Yet Another Earthquake Project

"BVA 504E - Big Data Technologies and Applications" Final Project Report

Burak Kılıç
Istanbul Technical University
Mechanical Engineering
Project Coordinator

Mehmet Akif Tür
Istanbul Technical University
Big Data and Business Analytics
Data Engineer

Şevval Aysal
Istanbul Technical University
Big Data and Business Analytics
Data Analyst

Esra Kayaalp
Istanbul Technical University
Big Data and Business Analytics
Data Scientist

Bilge Alper
Istanbul Technical University
Computational Science and Engineering
Data Engineer

***Abstract*— This paper presents a comprehensive study on the application of big data methods for earthquake visualization and correlation analysis with electric field data. It highlights the importance of utilizing large-scale datasets and advanced analytics techniques to gain valuable insights into earthquake patterns and potential relationships with other variables. The paper discusses the challenges and methodologies involved in acquiring and integrating earthquake and electric field data, emphasizing data quality and preprocessing. It explores the use of correlation analysis and visualization techniques to uncover meaningful connections between earthquakes and electric field data. The findings demonstrate the potential of big data analytics in enhancing earthquake research and risk assessment, and emphasize the need for further exploration and integration of electric field data in seismic monitoring systems.**

Keywords—Big Data, Earthquake, Electric Field, Streaming

I. Introduction

The field of earthquake research has experienced significant advancements in recent years, driven by the availability of vast amounts of data and the adoption of advanced data analytics technologies. Big data methods, powered by technologies such as Apache NiFi, Apache Kafka, Apache Spark, Elasticsearch, and Kibana, have played a pivotal role in revolutionizing earthquake visualization and the exploration of correlations between earthquakes and electric field data. These technologies enable the efficient collection, processing, analysis, and visualization of large-scale datasets, empowering researchers to gain valuable insights into the complex nature of seismic events and their potential relationships with other variables.

Earthquakes pose substantial risks to human life, infrastructure, and the environment, emphasizing the need for a deep understanding of their patterns, characteristics, and potential precursors. Traditional seismic monitoring systems relied on limited data sources, primarily seismic sensors, for earthquake detection and analysis. However, with the advent of technologies, data collection has become more comprehensive, incorporating diverse datasets from sources such as satellite imagery, geospatial data, social media, and electric field measurements. These technologies facilitate the integration and processing of multi-source data, enabling researchers to obtain a holistic view of seismic activity and investigate correlations between earthquakes and electric field data.

Examples of some big data projects related earthquakes, include ShakeAlert, a U.S. initiative to develop an earthquake early warning system [1]–[3]; QuakeML, an international project to standardize earthquake data exchange; the Earthquake Information Exchange (EIE) project [4], [5], which aims to create a comprehensive earthquake data repository; the Global Earthquake Model (GEM), a global initiative for earthquake risk assessment [6], [7]; and the QuakeSim project which led by the Southern California Earthquake Center (SCEC) that uses big data and advanced simulation models to study earthquake behavior and predict seismic activity [8]. There are many projects can be add to the list such as EARS, REPSEDORS, GeoNet, etc. [9]–[11]. All these projects highlight the potential of big data analytics to revolutionize earthquake research and contribute to more effective earthquake-related decision-making and mitigation strategies.

Through this research, we seek to contribute to the advancement of earthquake research by leveraging the power of big data technologies. By harnessing the capabilities of Apache NiFi, Apache Kafka, Apache Spark, Elasticsearch, and Kibana, researchers can gain deeper insights into earthquake phenomena, leading to enhanced earthquake monitoring, prediction, risk assessment, and disaster mitigation strategies. The integration of these technologies in earthquake research has the potential to revolutionize the field and facilitate more effective decision-making and mitigation strategies in the face of seismic events.

II. Used Data and Ingestion

A. Earthquake Data

We obtained earthquake data using the AFAD Earthquake API, which can be accessed through the following URL: "https://deprem.afad.gov.tr/apiv2/event/filter". The API provides earthquake data with two date parameters, "start" and "end," in the format "yyyy-MM-dd HH:mm:ss". To retrieve the data, we utilized the InvokeHTTP processor in NiFi.

For historical data, we set the start parameter to "2022-01-01 00:00:00" and the end parameter to the current time. We ran the processor once to retrieve the historical earthquake data (Figure 1). For real-time data, we set the start parameter to 30 minutes before the current time, and the end parameter to the current time. Due to the data not always being chronologically ordered, we couldn't use the value from the last data as the start parameter. To continuously fetch real-time data, we configured the processor to run every 30 seconds.

