UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Development and optimization of AI algorithms for the categorization of small texts from social media data related to disasters

Filipe Pedro Félix de Sousa

Mestrado em Engenharia Informática

Dissertação orientada por:
Prof. Doutor Nuno Cruz Garcia

2023

Acknowledgements

I would first like to express my gratitude to my supervisor Nuno Cruz Garcia for all his support during the development of this dissertation.

I would also like to thank INOV INESC Inovação for giving me the opportunity to develop this dissertation and my project manager Joana Rosa for helping me a lot in carrying out this work and for always being comprehensive about the delivery times of the dissertation.

In addition, I want to thank my INOV colleagues Nuno Antunes, Alexandre Vidal, and José Pereira for their essential help in the development of this work. This dissertation would have been considerably more difficult without them.

Finally, I want to express a special gratitude to my parents for all their help during my academic career. I wouldn't have been able to finish my degree, enroll in the master's degree of my choosing, or study the subject I like the most without their help.

Para os meus pais.

Resumo

As redes sociais constituem uma forma de comunicação adoptada por milhões de pessoas em todo o mundo para partilhar informações e estabelecer contactos. Uma das redes sociais mais famosas do mundo é o Twitter, que é frequentemente utilizada para dar opiniões e relatar notícias ou situações que foram testemunhadas pelos próprios utilizadores. Em caso de emergência, algumas destas situações vividas pelos utilizadores podem ser úteis para identificar pedidos críticos de ajuda. Estes pedidos tornam-se mais frequentes durante desastres em massa (como ataques terroristas ou desastres naturais), pelo que a capacidade de detetar pedidos críticos vindos do Twitter pode melhorar os serviços de socorro, optimizando os recursos disponíveis e ajudando no processo de tomada de decisão durante a ocorrência de um determinado desastre em massa.

O sistema desenvolvido neste trabalho tem como objetivo monitorizar as redes sociais durante desastres em massa com recurso a critérios de pesquisa relevantes que permitam detetar informação crítica de suporte à decisão que será posteriormente catalogada com recurso a algoritmos de IA. Este trabalho foi realizado no âmbito do projeto *Nightingale* financiado pelo programa *European Union's Horizon 2020*. A principal motivação deste projeto é melhorar o suporte de vida pré-hospitalar em cenários de catástrofe onde os recursos e o tempo de atuação são limitados, proporcionando às pessoas cuidados de saúde e de socorro de alto nível que a tecnologia moderna e os actuais sistemas de proteção civil podem oferecer.

Numa primeira fase do trabalho, foi selecionado um dataset relevante para o contexto, que consistiu num conjunto de 54 342 tweets relacionados com diversos desastres passados (naturais ou de causa humana) que foram posteriormente categorizados por voluntários com recurso a ferramentas de categorização disponíveis online. De seguida, foram definidas as categorias essenciais para a categorização tendo em conta o contexto do problema: "injured or dead people", "missing trapped or found people", "other useful information"e "non-urgent". As categorias em que se encontravam categorizados os dados foram então mapeadas para estas novas categorias. Durante este processo de preparação de dados, foi também necessário encontrar uma estratégia para resolver um problema de equilíbrio de dados relacionado com a classe "missing trapped or found people", devido à pouca representatividade da mesma.

Após a preparação dos dados, foi necessário definir as estratégias de processamento de dados que seriam usadas, de modo a que os textos fossem transformados em vetores numéricos e consequentemente pudessem ser usados pelos modelos de IA.

No âmbito deste trabalho foram testados 9 algoritmos clássicos e 5 algoritmos de deep-learning. As técnicas de processamento de dados foram diferentes dependendo do tipo de algoritmo (clássico,

de deep-learning ou BERT). Após ser delineado um processo para a seleção do melhor algoritmo e de serem testadas diversas parametrizações dos mesmos, o BERT foi o algoritmo selecionado para abordar o problema uma vez que obteve uma performance consideravelmente superior às obtidas pelos restantes algoritmos (obteve uma média de 85% de precisão, 88% de recall e 86% de F1 score).

Para além do serviço de categorização, o sistema desenvolvido neste trabalho também inclui um algoritmo de correlação que tem como objectivo encontrar possíveis correlações entre as informações retornadas pelos centros de chamadas de emergência e os tweets. Obtivemos informação relacionada com incidentes PSAP (call center de chamadas de emergência) disponibilizado pelo *NG-ES* (sistema centralizado de software instalado no PSAP) no âmbito do projeto *Nightingale*. O nosso algoritmo de correlação apenas usa os textos das descrições (quer dos tweets quer dos incidentes PSAP) e as datas de criação dos mesmos. Este algoritmo de correlação envolve o cálculo de três tipos de métricas: similaridade entre os textos - mede o cosseno do ângulo entre os vetores que representam os textos; similaridade temporal - verifica a distância temporal entre as datas de criação dos textos; similaridade NER - verifica as entidades comuns em ambos os textos (como localizações, nomes de pessoas, nomes de organizações, etc.). Cada uma destas métricas retorna um valor entre 0 e 1, onde 1 representa o valor de correlação mais alto e 0 o valor de correlação mais baixo. No cálculo final, a similaridade NER e a similaridade entre textos têm ambos um peso de 37,5%, enquanto que a similaridade temporal tem um peso de 25%.

Após o desenvolvimento destes dois serviços, foi necessário modelar os dados e criar uma base de dados para armazenar toda a informação gerada e extraída pelo sistema. Decidimos usar uma base de dados relacional e criámos as seguintes tabelas para armazenar a informação: Evento-tipo; Evento; Keyword; Categoria; Informação das Redes Sociais; Informação PSAP; Previsão; Correlação.

Para a operacionalização do serviço, foram efectuadas 3 tarefas bastante importantes: criação dos trabalhos periódicos, criação de uma API e armazenamento de toda a informação do sistema em containers. Para que o sistema seja útil na monitorização de um desastre em massa, é importante que o mesmo esteja preparado para receber e extrair nova informação que vai sendo gerada e publicada durante um desastre. Por isso, de modo a que os serviços deste sistema fossem realizados periodicamente, criámos os trabalhos periódicos. O primeiro trabalho periódico envolve todo o processo de extração, categorização, armazenamento na base de dados e correlação dos tweets. O segundo trabalho periódico envolve a extração da informação PSAP. Cada trabalho é efectuado de 30 em 30 minutos, de modo a que o sistema seja sempre atualizado com nova informação durante este intervalo de tempo.

De modo a partilhar a informação criada e extraída pelo sistema com parceiros e futuros utilizadores do nosso serviço (pessoas responsáveis pela monitorização de um determinado evento), criámos uma API que permite extrair diferentes tipos de informação. Por exemplo, podem ser feitos pedidos GET para extrair todos os tweets escritos a partir de uma determinada data ou extrair todos os tweets que estão relacionados com um determinado incidente PSAP ou obter as palavras

mais utilizadas em tweets que foram categorizados numa determinada classe.

