environment:
      - RETRY_INTERVAL=10 # Retry interval setting for handling source retries
      - LOG_LEVEL=info    # Set log level to track retrieval events
    networks:
      - backend_network # Connect service to the shared backend network
```



```

document_processing:
  image: platform_document_processing:latest
  environment:
    - TF_IDF_THRESHOLD=0.5 # Minimum relevance threshold for document filtering
    - NER_MODEL_PATH=/models/ner_model # Path to pre-trained NER model
  networks:
    - backend_network # Connect service to the shared backend network
networks:
  backend_network:
    driver: bridge # Use bridge network to link services in the same network

```

Database indexing and storage

Simple indexing in MongoDB to support efficient queries; sharding is omitted for the prototype stage:

```

from pymongo import MongoClient
client = MongoClient('mongodb://localhost:27017')
db = client.event_database
# create compound index to support fast searches on entity_name and event_date
db.events.create_index([("entity_name", pymongo.ASCENDING), ("event_date",
pymongo.DESENDING)])

```

Redis caching

Basic Redis setup for caching query results:

```

import redis
cache = redis.StrictRedis(host='localhost', port=6379, db=0)
# Cache query results with expiration to limit memory usage
def cache_query_result(query_key, result, expiration=300):
    cache.setex(query_key, expiration, result) # Store result for 5 minutes
# Retrieve cached result if it exists to speed up repeat queries
def get_cached_query_result(query_key):
    return cache.get(query_key)

```

Prototype data flow example

Data retrieval and processing pipeline for prototyping:

```

# Prototype data flow for event extraction
def event_extraction_pipeline(query):
    query_key = f"query_cache:{query}"
    # Check cache for existing results
    cached_results = get_cached_query_result(query_key)
    if cached_results is not None:
        return cached_results # Use cached results if available
    # Retrieve raw data from sources
    raw_data = data_retrieval_pipeline()
    # Process data for relevance and structure
    processed_documents = document_processing_pipeline(raw_data)
    # Filter results matching the query
    relevant_results = [
        doc for doc in processed_documents
        if query in doc.get('content', '') or any(query in entity for entity in doc.get('entities', []))
    ]
    # Cache the results for future use
    cache_query_result(query_key, relevant_results, expiration=300)
    return relevant_results # Return relevant documents

```

Appendix A.3. Pseudocode for Core Components

1. The data retrieval pipeline

```

# Retrieve and process data from sources
def data_retrieval_pipeline():
    sources = get_sources() # List of sources
    crawled_data = []
    for source in sources:
        schedule_interval = adjust_schedule(source) # Set interval
        raw_data = crawl(source, schedule_interval) # Fetch data
        if raw_data is None or raw_data.is_empty():
            log_error(source) # Log issue
            continue
        processed_data = preprocess(raw_data) # Clean data
        crawled_data.append(processed_data)
    return crawled_data

```

2. The document processing pipeline with pre-trained models for LSI and NER

```

# Process documents using pre-trained models
def document_processing_pipeline(crawled_data):
    relevant_docs = [] # Store processed documents
    for document in crawled_data:
        # Calculate TF-IDF score for filtering
        tfidf_score = compute_tfidf(document, global_tfidf_matrix)
        if tfidf_score > RELEVANCE_THRESHOLD:
            # Apply LSI for semantic clustering
            lsi_representation = apply_lsi(global_lsi_model, document)
            # Perform NER to extract key entities
            entities = apply_ner(document)
            # Structure the document for indexing
            structured_doc = {
                'content': lsi_representation,
                'entities': entities
            }
            relevant_docs.append(structured_doc)
    return relevant_docs

```

3. Query processing pipeline