To send the retrieved earthquake data to Kafka, we used the PublishKafkaRecord_2_6 processor. We specified the Kafka Brokers property with the appropriate Kafka port and set the Topic Name to the relevant topic. Since the data was obtained and sent in JSON format, we configured the RecordReader as JsonTreeReader and the RecordWriter as JsonRecordSetWriter.

Additionally, we utilized the PutS3Object processor to store the earthquake data in an S3 bucket. We provided the bucket name and set the region to "US East (N. Virginia)".

To route the earthquake data to both Kafka and S3, we connected the Response of the InvokeHTTP processor to the PublishKafkaRecord and PutS3Object processors.

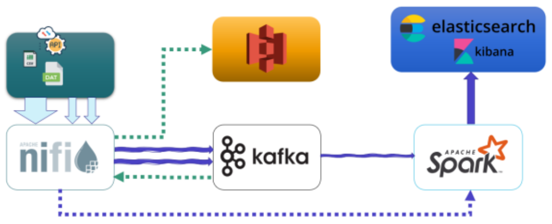


Figure. 1. Project Pipeline

B. *Electric Field Data*

To obtain electric field data, we retrieved it from the "http://deprem.itu.edu.tr" website. Since no API was available for accessing the data, we employed web scraping techniques.

III. Power of NiFi

In addition to utilizing NiFi for data ingestion, we extended its functionality to perform various tasks and executed specific scripts within the NiFi framework. These additional capabilities played a crucial role in enhancing the overall data processing workflow. Several key aspects of these advancements are elaborated below, highlighting their significance in our research endeavor.

A. *Put Electric Field Data to Amazon S3 Bucket*

To store the electric field data in an S3 bucket, we followed the subsequent steps. First, we retrieved the data from the relevant Kafka producer using the ConsumeKafka_2_6 processor. We configured the processor by setting the properties Kafka Brokers, Topic Name, and Group ID to ensure the correct retrieval of data from Kafka.

Next, we utilized the PutS3Object processor to save the retrieved data to an S3 bucket. This processor enables seamless integration with S3 storage. By configuring the appropriate properties, such as bucket name and region, we ensured that the data was correctly uploaded to the desired S3 bucket.

B. *Execute Python Scripts via NiFi*

To execute the Python scripts responsible for running Kafka and Spark within the NiFi environment, we employed the ExecuteStreamCommand processor. This processor allows the execution of external command-line programs or scripts.

In the ExecuteStreamCommand processor configuration, we set the Command Path property to the path of the Python installation on the NiFi instance. Additionally, we specified the Command Arguments property as the file path of the Python script to be executed.

It is important to note that in order to trigger the execution of an ExecuteStreamCommand processor, it is necessary to provide input data. In this case, we connected the ExecuteStreamCommand processor with a GenerateFlowFile processor using a success relationship. This ensured that a flow file was generated as input for the Python script execution.

By employing these techniques, we were able to seamlessly integrate the execution of Python scripts into the NiFi data flow, enabling the efficient processing and manipulation of data within the specified workflow.

IV. Kafka Streams

Real-time data processing has become increasingly important in various domains, as it enables organizations to make timely decisions based on up-to-date information. Kafka, a distributed streaming platform, provides the necessary infrastructure and tools for efficient real-time data processing. This part of the project aims to showcase a practical implementation of real-time data processing using Kafka and highlight the functionalities of different components involved.

A. *Data Retrieval*

The first component of the Kafka implementation focuses on data retrieval. The script initiates the process by downloading electric data from a specific source. To achieve this, it establishes a session, performs user authentication, and utilizes web scraping techniques to extract the data within a specified time range. By employing web scraping, the script ensures the collection of relevant electric data required for further processing.

B. *Data Processing*

Upon successful retrieval of the data, it undergoes several processing steps. Firstly, the data is transformed into a JSON format, where each data point is represented by a timestamp and corresponding value. This transformation facilitates compatibility with Kafka's data format and enables seamless integration. Additionally, the script handles invalid or missing data points appropriately to ensure data integrity. Subsequently, the processed data is divided into bulk messages, adhering to a predefined maximum size constraint. This division optimizes data transmission efficiency and reduces network overhead. Finally, the processed data is sent to a Kafka topic for further analysis and processing.