Posteriormente o serviço foi operacionalizado através do uso de 4 *containers Docker*. Um *container* responsável por criar todos os componentes necessários para a criação e suporte da base de dados, outro responsável pela API, outro que executa o software necessário para a realização dos trabalhos periódicos e por fim um *container* que contém uma interface que facilita a visualização dos conteúdos da base de dados.

Neste documento é explicada a estratégia de equilíbrio de dados utilizada e o seu impacto na performance dos modelos de IA utilizados, são recolhidas métricas de performance desses mesmos modelos de IA e é ainda apresentado um caso de uso da operacionalização do sistema que tem por base os tornados ocorridos nos Estados Unidos da América em Março de 2023.

Após analisar os resultados obtidos, consideramos que a categorização da grande maioria dos tweets foi feita sem erros, a capacidade de detetar informação urgente nas redes sociais e a possibilidade de encontrar correlações entre as informações retornadas pelo serviço de chamadas de emergência e os tweets, fazem deste sistema uma ferramenta útil que pode complementar os métodos já existentes para a monitorização de desastres em massa. No entanto, este trabalho deixa ainda espaço para melhorias, uma vez que todo o sistema foi desenhado para textos de publicações escritas em Inglês. Através de um módulo de tradução, este serviço poderia ser bastante mais amplo e útil em diversos países, de modo a conseguir detetar informação urgente mesmo em tweets ou publicações escritas noutras línguas. O serviço de correlação poderia também ser melhorado e ter melhor performance se fossem disponibilizados mais dados pelo serviço de chamadas de emergência. No entanto, a operacionalização deste sistema constitui uma importante base de apoio à monitorização das redes sociais que são cada vez mais utilizadas para partilhar informação entre os cidadãos e permite automatizar e recolher pedidos críticos de ajuda em tempo real que podem salvar vidas em contextos operacionais desafiantes.

Palavras-chave: Aprendizagem Automática, Monitorização de Desastres, Processamento de Linguagem Natural, Mineração Textual e Análise de Redes Sociais

Abstract

Social media is an internet-based form of communication adopted by billions of people around the world to share information and make connections. One of the most famous social networks is Twitter, which is often used to give opinions and report news or situations that have been witnessed by the users themselves. In case of emergencies, some of these situations experienced by users can be useful to identify critical requests. These requests become more frequent during mass disasters, so the ability to detect critical requests coming from Twitter can improve rescue services by optimizing the resources available and helping in the decision-making process during the occurrence of a certain MCI (Mass Casualty Incident).

The system developed in this work aims to monitor social networks during MCIs, using relevant keywords to find tweets and AI algorithms to categorize tweet texts. By splitting the tweets into certain categories, it is possible to identify those that reveal urgent information. A correlation algorithm was also included in the system in order to identify possible associations between emergency call information and tweets.

The BERT model was chosen to address the problem after multiple experiments with machine-learning models since it achieved an average of 85% precision, 88% recall, and 86% F1 score in categorizing the tweets using the testing set.

The categorization of most tweets without errors, the ability to detect urgent tweets, and the possibility of correlating tweets with emergency call information make this system a useful tool that can complement the current methods used to monitor MCIs.

Keywords: Social Media Analysis, Machine Learning, Disaster Response, Text Mining, Natural Language Processing

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The occurrence of mass disasters, generally referred to as Mass Casualty Incidents (MCIs), often leave a profound effect on communities, posing significant challenges and prompting crucial responses for recovery and resilience. These incidents can be natural-caused (floods, droughts, earthquakes, storms, wildfires, etc.) or human-caused (accidents, terrorist-related events, armed conflicts, etc.), but regardless of the cause their impact is massive, leading to the different sectors of Civil Protection having to deal with limited resources, unpredictable amount of citizens in need of care whilst requiring numerous treatment protocols to be set in place, extreme difficulties in collaboration and communication, time pressure, and effort-consuming procedures in uncharted surroundings. The most common MCIs are natural disasters which have already affected 106.15 million people and caused 12 886 deaths, and 26 345 injuries since 2020 [39]. Accidents and terrorist-related events affect 50 million people worldwide each year [40].

To mitigate the damage that these events cause, it is necessary for countries to be properly prepared to respond to them so that they are able to provide the best possible emergency services to the population. Most interactions between people and emergency services are done through calls (countries usually provide an emergency call number that people should use to report an emergency). Although nowadays most people have access to smartphones which can be useful for exchanging other types of information or data that can lead to fast and appropriate decisions by decision-makers and first responders. One of the types of information that can be helpful is information from social media because people often exchange information during mass disasters or emergency cases [18]. Therefore, the analysis of that information could be important to save people's lives. For example, during Hurricane Harvey in 2017 many people were rescued not by first responders but by fellow citizens responding to requests for help on social media [18].

One of the most famous social networks is Twitter, which is often used to share and publish small texts, videos, and photos. It has been extensively used as an active communication channel, especially during mass casualty incidents [29], so Twitter can be the ideal platform to find critical requests for help during MCIs. However, there is a large information overload on this social network, and most of that information is related to non-urgent tweets, i.e., tweets that do not contain important information about critical requests, which represents noisy information for the decision-makers and first responders. Tweet text categorization can be very helpful to detect

important information about emergency requests in the midst of a lot of noisy information. Then, this urgent information can be sent to decision-makers, who will have more detailed information to allocate resources, which are often limited during MCIs considering the number of victims.

This categorization process consists of classifying each incoming Tweet according to a predefined set of classes that reveal its nature (type of information that is portrayed in it) using Machine-Learning algorithms. For instance, Tweets that reveal helpful information about injured people and Tweets that reveal emotional support belong to different classes.

Machine-Learning can be divided into two main approaches: supervised and unsupervised. Unsupervised learning uses Machine-Learning (ML) algorithms to analyze unlabeled data, i.e., these algorithms discover hidden patterns in new data without historical data or human intervention. Supervised Machine-Learning algorithms use historical data to identify patterns and predict output values for new data [2]. Supervised learning was the approach used to address this work because texts of disaster-related tweets categorized by humans were used to try to predict the categorization of new tweets.

There are several ML models, which have distinct mathematical methods to identify these patterns in the data. The performance of these algorithms depends on the type of problem that they are dealing with, so it is necessary to understand which algorithm is the best to address the problem of this work. Therefore, these models have to be trained and evaluated with an appropriate amount of data. The training process consists in detecting patterns in the data so that later the algorithms can predict outputs and the evaluation phase is the process of using metrics to understand the ML model's performance.

The data consists of tweets that reveal human language in the form of text, which requires preparation and processing processes before being used by the algorithms. These processes are accomplished through the use of Natural Language Processing (NLP) techniques.