```

# Process user queries for document retrieval
def query_processing_pipeline(user_query):
    # Search index for matching documents
    candidate_docs = search_inverted_index(user_query)
    # Return cached results if available
    if is_cached(user_query):
        return retrieve_cache(user_query)
    # Rank documents by relevance
    ranked_results = rank_documents(candidate_docs, user_query.preferences)
    # Cache results for future queries
    cache_results(user_query, ranked_results)
    # Return ranked results
    return ranked_results

```

Appendix A.4. Distributed Architectures for Scalability

The yaml configuration demonstrates Kubernetes-based autoscaling to handle high query loads dynamically. Some early tests suggest that Kubernetes-based autoscaling could possibly reduce average latency by approx. 30% during peak loads and enable the platform to handle up to 1500 queries per second (while maintaining response times below 200 ms).

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: query-processor
spec:
  replicas: 1
  selector:
    matchLabels:
      app: query-processor
  template:
    metadata:
      labels:
        app: query-processor
    spec:
      containers:
      - name: query-processor
        image: query-processor:latest
        resources:
          limits:
            cpu: "2"
            memory: "1Gi"
          requests:
            cpu: "1"
            memory: "512Mi"
---
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: query-processor-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: query-processor
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 70

```

Appendix A.5. Sentiment Analysis for Credibility Signals

This shows how sentiment analysis based on transformer models can complement authority-based ranking methods by assessing the credibility of semi-structured data sources (like social media). This can identify reliable information and filter out potentially misleading sources in dynamic environments, such as during crisis situations.

```

from transformers import pipeline
# Load a pre-trained sentiment-analysis model
sentiment_pipeline = pipeline("sentiment-analysis")
def assess_sentiment(content):
    """
    Analyze sentiment, so as to assess credibility and reliability of input text
    Args:
        content (str): Text from the social media or from some other semi-structured sources
    """

```

```

Returns:
    Dict: Sentiment label and confidence score
    """
    sentiment_result = sentiment_pipeline(content)
    return {
        "label": sentiment_result[0]['label'], # Positive, Neutral, or Negative sentiment
        "score": sentiment_result[0]['score'] # Confidence of the sentiment prediction
    }
# example -> assess the sentiment of a social media post
document_content = "this news article about the crisis is spreading panic among readers"
sentiment = assess_sentiment(document_content)
print(f"Sentiment: {sentiment['label']} (Confidence: {sentiment['score']})")

```

Appendix A.6. Attention Visualization for Explainability

```

import matplotlib.pyplot as plt
import torch
from transformers import BertTokenizer, BertModel
# this is loading the BERT model, and the tokenizer
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
model = BertModel.from_pretrained("bert-base-uncased", output_attentions=True)
def visualize_attention(sentence):
    """
    visualizes the attention weights for the input text
    Args:
        sentence (str): Text input.
    """
    inputs = tokenizer(sentence, return_tensors="pt")
    outputs = model(**inputs)
    # extract the attention weights (last layer)
    attentions = outputs.attentions[-1][0]
    mean_attention = torch.mean(attentions, dim=0).detach().numpy()
    # plot attention matrix
    plt.matshow(mean_attention, cmap="viridis")
    plt.title("Attention Visualization")
    plt.colorbar()
    plt.show()
# an example
example_sentence = "The company announced a groundbreaking mRNA-based treatment."
visualize_attention(example_sentence)

```

Appendix A.7. Hyperparameter Tuning

These next hyperparameters were utilized for the prototyping phase of the platform. These values were selected based on iterative small-scale testing, which balances computational efficiency and functional performance.

Table A1. Hyperparameters used for the prototype configuration and their values.

Component	Hyperparameter	Value
NER	Learning rate	0.001
NER	Batch size	32
NER	Epochs	100
LSI	Retained singular values	100
TF-IDF	Relevance threshold	0.5

Values like learning rate (dimensionless), batch size (number of samples) and relevance threshold (range 0–1) are specified without explicit units because they relate to algorithm-specific settings.

Appendix A.8. Ablation Study

We conducted an ablation study to evaluate the contribution of each component of the platform (TF-IDF, LSI, NER) to its overall performance. For this, we individually disabled each component and then measured the impact on precision, recall and F1-score. The results are shown in the table below.

Table A2. Impact of removing each component on performance metrics.

Component Removed	Precision Drop (%)	Recall Drop (%)	F1-Score Drop (%)
NER	15	10	12
LSI	10	8	15
TF-IDF	25	18	25

The removal of TF-IDF determined a most significant drop in F1-score (25%), which shows its role in filtering irrelevant data. The removal of LSI reduced semantic clustering accuracy, and it resulted in a 15% drop in F1-score. The removal of NER reduced precision by 15%, which underlines its role in identifying relevant entities.