C. *Kafka Integration*

The project utilizes the Confluent Kafka library to interact with Kafka and leverage its producer functionality. The script establishes a connection with the Kafka cluster and utilizes the producer functionality to send bulk messages to the specified Kafka topic. This integration ensures reliable delivery of messages to the Kafka cluster, thereby facilitating efficient data streaming and processing.

D. *Near-Real-time Updates*

To ensure real-time updates, the script periodically checks for the latest electric data within the specified time range. By retrieving the most recent data, the script enables timely processing and analysis. The processed data is then sent as a single message to the Kafka topic. This process is repeated at regular intervals to maintain an up-to-date stream of data in Kafka, facilitating real-time analysis and decision-making.

V. Streaming with Spark

Streaming data processing has gained significant importance in recent years due to the continuous generation of data from various sources. In this part, we discuss the implementation of a stream processing pipeline that involves reading data from two Kafka topics, namely electricRaw and earthquakeRaw, and continuously writing it into “.parquet” files using Apache Spark's ReadStream() and WriteStream() functionalities.

The first step in the process is to establish connections to the Kafka cluster and configure the necessary parameters for consuming the data streams. This includes specifying the Kafka brokers, the topics from which data needs to be consumed (electricRaw and earthquakeRaw), and other relevant configurations such as the serialization format and consumer groups.

Once the connections are established, the ReadStream() function is used to create streaming DataFrames from the Kafka topics. This function ensures that data is read from Kafka in a continuous and real-time manner, enabling the processing of incoming messages as they arrive. The schemas of the DataFrames are inferred based on the structures of the data in the respective Kafka topics.

Next, we define the desired transformations and operations to be performed on the streaming DataFrames. These transformations can include filtering, aggregation, feature extraction, and any other necessary data manipulations based on the specific requirements of the project. It is important to note that these transformations are applied on the streaming DataFrames, allowing for real-time data processing.

After the required transformations have been applied, the processed data is written to “.parquet” files using the WriteStream() function provided by Apache Spark. This function allows for continuous writing of the streaming DataFrames into “.parquet” files in an append mode. The “.parquet” format is chosen due to its columnar storage and compression capabilities, which optimize data storage and retrieval.

To ensure the continuous execution of the streaming pipeline, an infinite loop is implemented around the streaming process. This loop allows for the continuous reading of data from the electricRaw and earthquakeRaw topics and writing it into “.parquet” files without interruption. By adopting this approach, the pipeline can handle the incoming data streams in a continuous and seamless manner, providing up-to-date and persistent storage of the processed data.

VI. Analyzing Data

The advent of big data has necessitated the development of robust tools and technologies capable of effectively managing and processing massive volumes of information. Apache Spark, a distributed computing framework, offers a comprehensive solution for performing big data analytics, owing to its remarkable scalability, exceptional processing speed, and user-friendly features. Spark's fundamental components, including Resilient Distributed Datasets (RDDs) and Spark SQL, facilitate seamless manipulation and transformation of data. In the present study, the objective was to harness the capabilities of Spark to analyze earthquake and electrical datasets, thereby investigating potential correlations and gaining valuable insights through their combined analysis. By leveraging Spark's functionalities, this research aimed to uncover hidden patterns and relationships within the datasets, providing a deeper understanding of the phenomena under investigation.

In the process of analyzing the datasets, we employed various techniques to extract meaningful features. Specifically, for the earthquake data, we aimed to identify the main fault line by utilizing the available city information. Initially, our intention was to leverage latitude and longitude data to accomplish this task. However, we encountered a challenge as the latitude and longitude information for the fault lines was not readily accessible. As a result, we adapted our approach by modifying the data types and creating a new variable that combined the latitude and longitude values. This alteration enabled us to utilize the combined variable effectively in subsequent analyses, particularly in relation to elastic search functionality. By employing these strategies, we aimed to enhance the overall quality and relevance of the extracted features from the earthquake dataset.

In the data analysis phase of our study, we employed Apache Spark as a powerful tool for analyzing earthquake and electrical datasets. Initially, we conducted a comprehensive analysis by computing descriptive statistics for various features present in the datasets. Specifically, we focused on the following features: depth, magnitude, longitude, latitude, eventID, and rms.

The computed descriptive statistics offer valuable insights into the distribution and characteristics of the data. For each numeric feature, we computed statistics such as count, sum, mean, variance, standard deviation, maximum, and minimum values. These statistics provide a holistic view of the datasets, allowing us to understand the data distribution and extract key characteristics. This information serves as a foundation for further analysis and decision-making processes.