Natural Language Processing is a set of methods for making the human language accessible to computers and is increasingly part of our daily lives - text classification keeps our email inboxes from collapsing under a deluge of spam, automatic machine translation is ubiquitous on the web and in social media, search engines have moved beyond string matching and network analysis to a high degree of linguistic sophistication, etc. [23]

In this work, datasets of human-labeled disaster-related tweets were transformed and processed through data mining and NLP techniques. Using the data resulting from this transformation and processing, several machine-learning models were trained and evaluated. The best-performing algorithm was chosen to address the problem. Then, Twitter was monitored through search tools and specific keywords to find relevant information about specific MCIs. An automated system was then designed to store, categorize and correlate these tweets with information generated by the responsible entities after emergency calls. Afterward, certain mechanisms were used so that this system could run periodically.

1.1 Motivation and Scope

This dissertation aims to monitor Twitter (one of the most famous social networks in the world to share small texts) to find relevant information that is posted online by citizens during MCIs.

The work of this dissertation was jointly developed at Faculdade de Ciências da Universidade de Lisboa (FCUL) and Instituto de Engenharia de Sistemas e Computadores Inovação (INOV-INESC) and comes within the framework of a project called Nightingale, which is a large-scale research and innovation project funded by the European Union's Horizon 2020 program. The main motivation and challenge of this project is to improve pre-hospital life support and triage by providing affected people top-level health care that modern technology and current civil protection systems can offer.

In the last decades, cases of major emergencies, crises, and disasters have become more frequent. In these situations, responsible authorities deal with limited resources and face problems in allocating them, especially because there are usually so many people in need of help. Therefore, the Nightingale project aims to arm medical and public safety agencies with all the tools that modern technology has to offer in order to minimize these difficulties.

This project includes several components and partners, however, the following three are the ones that were most involved in this work:

1. *Nightingale - Emergency System (NG-ES)*: a centralized software system, to be installed at 112 Public Safety Answering Points (PSAP) - entity responsible for receiving emergency calls and processing those calls according to a specific operational policy - in Member States, that enables multimedia streams between citizens and PSAP;

2. *SoftWare mobile APPlication for citizens (SWAPP)*: a smartphone application to be installed by citizens that contains an offline and an online mode, and whose purpose is to improve emergency response, bring modern communications and data exchange capabilities to citizens in emergencies;

3. *ECHO*: a software component that is connected to SWAPP and that works as a situational awareness tool for emergency management services designed to display a large volume of emergency-related information, enabling the generation of the "big picture" of ongoing incidents and allowing professionals to understand truly the nature and level of the emergencies.

The system developed in this work is within a service called Social Media Service which is a component integrated into the ECHO component to enhance situational awareness on the field by incorporating crowdsourcing information from open sources. This service will be also used to complement the information already available for PSAP operators and field responders, as described in Figure 1.1, which also shows all the features of this service. The categorization of texts from Twitter is a task of the "Identification of critical information" feature, and the correlation of tweets with information generated by the responsible entities after emergency calls (PSAP

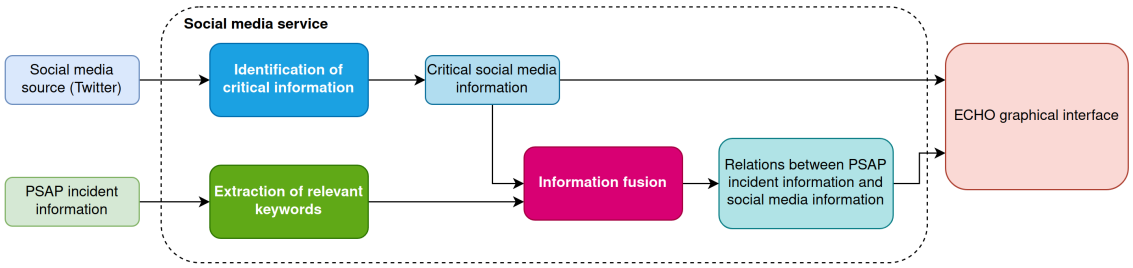

Figure 1.1: Social Media Service features and their integration in the ECHO component

information) is the ”Relations between PSAP incident information and social media information” feature.

1.2 Objectives

Considering the aforementioned motivation and challenges, the research of this dissertation aims to help decision-makers allocate resources by giving more detailed information about existing incidents or new information about incidents that were not known yet.

Therefore, this work seeks to develop a periodic and automated system able to retrieve tweets related to specific MCIs, categorize them using the best possible ML model to address the problem, and correlate them with PSAP information. Lastly, it is necessary to make available for stakeholders and partners the information related to tweets that were classified with urgent classes as well as their correlations with PSAP incidents.

There are a few elements incorporated into this system that are really important for its operation, such as:

1. Tweets Search Tool using keywords
2. Good-performing Machine-Learning algorithm that is responsible for the information categorization process
3. Database to store all system information (tweets, categories, keywords, correlations, etc.)
4. Service to retrieve PSAP information periodically
5. Correlation Algorithm able to identify possible associations between tweets and PSAP incident information

These elements exchange data between themselves, are dependent on each other, and all have been developed in the work carried out in this dissertation. The interactions between them are necessary for the system’s proper functioning and are fully detailed in Chapter 3.

This system/artifact pretends to answer the following three research questions of this work:

1. *Can relevant information really be detected amidst a lot of non-urgent information?*

2. *Can the selected Machine-Learning Algorithm ensure that the vast majority of tweets are categorized without error?*

3. *Is it possible to create an algorithm able to detect correlations between tweets and the information that is generated after emergency calls (PSAP information)?*

1.3 Contributions

In order to achieve this dissertation's objectives and answer the aforementioned research questions, several techniques from different areas were used. During the initial phase of this work, the areas of Data Mining, Machine-Learning, and Natural Language Processing were crucial for preparing and processing the data used for training Artificial Intelligence (AI) models. Other areas such as Data Modeling, Service Scheduling, and Docker Containerization/Orchestration were also important for the operationalization phase.

This work also contributes to the advancement of emergency services responses to MCIs. This dissertation not only reports the best ML model to address the problem and the best NLP techniques needed to process the tweets, but it also offers a system that can be a solution for the use of relevant crowd-sourced information by stakeholders and decision-makers in the fight against MCIs.

The contributions of the present work will be implemented and evaluated within the scope of the Nightingale European Project.

1.4 Document structure

This dissertation is composed of 6 different Chapters, including the Introduction. It is structured as follows:

- **Chapter 2** - presents the summary of the entire related work that was useful for the development of this dissertation, including datasets of disaster-related tweets, NLP techniques, results of Machine-Learning models, data extraction techniques, and textual similarity metrics, as well as the selection system used for the inclusion of papers in our state-of-art

- **Chapter 3** - provides an overview of the entire system. **Section 3.1** shows and explains the architecture of the system. **Section 3.2** shows a high-level view of each work module that was necessary to develop.

- **Chapter 4** - provides the full details of the development of each module. **Section 4.1** presents the whole process of searching tweets related to the MCI events. **Section 4.2** reveals the implementation of the whole categorization process. **Section 4.3** shows the developed data model and details about the relational database implementation. **Section 4.4** illustrates all the development required for the correlation between tweets and PSAP information. **Section 4.5** shows the entire operationalization process of the system.