Appendix A.9. Performance Comparison with Existing Systems

Results summarize precision, recall and F1-score for our prototype, Deep-Eware and TwitterNews+ based on the datasets used. The metrics for Deep-Eware are sourced from its original study; the F1-score for TwitterNews+ is recalculated for consistency based on the Formula (7) used in our methodology.

Table A3. The comparison of performance metrics across event detection systems.

Platform	Precision	Recall	F1-Score	Dataset	Comments
Our prototype	0.84	0.82	0.83	Kaggle, GDELT, synthetic data	balanced performance, some slight challenges with overlapping categories
Deep-Eware [34]	0.99 (incidents)	0.95 (incidents)	0.97 (incidents)	8137 tweets (annotated)	strong results for structured topics, but less effective for mixed clusters,
	1.00 (promotions)	0.26 (promotions)	0.41 (promotions)		challenges in mixed/ambiguous categories
TwitterNews+ [33]	0.89	0.96	0.92	Events2012	robust for structured data, effective noise handling

Appendix A.10. Example of Retry, Fallback and Authenticity Checks

The prototype handles data retrieval failures with retries of the source and switches to backups when needed, and also verifies key metrics like CTR and engagement in order to ensure data authenticity:

```
def retrieve_data_with_fallback(source_id):
    for attempt in range(MAX_RETRIES):
        try:
            data = fetch_from_source(source_id)
            if data and validate_data_authenticity(data):
                return data
        except Exception as e:
            log_error(f'Attempt {attempt + 1} failed for source {source_id}: {e}')
    log_error(f'Source {source_id} failed after {MAX_RETRIES} retries.')
    return fetch_from_cache_or_secondary(source_id)

def validate_data_authenticity(data):
    if data.get('clicks', 0) > MAX_CLICKS_THRESHOLD:
        return False
    if data.get('unique_users', 0) / data.get('total_users', 1) < MIN_UNIQUE_RATIO:
        return False
    return True
```