Furthermore, in our analysis, we explored specific regions and earthquakes of magnitude 4 or higher along fault lines in three distinct regions: Doğu Anadolu, Kuzey Anadolu, and Batı Anadolu. By examining earthquakes with a magnitude of 4 or higher along these fault lines, we gained insight into the seismic activity within these regions. Additionally, we investigated the cities associated with these earthquakes.

To delve deeper into the earthquake data, we utilized Apache Spark's RDD (Resilient Distributed Dataset) functionalities to perform filtering and mapping operations. We filtered and counted earthquakes with a magnitude of 4 or higher based on their location on different fault lines. This analysis provided valuable information on the distribution of earthquakes in terms of fault lines and magnitudes, aiding in the understanding of seismic activity patterns across different regions.

Furthermore, we employed grouping and counting techniques to examine the location-based distribution of earthquakes. The earthquakes based on their province (key) and subsequently counts the number of earthquakes within each group. This analysis enables us to identify regions with higher seismic activity and gain insights into the geographical distribution of earthquakes.

VII. Correlation Between Earthquakes and Electric Field Data

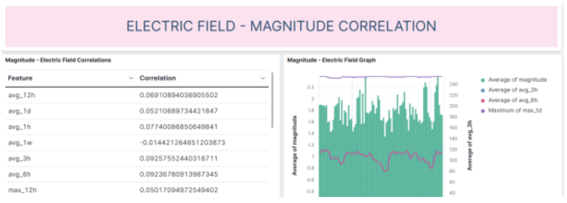
In order to examine the potential correlation between weekly electric field data and earthquake magnitudes, we conducted a correlation analysis. Our objective was to determine whether changes in the electric field could impact earthquake magnitudes. For this analysis, we retrieved two datasets from the path '/home/ubuntu/parquet-files/el_weekly/': one containing electric field data and the other containing earthquake data (Figure 2).

To begin, we merged the electric field and earthquake datasets based on their respective dates, using an inner join operation. This allowed us to establish a connection between the two datasets and perform joint analysis.

Subsequently, we focused on the electric field data and calculated statistical measurements, such as maximum, minimum, and average voltage fluctuations, for specific time intervals. These measurements provided insights into the characteristics and variations of the electric field data within the given time range.

In the next step, we calculated the correlations between the electric field data and earthquake magnitudes. This analysis aimed to quantify the relationship between these variables and assess whether any significant correlation existed.

Moving on to data export, we utilized Elasticsearch, a widely used distributed search and analytics engine, to export the earthquake dataset. We defined the features and their corresponding data types, and then inserted the earthquake data into Elasticsearch for further analysis and exploration.


ELECTRIC FIELD - MAGNITUDE CORRELATION

Magnitude - Electric Field Correlations

Feature	Correlation
avg_12h	0.06910894036905502
avg_1d	0.05210689734421847
avg_1h	0.07740086850649841
avg_1w	-0.014421264651203873
avg_3h	0.09257552440318711
avg_6h	0.09236780913967345
max_12h	0.05017094972549402

Figure 2. Electric field - magnitude correlation

Additionally, we exported the correlation data to Elasticsearch for further utilization. The correlation data contained information about the newly created features and their correlations with earthquake magnitudes. This allowed us to explore and examine the relationships between different variables.

By exporting the earthquake and correlation data to Elasticsearch, we created opportunities for advanced search, analysis, and visualization of the datasets. Elasticsearch's capabilities enable efficient querying and exploration, facilitating deeper insights into the relationship between electric field changes and earthquake magnitudes.

VIII. Visualization

For efficient operation and performance, Elasticsearch and Kibana were installed on a separate EC2 instance. To enable external access, a specific port is assigned and made public for Kibana. Data transmission to Elasticsearch was facilitated through Spark. Two separate indexes were created in Elasticsearch to store the earthquake data received from Kafka and the correlation data derived from the analysis between earthquake and electric field data.

Prior to transmitting the earthquake data to Elasticsearch, certain modifications were made. Firstly, fault line data was incorporated into the earthquake dataset. A function was implemented to assign fault lines based on provinces, determining the fault line associated with each incoming earthquake and adding it to the dataset as a new column named "fault_line."

Another modification involved the creation of a "pointLocation" field in string format by combining the latitude and longitude data in order to generate geo-point type data for mapping purposes in Kibana. This transformation was achieved through a mapping function applied to the RDD.