- **Chapter 5** - demonstrates the functionalities and results of the system developed. **Section 5.1** shows how the data balancing process improved our results and also illustrates the caliber of the synthetic data generated following this balancing process. **Section 5.2** shows the results of all models tested in the development of the categorization process, as well as reveals which model was chosen to address the problem. **Section 5.3** exemplify and demonstrate in detail the monitoring of a particular event using our system.

- **Chapter 6** - provides the conclusions of this dissertation. **Section 6.1** summarized the conclusions drawn from the system developed in this dissertation. **Section 6.2** explains the limitations of this work. **Section 6.3** shows what could be improved and leaves some recommendations for future research on the subject.

Chapter 2

Related Work

Social media platforms serve as active communication channels during mass and emergency events [29]. Its data can be used for disaster-specific data analysis, which can be used to extract disaster logistics, identify patterns, and provide public preparedness assistance. Twitter and Facebook have reached an irreplaceable stature in disaster management due to the availability of colossal data that can be collected [6]. Twitter has been particularly used in emergency cases since the 2011 Great East Japan Earthquake [42].

During the last few years, several works and research have been carried out to identify critical requests for help on social media during mass disasters or major emergencies. Many of these studies also used Twitter as the main data source: Human-annotated datasets of tweets have been created, many NLP techniques were used to process tweet texts, and different machine-learning algorithms were tested to classify tweets. Beyond these works, there has been also further research related to the similarity/correlation between texts, and the use of different tools to extract data from Twitter.

Google Scholar and *Scopus* were used as research databases for this dissertation. In order to find the most useful research papers, the *Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA)* methodology [34] was used, as well as the following filtering criteria that were previously defined:

- Publication date from 2011 onwards
- Language used must be Portuguese or English
- The content must be related to tools/techniques/data that can be used in the context of the problem that this dissertation intends to solve
- Only free papers or non-free papers within the FCUL's license are eligible, and priority is given to scientific papers published in journals or conferences

PRISMA methodology is composed of several filtration stages, namely: Identification - involves all papers returned from the research databases; Screening - involves all papers that were screened; Eligibility - involves all papers that satisfy the criteria and are eligible to be used; Included - corresponds to the papers that were actually used in this dissertation's work.

Filtering phase	Number of studies
Identification	2130
Screening	132
Eligibility	18
Included	4

Table 2.1: Number of studies involved in the different filtration stages related to query 1

Filtering phase	Number of studies
Identification	16200
Screening	154
Eligibility	9
Included	2

Table 2.2: Number of studies involved in the different filtration stages related to query 2

Firstly, the following search query was used in order to search datasets with disaster-related tweets, find the best data processing techniques, and understand which are the best machine-learning algorithms to address the problem:

(Human-annotated AND NLP AND Twitter AND datasets) OR (Embeddings AND Twitter data AND disasters) OR (Machine-learning AND social-media) [1]

Table 2.1 shows the number of papers involved in each filtration phase related to query 1. 2130 research works were identified, 132 were screened, 18 were eligible to be used, however only 4 research works related to datasets with disaster-related tweets, data processing techniques, and machine-learning algorithms were used in this dissertation's work.

Then, in order to analyze the different methods of extracting data from Twitter, the following search query was used:

Extract data from Twitter AND API AND Techniques AND Analyze Twitter data [2]

Table 2.2 shows that only 2 research works related to data extraction from Twitter were included in this work, although 16200 have been identified, 154 screened, and 9 were considered eligible.

Finally, seeking to understand mechanisms for the correlation between texts, the following query was used in the research:

(Text Similarity AND Metrics AND Measurement) OR (NER AND Entities AND Models) [3]

Table 2.3 shows that 16500 studies were identified, 141 were screened, 17 were eligible to be used, but only 2 were included in this work.

Filtering phase	Number of studies
Identification	16500
Screening	141
Eligibility	17
Included	2

Table 2.3: Number of studies involved in the different filtration stages related to query 3

In total, 8 research works were included in the state-of-art of this dissertation, 4 related to disaster-related datasets, data processing techniques, and machine-learning algorithms, 2 related to data extraction from Twitter, and 2 others about the correlation between texts.

2.1 Disaster-related Datasets

Over the years, many datasets have been built to address this dissertation work's problem. Different data collection processes, annotation schemes, and data annotation processes have been used.

In 2016, Muhammad Iran et al. [29] presented a dataset of human-annotated messages that were pulled from Twitter during 19 different disasters that took place between 2013 and 2015. They collected these messages through the Artificial Intelligence for Disaster Response (AIDR) platform that was created by Muhammad Iran et al. in 2014 [28]. This platform uses Twitter's streaming Application Programming Interface (API) to collect data through different strategies, for example, collecting tweets that contain specific keywords.

They also defined their annotation scheme which was composed of 9 different classes: 1) Injured or dead people; 2) Missing, trapped, or found people; 3) Displaced people and evacuations; 4) Infrastructure and utility damage; 5) Donation needs or offers or volunteering services; 6) Caution and Advice; 7) Sympathy and emotional support; 8) Other useful information; 9) Not related or irrelevant. Their annotation process was the following: first, the tweets were classified by volunteers, then if there was a very high-class imbalance, they used a paid crowdsourcing platform.

After a few years, in 2020, Ashwin Devaraj et al. [18] used over 5 million tweets posted during 2017's Hurricane Harvey in Houston to detect life-or-death nature tweets. They collected a dataset of tweets directly purchased from Twitter via its data reseller, Gnip. Regarding the annotation scheme, they think that the scheme used by Muhammad Iran et al. [29] is too broad to be directly used by first responders, so they used a labeling scheme with just two labels - urgent and not urgent. The tweets are classified as urgent if 1) the user is urgently requesting help; 2) the tweet suggests that people's lives are in danger and/or rescue is necessary; 3) specific information about the danger is included in the text. The other tweets are classified as not urgent. In this dataset, all the tweets were manually labeled by 2 authors, with intercoder reliability measured to be 1.0 (100% of agreement) on a random sample of 70 tweets.

In 2021, Firoj Alam et al. [5] presented a new large dataset with 77 000 human-labeled tweets across 19 disaster events that happened between 2016 and 2019. Like Imran et al. in 2016 [29], they also performed the AIDR platform [28] with event-specific keywords and hashtags to collect data. Their annotation scheme was composed of 11 different labels: 1) Caution and advice; 2) Displaced people and evacuations; 3) Infrastructure and utility damage; 4) Injured or dead people; 5) Missing or found people; 6) Not humanitarian; 7) Don't know or can't judge; 8) Other information; 9) Requests or urgent needs; 10) Rescue, volunteering or donation effort; 11) Sympathy and support. For the manual annotation, they used the Amazon Mechanical Turk (AMT) platform,

which is a crowdsourcing platform [44]. To evaluate the workers of this crowdsourcing platform, they created a qualification test, to pass the test, the annotator needs to correctly answer 6 out of 10 tweets. Only annotators who passed the test participated in the task.