References

1. Aggarwal, C.C.; Zhai, C.X. *Mining Text Data*; Springer: New York, NY, USA, 2012; ISBN 978-1-4614-3222-7.
2. Adzic, G.; Chatley, R. Serverless Computing: Economic and Architectural Impact. In Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Paderborn, Germany, 4–8 September 2017; Volume 2017.
3. Bartelheimer, C.; zur Heiden, P.; Lüttenberg, H.; Beverungen, D. Systematizing the Lexicon of Platforms in Information Systems: A Data-Driven Study. *Electron. Mark.* **2022**, *32*, 375–396. [[CrossRef](#)]
4. Al-Ruithe, M.; Benkhelifa, E.; Hameed, K. A Systematic Literature Review of Data Governance and Cloud Data Governance. *Pers. Ubiquitous Comput.* **2019**, *23*, 839–859. [[CrossRef](#)]
5. Borowiec, M.; Piszko, R.; Rak, T. Knowledge Extraction and Discovery about Web System Based on the Benchmark Application of Online Stock Trading System. *Sensors* **2023**, *23*, 2274. [[CrossRef](#)] [[PubMed](#)]
6. Joachims, T.; Granka, L.; Pan, B.; Hembrooke, H.; Radlinski, F.; Gay, G. Evaluating the Accuracy of Implicit Feedback from Clicks and Query Reformulations in Web Search. *ACM Trans. Inf. Syst.* **2007**, *25*, 2. [[CrossRef](#)]
7. Rajpurkar, P.; Jia, R.; Liang, P. Know What You Don't Know: Unanswerable Questions for SQuAD. In Proceedings of the ACL 2018—56th Annual Meeting of the Association for Computational Linguistics, (Long Papers), Melbourne, Australia, 15–20 July 2018; Volume 2.
8. McMahan, H.B.; Holt, G.; Sculley, D.; Young, M.; Ebner, D.; Grady, J.; Nie, L.; Phillips, T.; Davydov, E.; Golovin, D.; et al. Ad Click Prediction: A View from the Trenches. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, IL, USA, 11–14 August 2013. Volume Part F128815. [[CrossRef](#)]
9. Liang, Z.; Guo, J.; Qiu, W.; Huang, Z.; Li, S. When Graph Convolution Meets Double Attention: Online Privacy Disclosure Detection with Multi-Label Text Classification. *Data Min. Knowl. Discov.* **2024**, *38*, 1171–1192. [[CrossRef](#)]
10. Nasraoui, O.; Spiliopoulou, M.; Srivastava, J.; Mobasher, B.; Masand, B. WebKDD 2006: Web Mining and Web Usage Analysis Post-Workshop Report. *ACM SIGKDD Explor.* **2006**, *8*, 84–89. [[CrossRef](#)]
11. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.; Le, Q.V. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Volume 33.
12. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the NAACL HLT 2019—2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies—Proceedings of the Conference, Minneapolis, MN, USA, 2–7 June 2019; Volume 1.
13. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 2017.
14. ISO/IEC 9126; Software Engineering—Product Quality. Part 1: Quality Model. International Organization for Standardization: Geneva, Switzerland, 2001.
15. Salton, G.; Buckley, C. Term-Weighting Approaches in Automatic Text Retrieval. *Inf. Process Manag.* **1988**, *24*, 513–523. [[CrossRef](#)]
16. Manning, C.D.; Raghavan, P.; Schütze, H. *Introduction to Information Retrieval*; Cambridge University Press: Cambridge, UK, 2008.
17. Baeza-Yates, R.; Ribeiro-Neto, B. *Modern Information Retrieval*, 1st ed.; Addison Wesley: Boston, MA, USA, 1999; ISBN 978-0201398298.
18. Deerwester, S.; Dumais, S.T.; Furnas, G.W.; Landauer, T.K.; Harshman, R. Indexing by Latent Semantic Analysis. *J. Am. Soc. Inf. Sci.* **1990**, *41*, 391–407. [[CrossRef](#)]
19. Dumais, S.T. Latent Semantic Analysis. *Annu. Rev. Inf. Sci. Technol.* **2004**, *38*, 188–230. [[CrossRef](#)]
20. Nadeau, D.; Sekine, S. A Survey of Named Entity Recognition and Classification. *Linguisticae Investig.* **2007**, *30*, 3–26. [[CrossRef](#)]
21. Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; Dyer, C. Neural Architectures for Named Entity Recognition. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016—Proceedings of the Conference, San Diego, CA, USA, 12–17 June 2016.
22. Li, X.; Feng, J.; Meng, Y.; Han, Q.; Wu, F.; Li, J. A Unified MRC Framework for Named Entity Recognition. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020.
23. Yadav, V.; Bethard, S. A Survey on Recent Advances in Named Entity Recognition from Deep Learning Models. In Proceedings of the COLING 2018—27th International Conference on Computational Linguistics, Santa Fe, NM, USA, 20–26 August 2018.
24. Akbik, A.; Bergmann, T.; Vollgraf, R. Pooled Contextualized Embeddings for Named Entity Recognition. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–7 June 2019; Association for Computational Linguistics (ACL): Minneapolis, MN, USA, 2019; Volume 1, pp. 724–728.
25. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. In Proceedings of the 1st International Conference on Learning Representations, ICLR 2013—Workshop Track Proceedings, Scottsdale, Arizona, 2–4 May 2013.
26. Pennington, J.; Socher, R.; Manning, C.D. GloVe: Global Vectors for Word Representation. In Proceedings of the EMNLP 2014—2014 Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014.
27. Ethayarajh, K. How Contextual Are Contextualized Word Representations? Comparing the Geometry of BERT, ELMO, and GPT-2 Embeddings. In Proceedings of the EMNLP-IJCNLP 2019—2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Hong Kong, China, 3–7 November 2019.