The resulting data transmitted to Elasticsearch, initially appeared to contain all fields as strings. However, necessary adjustments were made to the data types by creating an index template. This template facilitated the correct interpretation of the data, ensuring that the pointLocation field was recognized as a geo-point, while the depth, magnitude, latitude, longitude, and eventID fields were treated as floating-point numbers.

In this project, the earthquake data was sent to the "earthquake-all" index. To enable visualization in Kibana, an index pattern named "earthquake-" was created, allowing access to the data stored in this index. Additionally, an index template named "geo-point" was created to define the data types for the `earthquake-*` index pattern. This template ensured the conversion of the pointLocation field from a string to a geo-point format, and the conversion of depth, magnitude, latitude, longitude, and eventID fields to floating-point numbers. These modifications enabled proper indexing and accurate visualization of the earthquake data in Kibana.

The dashboard created for visualization purposes primarily focuses on displaying relational graphs and grouped forms of earthquake data. It incorporates filters to refine the displayed data, allowing users to filter earthquakes based on fault line information and select a specific range of earthquake magnitudes (Figure 3).

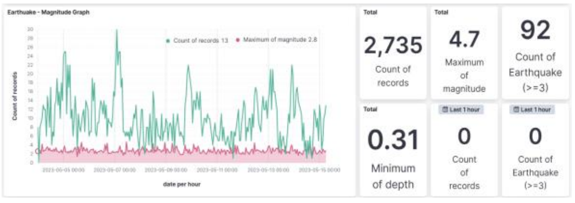


Figure 3. Information table

One of the key visualizations in the dashboard is a mixed chart comprising line and area charts, which presents the maximum earthquake magnitude and the number of earthquakes within specific time intervals. Alongside this graph, a summary field provides concise information about the earthquakes within the selected time range.

In the subsequent section of the dashboard, pie charts illustrate the distribution of earthquakes across provinces and fault lines. Additionally, a graph showcases the relationship between the maximum magnitude and average depth of earthquakes (Figure 4).

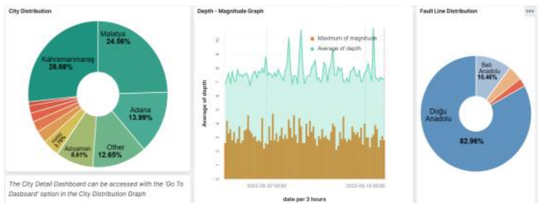
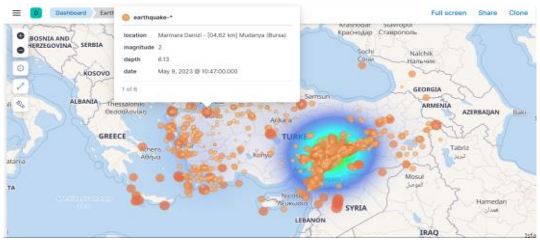


Figure 4. Average depth of earthquakes

To provide a spatial representation of the data, a map has been incorporated into the dashboard, displaying the earthquake data in two distinct layers. The first layer is a heat map indicating areas of high earthquake intensity, while the second layer represents individual earthquakes as points on the map (Figure 5).

Furthermore, a drilldown functionality has been implemented to enable further exploration of the data. By clicking on a city in the City Distribution graph, users can access a separate dashboard that visualizes the analysis of each province in greater detail.


earthquake-*

location Marmara Denizi - [04.62 km] Mudanya (Bursa)
magnitude 2
depth 6.13
date May 9, 2023 @ 10:47:00.000

1 of 6

Figure 5. Map of the realized earthquakes

Lastly, the dashboard includes the correlation analysis data between the electric field and earthquake magnitude obtained through Spark. This information provides insights into the potential relationship between changes in the electric field and earthquake magnitudes.

The created dashboard can be accessed through the provided link through GitHub repository, allowing users to interact with and explore the visualized earthquake data and correlation analysis results.

IX. Conclusion

The application of big data methods and technologies, including Apache NiFi, Apache Kafka, Apache Spark, Elasticsearch, and Kibana, has significantly transformed earthquake research by enabling the efficient collection, processing, analysis, visualization, and correlation of vast amounts of seismic and electric field data. This paper has highlighted the importance and advantages of utilizing these technologies in earthquake visualization and exploring the relationships between seismic events and electric field measurements.

Through the integration of Apache NiFi, researchers can seamlessly acquire, integrate, and preprocess diverse datasets from multiple sources, ensuring comprehensive data collection for earthquake research. Apache Kafka facilitates the reliable and scalable transmission of data, enabling real-time or near real-time processing and analysis. Apache Spark empowers researchers to leverage distributed computing capabilities to process and analyze large-scale datasets, uncovering correlations and patterns between earthquakes and electric field data.