Their dataset also went through 6 filtration steps: 1) Date-based filtering - they restricted the data sampling period to the actual event days because some tweets are made after the disaster; 2) Location-based filtering - they discarded all tweets outside the disaster-hit areas; 3) Language-based filtering - they deleted all tweets that were not written in English; 4) Classifier-based filtering - They trained a Random Forest classifier to classify and eliminate all the irrelevant tweets (tweets classified as not-humanitarian); 5) Word-count-based-filtering - they retained tweets that contained at least three words or hashtags, but URLs and numbers were discarded; 6) Near-duplicate filtering - they removed exact and near-duplicate tweets using their textual context.

2.2 Data Processing Techniques

Twitter texts have to be processed and transformed before being used to train machine-learning models. This process can be done in different ways and with the use of different techniques. Some of these processes may lead to better results than others.

Muhammad Iran et al. [29] started by removing stop-words, URLs, and user mentions from Twitter messages. Then, to represent the words in the text, they used word embeddings (distributed word representations for text analysis) that were generated using Continuos Bag Of Words (CBOW) architecture [33] with negative sampling along with 300-word representation dimensionality. They also trained word embeddings using 52 million Twitter messages through word2vec, which is a very popular software to train word embedding [31]. Finally, they normalized Out-Of-Vocabulary (OOV) terms that were present in the texts. To normalize these terms, they used the CrowdFlower crowdsourcing platform in which the workers read the given messages and provide a tag that specifies the type of the OOV word present in the text (i.e. slang/abbreviation/acronym, a location name, an organization name, a misspelled word, or a person name). If an OOV is a misspelled word, the worker should also provide its correct form.

Ashwin Devaraj et al. [18] used another procedure. First, they split the tweet into tokens (individual logical symbols that make up human language such as words, numbers, and punctuation), for instance, the sentence 'The world is beautiful!' is transformed into the following list of tokens: ['the', 'world', 'is', 'beautiful', '!']. To do this tokenization process, they first put the text in lowercase and then used the Python Natural Language Toolkit (NLTK)'s [7] TweetTokenizer class. To represent URLs, usernames, and numbers they used the tokens [URL], [user], and [number] because they realized that are too many different users, numbers, and URLs to represent individually with distinct tokens. They also removed the hashtag symbol, e.g., #flood becomes flood. After this step, they applied lemmatization, which is a process to reduce different forms of a word to one single form ('lemma'), e.g., the words 'building' and 'builds' are reduced to 'build'. They used WorldNet Lemmatizer found in the NLTK library for this step. To attempt to correct incorrectly-spelled words in each tweet, they used a Python implementation of the SymSpell word

segmentation tool [25].

After these two steps, it was necessary to represent each token with a numerical vector or matrix that represents its meaning. They tested different techniques, such as: 1) full word embeddings; 2) average embeddings; 3) unigrams without negation detection; 4) unigrams with negation detection; 5) unigrams and POS tags. The Convolutional Neural Network (CNN) model just takes a matrix of embeddings in which the rows are the tweet tokens and the columns are the embedding vectors of each token, i.e., full-word embeddings. For the non-convolutional models that were used (Naive Bayes, Decision Tree, AdaBoost, Support Vector Machine (SVM), Multi Layer Perceptron (MLP), Logistic Regression, and Ridge regression), they compared the performance of average embeddings with the performance of unigrams and Part-Of-Speech (POS) tags. The average embedding used consists of the average of embedding vectors that represent the tokens of each tweet, e.g., [6,2,3] and [1,4,5] averaged together resulting in the vector [3.5,3,4]. This type of embedding produces a fixed-length vector of 100 numerical features.

An n-gram is a sequence of n-words observed in a text, e.g., 'That dog is pretty.' has the following bi-grams (n-gram of size 2) ['That', 'dog'], ['dog', 'is'], ['is', 'pretty'] and the following uni-grams (n-grams of size 1) ['That'], ['dog'], ['is'], ['pretty']. Since they noticed that uni-grams gave much better results than other types of n-grams, they only reported the uni-grams results. Uni-grams were tested with and without negation detection which consists in not representing negation words as individual tokens but rather merging them with the immediately adjacent word [35]. They also tested uni-grams with POS tags (counts of the number of different parts of speech for each tweet). The POS tags are added to the uni-gram feature sets without negation detection.

Their results showed that the SVM performance improves with the use of average embedding, however, the other machine-learning models had better results using uni-grams.

In 2021, Ashis Kumar Chanda showed the efficacy of Bidirectional Encoder Representations from Transformers (BERT) embeddings in predicting disaster-related tweets [13]. In his research work, BERT embeddings (contextual embeddings) were compared with BOW embeddings and context-free embeddings.

Bag Of Words (BOW) embeddings are a simple way of representing text. It is a binary vector of length equal to the number of vocabulary words, in which each position is relative to a word if the text contained that word, that position would have the value 1, otherwise, it would have the value 0. Ashis [13] tested this type of embedding in three machine-learning models: Decision Tree, Random Forest, and Logistic Regression. The downside of BOW embeddings is that it has no context information, which is important for text analysis.

Context-free embeddings learn word embeddings from the co-occurrences of words in documents. Ashis [13] tested 3 types of context-free embeddings: Skip-gram [**?**], GloVe [37], and FastText [9]. These types of embeddings were only used in neural network models.

Unlike context-free and Bow embeddings, BERT uses contextual information, i.e., it generates different vectors for the same word in different contexts. In [13], pre-trained embeddings of BERT models were used in the same neural network models as context-free embeddings.

After a few experiments, it was noticed that BERT embeddings outperformed context-free and BOW embeddings.

2.3 Machine-Learning Algorithms and Results with disaster-related data

Many machine-learning algorithms have already been tested in different research works and some tend to outperform others.

Muhammad Iran et al. [29] used Naive Bayes, SVM, and Random Forest to classify the tweets. They showed the Area Under ROC Curve [11] (evaluation metric) for all the classes of each disaster event dataset and realized that all three different classifiers had decent results (higher than 80%) for most of the existing classes, except for the "Missing trapped or found people" class, which was the smallest class in term of proportion. However, this research work was not very specific and clear in the evaluation of ML models.

Ashwin Devaraj et al. [18] showed more concrete results. They realized that SVM and CNN outperformed the other models (MLP, AdaBoost, Logistic Regression, Naive Bayes, Decision Tree, and Ridge Classifier). Both achieved an F1 score of 0.87. However, they dealt with a binary problem which is easier to classify than multi-class problems.

Unlike [18], the research work done by Firoj Alam et al. [5] dealt with a multi-class dataset. They also tested different ML models (most were transformer-based models) and realized that RoBERTa [30] outperformed the other models (Random Forest, SVM, FastText, BERT, and DistilBERT) with an average weighted F1 score of 0.781, which is an excellent result considering that there were 11 possible labels to classify the tweets. Between classical algorithms, SVM outperformed Random Forest. They also noticed that SVM and FastText had very close performances and that transformer-based algorithms outperformed the other types of models.