28. Bommasani, R.; Hudson, D.A.; Adeli, E.; Altman, R.; Arora, S.; von Arx, S.; Bernstein, M.S.; Bohg, J.; Bosselut, A.; Brunskill, E.; et al. On the Opportunities and Risks of Foundation Models. *arXiv* **2022**, arXiv:2108.07258v.
29. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, arXiv:1907.11692.
30. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. *arXiv* **2019**, arXiv:1910.0110.
31. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* **2020**, *21*, 1–67.
32. Algiriyage, N.; Prasanna, R.; Stock, K.; Doyle, E.E.H.; Johnston, D. DEES: A Real-Time System for Event Extraction from Disaster-Related Web Text. *Soc. Netw. Anal. Min.* **2023**, *13*, 6. [\[CrossRef\]](#)
33. Hasan, M.; Orgun, M.A.; Schwitter, R. Real-Time Event Detection from the Twitter Data Stream Using the TwitterNews+ Framework. *Inf. Process Manag.* **2019**, *56*, 1146–1165. [\[CrossRef\]](#)
34. Afyouni, I.; Khan, A.; Aghbari, Z. Al Deep-Eware: Spatio-Temporal Social Event Detection Using a Hybrid Learning Model. *J. Big Data* **2022**, *9*, 86. [\[CrossRef\]](#)
35. George, Y.; Karunasekera, S.; Harwood, A.; Lim, K.H. Real-Time Spatio-Temporal Event Detection on Geotagged Social Media. *J. Big Data* **2021**, *8*, 91. [\[CrossRef\]](#)
36. Li, Q.; Li, J.; Sheng, J.; Cui, S.; Wu, J.; Hei, Y.; Peng, H.; Guo, S.; Wang, L.; Beheshti, A.; et al. A Survey on Deep Learning Event Extraction: Approaches and Applications. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 6301–6321. [\[CrossRef\]](#) [\[PubMed\]](#)
37. Xiang, W.; Wang, B. A Survey of Event Extraction from Text. *IEEE Access* **2019**, *7*, 173111–173137. [\[CrossRef\]](#)
38. Pais, S.; Cordeiro, J.; Jamil, M.L. NLP-Based Platform as a Service: A Brief Review. *J. Big Data* **2022**, *9*, 54. [\[CrossRef\]](#)
39. Barker, J.L.P.; Macleod, C.J.A. Development of a National-Scale Real-Time Twitter Data Mining Pipeline for Social Geodata on the Potential Impacts of Flooding on Communities. *Environ. Model. Softw.* **2019**, *115*, 213–227. [\[CrossRef\]](#)
40. Klein, G.; Kim, Y.; Deng, Y.; Senellart, J.; Rush, A.M. OpenNMT: Open-Source Toolkit for Neural Machine Translation. In Proceedings of the ACL 2017—55th Annual Meeting of the Association for Computational Linguistics, Proceedings of System Demonstrations, Vancouver, BC, Canada, 30 July–4 August 2017.
41. Kaggle Multilingual Disaster Response Messages Dataset. Available online: <https://www.kaggle.com/datasets/landlord/multilingual-disaster-response-messages> (accessed on 6 September 2024).
42. The GDELT Project GDELT Event Database. Available online: <http://data.gdeltproject.org/events/index.html> (accessed on 6 September 2024).
43. X.com Twitter (X) API. X Developer Platform. Available online: <https://developer.x.com/en/docs/x-api> (accessed on 4 September 2024).
44. Baumgartner, J.M.; Lazzarin, E.; Seiler, A. Pushshift Reddit API. Available online: <https://github.com/pushshift/api> (accessed on 5 October 2024).
45. Joachims, T. Optimizing Search Engines Using Clickthrough Data. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, AB, Canada, 23–26 July 2002.
46. Dean, J.; Barroso, L.A. The Tail at Scale. *Commun. ACM* **2013**, *56*, 74–80. [\[CrossRef\]](#)
47. Markov, Z.; Larose, D.T. *Data Mining the Web: Uncovering Patterns in Web Content, Structure, and Usage*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2007; ISBN 978-0-471-66655-4.
48. Höfer, C.N.; Karagiannis, G. Cloud Computing Services: Taxonomy and Comparison. *J. Internet Serv. Appl.* **2011**, *2*, 81–94. [\[CrossRef\]](#)
49. Kaur, K.; Sharma, S.; Kahlon, K.S. Interoperability and Portability Approaches in Inter-Connected Clouds: A Review. *ACM Comput. Surv.* **2017**, *50*, 1–40. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.