The storage, retrieval, and visualization of earthquake data are efficiently supported by Elasticsearch and Kibana. These technologies provide powerful search and analytics capabilities, enabling researchers to index, store, and explore earthquake data in real-time dashboards, graphs, and maps. The visualization capabilities offered by Kibana facilitate the identification of trends, anomalies, and spatial patterns, enhancing earthquake monitoring and risk assessment.

By leveraging the potential of big data methods and technologies, researchers can unlock valuable insights into earthquake behavior, improve prediction models, and contribute to more effective earthquake-related decision-making and mitigation strategies. The comprehensive analysis of seismic events and their correlation with electric field data can enhance our understanding of earthquake mechanisms and potential precursors.

In conclusion, the adoption of big data methods and technologies in earthquake research has revolutionized the field, offering new avenues for exploration and discovery. As technology continues to evolve, it is crucial for researchers and practitioners to embrace these advancements and leverage the power of Apache NiFi, Apache Kafka, Apache Spark, Elasticsearch, and Kibana to advance earthquake research, improve monitoring systems, and ultimately contribute to the safety and resilience of communities affected by seismic events.

Acknowledgment

The authors would like to express their sincere appreciation to Prof. Burak Berk Üstündağ from Istanbul Technical University (ITU), who is the head of the project titled "Development of Seismic Risk Monitoring Method with Local Stress-Strain Measurements and Data Fusion Based on Deep Learning," granted by TÜBİTAK. We extend our gratitude to Prof. Üstündağ for generously providing us with the electric field data used in this research.

The authors would like to extend their heartfelt gratitude to Serkan Macit, one of the researchers involved in the TÜBİTAK project, for his invaluable support, technical expertise, and insightful discussions have significantly contributed to the success of this research.

References

[1] 'Developing post-alert messaging for ShakeAlert, the earthquake early warning system for the West Coast of the United States of America | Elsevier Enhanced Reader'.

[2] M. D. Kohler et al., 'Earthquake Early Warning ShakeAlert System: West Coast Wide Production Prototype', Seismological Research Letters, vol. 89, no. 1, pp. 99–107, Jan. 2018

[3] M. D. Kohler et al., 'Earthquake Early Warning ShakeAlert 2.0: Public Rollout', Seismological Research Letters, vol. 91, no. 3, pp. 1763–1775, May 2020, doi: 10.1785/0220190245.

[4] QuakeML - QuakeML'. https://quake.ethz.ch/quakeml/ (accessed Apr. 09, 2023).

[5] D. Schorlemmer, F. Euchner, P. Kästli, and J. Saul, 'QuakeML: Status of the XML-based seismological data exchange format', Annals of Geophysics, vol. 54, no. 1, pp. 59–65, 2011, doi: 10.4401/ag-4874.

[6] M. Pagani et al., 'OpenQuake Engine: An Open Hazard (and Risk) Software for the Global Earthquake Model', Seismological Research Letters, vol. 85, no. 3, pp. 692–702, May 2014, doi: 10.1785/0220130087.

[7] M. Pagani et al., 'The 2018 version of the Global Earthquake Model: Hazard component', Earthquake Spectra, vol. 36, no. 1_suppl, pp. 226–251, Oct. 2020, doi: 10.1177/8755293020931866.

[8] A. Donnellan et al., 'QuakeSim: Integrated modeling and analysis of geologic and remotely sensed data', in 2012 IEEE Aerospace Conference, Mar. 2012, pp. 1–9. doi: 10.1109/AERO.2012.6187219.

[9] T. Petersen, K. Gledhill, M. Chadwick, N. H. Gale, and J. Ristau, 'The New Zealand National Seismograph Network', Seismological Research Letters, vol. 82, no. 1, pp. 9–20, Jan. 2011, doi: 10.1785/gssrl.82.1.9.

[10] M. Avvenuti, S. Cresci, A. Marchetti, C. Meletti, and M. Tesconi, 'EARS (earthquake alert and report system): a real time decision support system for earthquake crisis management', in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, in KDD

[11] X. Wang, A. Dou, X. Ding, and X. Yuan, 'The Development of Rapid Extraction and Publishing System of Earthquake Damage Based on Remote Sensing', in IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium, Jul. 2018, pp. 2921–2924. doi: 10.1109/IGARSS.2018.8517663.