Ashis Kumar Chanda's research work [13] shows the performance of different ML models for different types of embeddings. Regarding classic models that were tested with BOW embeddings, Logistic Regression outperformed Decision Tree and Random Forest. In neural networks, Bidirectional Long-Short Term Memory (BI-LSTM) always outperformed Softmax regardless of the embedding type used.

2.4 Data Extraction from Twitter

Nowadays, data extraction is a vital task for most research works, business processes, and projects. This process consists of obtaining relevant data or metadata useful for diverse purposes through specific techniques from certain sources [21]. In this work, the data extraction process involves extracting disaster-related information from Twitter, i.e., disaster-related tweets.

In 2019, Priyanka Tyagi and Dr. R.C. Tripathi [45] collected Twitter data to perform sentiment analysis (analyzing text to determine if the emotional tone of the message is positive, neutral, or negative). They used a Twitter API called Tweepy to extract the tweets. The search of the tweets

was done through queries, which allowed to set filtering parameters so that tweets matched specific criteria. The output of these queries is relevant Twitter source information such as tweet id, tweet text, creation date, etc. Also, if any user makes his location public, data related to the location (latitude and longitude) from where the tweet was posted can be extracted, however, people have stopped sharing their location since 2012 due to security issues and client protections. In their research work, the Twitter data was directly embedded within the MYSQL database for being utilized.

Tweepy was the only tool used in [45] to extract the tweets, however, there are other tools for Twitter data extraction. Irvin Dongo et al. [21] presented a research work in which they compared Tweepy with another extraction technique: Web Scraping. The behavior and performance of both techniques were tested with Twitter data, and their advantages and disadvantages were discussed. They started by explaining both techniques and the main differences between them.

Web Scraping, also known as web extraction or harvesting, is a technique to extract data from World Wide Web (WWW) and save it to a file system or database. This technique can be useful for retrieving and processing large amounts of data quickly from a specific website, particularly in website pages that use markup languages such as HyperText Markup Language (HTML) or eXtensible HyperText Markup Language (XHTML) [49]. The web data scraper (which can be a software agent, a web robot, or a script) establishes communication with the target website through the HTTP Protocol and extracts the contents of interest.

This technique involves site-specific programming, which means that when the HTML source changes, the whole script must be updated and adapted to the new format. Web Scraping maintenance is critical and can involve time-consuming programming tasks. The network speed may also be another limitation of this technique.

On the other hand, an API allows developers to access data from a certain source through a reference program library. It is a component of object-oriented programming languages that uses an application program or an operating system in which a requester can make requests expecting responses from them. It also includes the specification of data structures, protocols, object classes, and runtimes, which facilitates access to their services. The API is called through an endpoint (the component that listens when a request is being made from the client side via HTTP and expects a response to be returned). One of the main disadvantages of APIs is that their access is limited due to data protection laws related to privacy.

Irvin Dongo et al. [21] have done several tests to compare the performance of both techniques in terms of Twitter data extraction. The Standard Twitter API was used, which is free and allowed 15 requests per 15 minutes. There are other types of Twitter APIs that are paid, however, the information provided by the Standard API may be sufficient depending on the project's needs. The negative point of this API is the access to only the last 7 days.

In the first experiments, a web scraper developed in [22] did not perform because of changes in Twitter HTML, then they developed another one but the Twitter HTML format changed again. So a few experiments with Web Scraping were done in [21], however, they were not able to validate

the results. Twitter API, in contrast, was able to easily manage dates, numbers, and other types of information in a different way from the one shown in the browser. After a few tests, Irvin Dongo et al. realized the main differences between web scraping and Twitter API in Twitter data extraction: the network speed and the constant changes in Twitter HTML affect the performance of web scraping but its use is free; Twitter API provides data in a structured format, is simple and easy to use, and it's not affected by the changes in Twitter HTML, however, the access to extract data is limited (15 requests per 15 minutes if using the Standard API) or not free (if using the Premium APIs). They also carried out some tests on the credibility of the tweets extracted by both techniques, which were measured through a credibility model proposed in [22] based on text, user, and social credibility. Results show that a solid normalization process on the text obtained by the two extraction techniques produces similar credibility values.

According to the experiment results reported in [21], Tweepy (Twitter API) seemed to be the best alternative to extract Twitter data when compared to web scraping, however, the research work was done to extract any type of tweets and this dissertation work essentially involves disaster-related tweets.

In 2022, Aswathy A. et al. [6] proposed a data collection system to capture disaster-related tweets, eliminate irrelevant information, and retain relevant data such as media and location without breaching social media ethics privacy policies. They started by creating a developer account (the type of Twitter account in which you can request access to Twitter data) and building an app in the developer account from which credentials are accessed once Twitter approves the account. Then, some tests were carried out to test three different Twitter libraries: Tweepy, Twython [1], and Python Twitter tools. Was noted that the use of Tweepy API has many advantages: offers a consistent output in terms of providing full text and invoking exclusion keywords; is able to collect many details associated with a tweet; is helpful for beginners because the documentation is up-to-date. Therefore, Aswathy A. et al. have chosen Tweepy to develop the mechanism for data collection in their research work.

2.5 Similarity and Correlation between texts

Currently, text similarity measurement plays an important role in several systems that use automatic question-answering, machine translation, dialogue systems, document matching, etc. It's a crucial NLP task that calculates how close two documents are to each other through metrics. [47]

Generally, during these types of NLP tasks, the text is represented by vectors. Jiapeng Wang and Yihong Dong [47] approached the three ways of measuring the distance between them: Length distance, Distribution distance, and Semantic distance. Distribution distance uses the statistical characteristics of the data and is used to compare whether documents come from the same distribution. Regarding the Semantic distance, it measures the distance of the vectors that represent text on a semantic level.

Length distance uses numerical characteristics of texts to calculate the distance between their vectors. The most popular Length distance metrics are Euclidean distance, Manhattan distance,

Hamming distance, and Cosine distance. Euclidean distance is simply the distance between two points in Euclidean space [16]. Manhattan distance consists of the sum of the absolute differences between the two vectors. Hamming distance is only used to compare binary vectors of equal length and corresponds to the number of positions in which the two bits are different. On the other hand, the Cosine distance measures the angle between two vectors in a multi-dimensional space. It is commonly used in NLP because it measures the similarity between two documents regardless of the magnitude, i.e., if a particular word appears 70 times in one document and 10 in another, there is a clear difference in magnitude, however, the documents can still be considered similar if only the angle is taken into account. [3]

In addition to this similarity analysis, it's also very important for this dissertation work to understand whether different texts refer to the same event, same location, same temporal space, etc. Named Entity Recognition (NER) is a process in which anything present in a certain sentence that can be denoted by a proper name or tag (a location, an organization, a person, etc.) is identified as an entity [41]. For instance, in the following sentence, the entities are marked: "Filipe Sousa [PERSON] worked on a dissertation that was jointly developed with INOV [ORGANIZATION] and FCUL [ORGANIZATION] in 2023 [DATE]". This sentence contains four entities: two labeled as ORGANIZATION, one labeled as PERSON, and another one labeled as DATE.

In 2020, Hemlata Shelar et al. [41] explained and compared the performance of different tools for NER models including Python's Spacy, Apache OpenNLP, and TensorFlow. The comparison between these tools was mainly focused on time, accuracy, and quality of performance.

TensorFlow is an open-source library for data programming [19], and it accepts only numerical data as input, so it's necessary to transform the text into numerical data. In the work developed in [41], they build a small NER model using TensorFlow and Keras. SpaCy [46] is another open-source library that was specially built to do NLP tasks and comes with pre-build NER models for a lot of languages. It utilizes a hybrid of Hidden Markov Models (HMMs) [8], decision tree analysis, and Maximum Entropy Models (MEMMs) [38].

Apache OpenNLP is a Machine-Learning-based toolkit that processes natural language text. This NLP library also provides pre-trained NER models for different languages and annotated text resources from which those models are derived. It uses Perceptron-based and Maximum Entropy models. The Name Finder of Apache OpenNLP is able to detect entities in the text, but it's necessary to have a model to detect them. An OpenNLP model is an entity-type-dependent model for the entity being trained. OpenNLP projects offer many pre-trained Name Finder models that are trained or different data, to find entities from the input text data. [4]

In the experiments carried out by Hemlata Shelar et al. [41], an International Business Machines corporation (IBM) dataset of advanced search panels which contains possible search queries to predict search statements was used to test the different NER tools. The TensorFlow NER model used was created by using Keras and Long-Short Term Memory (LSTM). They used the SpaCy 'en' model, which corresponds to the English language model, however, labels named attribute, criteria, and value had to be added in order to customize the model. Regarding Apache OpenNLP,

Token Name Finder was used in the experiments. In contrast to SpaCy, they verified that was not necessary to add labels, however, [START:] and [END] tags have been added to the training data.

Some conclusions could be drawn from the results of these experiments. TensorFlow proved to have good accuracy, however, they discovered a big problem when it came to separating the terms, i.e., the NER model tokenized the text data input which divides a single entity into two tokens, and it takes multiple entries for a single entity. For instance, if the text input contains South Africa, South and Africa would be classified as two different entries. They also noted that TensorFlow tries to remember the sequence of the labels without considering the relationship between labels and tokens, and if they label the same token in two different labels, the NER model will not be able to identify the differences and will always predict the wrong label.

The results also showed that OpenNLP had a better performance (in terms of accuracy) than TensorFlow, and it works with text data so it's not necessary to convert text data into numerical values. However, they found that it also addresses unknown tokens at the time of prediction and that the NER model size increases radically with the increasing size of training data.

SpaCy custom NER model outperformed the other models and proved to be a better and more accurate recognition system. It works directly with text data as OpenNLP, the prediction time is very low, the model in addition to giving the right prediction also adds layers of details to the output, and the training data size does not affect the NER model size.

Chapter 3

System Overview

The last chapter revealed several research works and concepts that were crucial for the development of the system present in this dissertation.

On the other hand, this chapter presents an overview of the system. We start by showing and explaining the architecture that was chosen to address the problem (in section 3.1) and then by explaining each work module that was necessary to develop (in section 3.2). The explanation of the modules will be relative to their functionality since the details of their development will be described in Chapter 4.

3.1 System Architecture

The system developed in this dissertation involves several work modules that communicate with each other in order to exchange data that are necessary for its proper functioning. Firstly, some keywords are associated with the MCI event that is intended to be monitored, and then these keywords are used to search and obtain the tweets related to the event. Tweets are then directed to the categorization process, in which each tweet is placed in a specific category according to its content. After that, Tweets information and their categorization are stored in a database.

On the opposite side of the system, information about emergency calls (PSAP information) is received from *NG-ES* (explained in Chapter 1) through certain endpoints. This information is also stored in the database. Subsequently, the new tweets in the database are compared with the PSAP information through a correlation algorithm, which detects whether or not a possible correlation exists.

Finally, the Tweets classified with the most critical categories, as well as all the correlations between them and PSAP information, are sent to the decision-makers, who will have more information and insight into what is happening on the field in an MCI.

Figure 3.1 shows a diagram representing the design and architecture of this system.

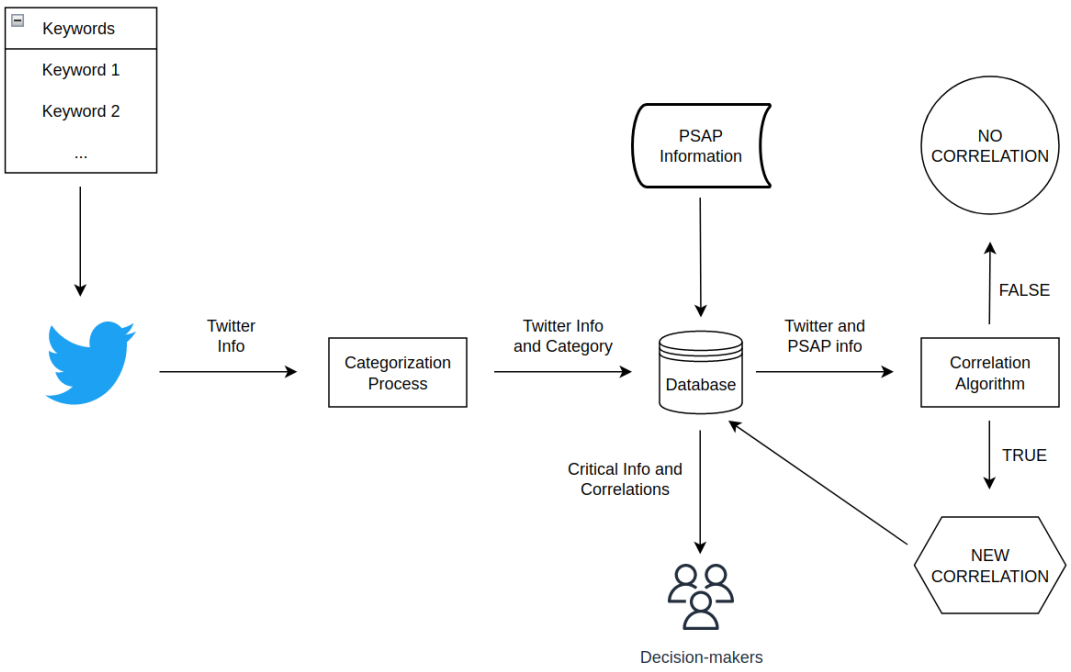

Figure 3.1: System Architecture Diagram

3.2 Work Modules and Functionalities

Considering the design and architecture of the system, the following work modules were identified: Tweet Search, Categorization Process, Database, and Correlation between PSAP and Twitter information.

3.2.1 Tweet Search

This work module is responsible for extracting tweets related to MCIs that are intended to be monitored.

Before starting to search for tweets, it is important to define the event that will be monitored, as well as its characteristics, i.e., the type of event, the city and country where it is taking place, the name assigned to the event, etc. Based on these characteristics, some keywords associated with the event are defined. General keywords are also associated with the purpose of searching for possible urgent requests from people that may be in the MCI field, e.g., 'missing', 'help', 'destruction', etc.

Some search queries are then created based on the keywords that have been defined and the name of the MCI. Subsequently, these queries are used by the API to extract the information from the tweets. The text of the extracted tweets must contain all the words included in the search queries.

Since there are very similar tweets (with the same information) made from different accounts, the tweets pass through a similarity filter. In this way, we avoid categorizing repeated information

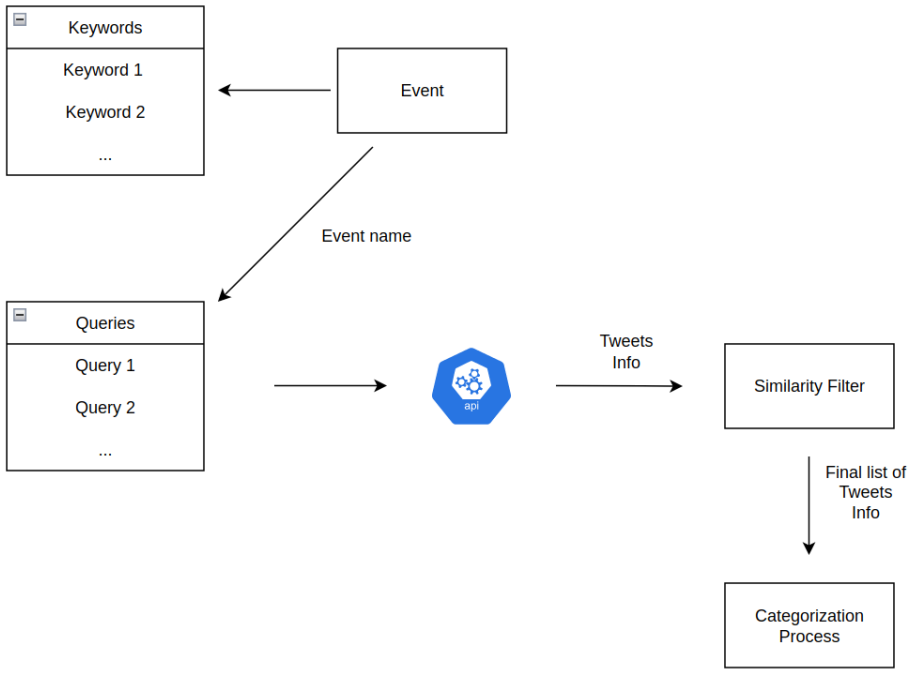

Figure 3.2: Tweet Search Diagram

in order to make the process faster and more efficient.

Figure 3.2 demonstrates the functioning of this process through a diagram. The remaining details of this process, including how the search queries are formed, the type of information that is extracted by the API, and the operation of the similarity filter are explained in section 4.1 of Chapter 4.

3.2.2 Categorization Process

The categorization process consists of defining a category for the tweets extracted through the module described in the previous sub-section, in order to determine their nature. We can thus distinguish non-urgent tweets from urgent tweets and beyond that understand the type of urgency that is reported in the tweet.

Initially, it is extremely necessary to find a dataset with tweets written during MCIs and related to them. This dataset is divided into a training dataset and a testing dataset. The training dataset corresponds to a subset of the whole dataset that is used to train the machine-learning algorithms, i.e., a set of labeled data that helps the algorithms understand how to apply their methodologies in order to predict an outcome (since it is a supervised learning problem). In this case, during the training process, besides the text of the tweets, the category to which they belong is also given as input to the algorithms (labeled data). On the other hand, testing data corresponds to a set of unlabeled data (in this context, the category of the tweet is not given as input) that is used to test

the models and evaluate their performance.

In this work module, several machine-learning models were used. After their training, they went through a test process, in which their performance was measured according to certain metrics. After the performance analysis of the algorithms and their different parameterizations, the best ML algorithm (as well as its best parameterization), is selected to address the problem of this dissertation and become part of the developed system.

Subsequently, the data that comes from the Tweet Search module is used. The information from tweets (text included) is stored in the database. Only the texts are given as input to the selected ML algorithm, which predicts the category of the tweets based on them. Each predicted category is then sent to the database associated with the tweet in concern.

All the details related to the implementation of this process are described in section 4.2 of Chapter 4. The preparation of the dataset, its transformation, the definition of the labeling scheme into which the tweets can be categorized, and other related processes are detailed in sub-section 4.2.1.

Most machine-learning models do not receive raw texts as input, so it is necessary to represent the texts in other ways (e.g. through vectors). Depending on the AI models that were tested, several data processing approaches were used for this purpose. All these approaches are fully detailed in sub-section 4.2.2. Regarding ML models, sub-section 4.2.3 details all the models used, their performances, and the selection of the best one to address the problem.

Figure 3.3 gives a flowchart illustrating the overall structure of this categorization process.

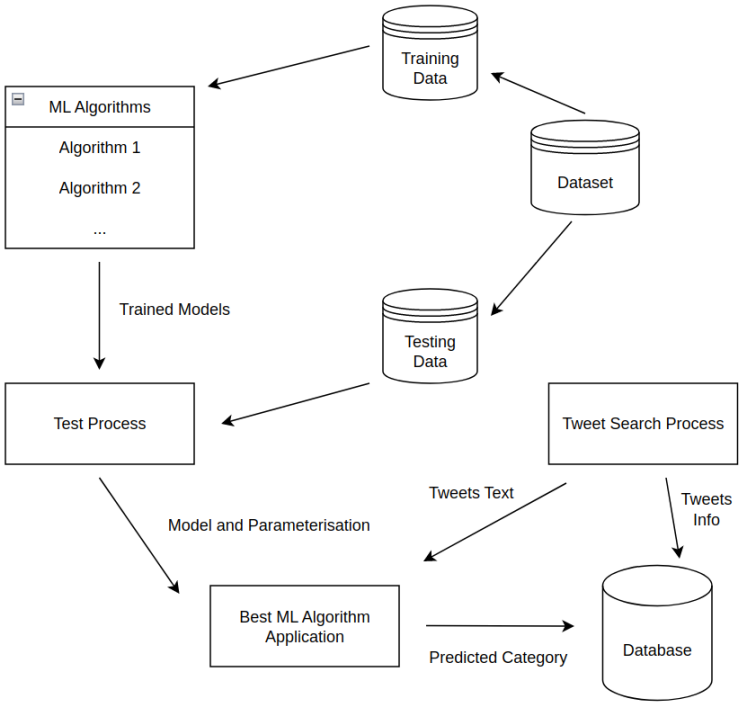

Figure 3.3: Categorization Process Diagram

3.2.3 Database

To conduct this work, it's important to store all the information used and generated in it. All the tweets extracted, their categorizations, their correlations, the MCIs monitored, the PSAP information, and other types of data must be saved. For this purpose, a relational database was used, more specifically MariaDB [24], which is a powerful and adaptable open-source relational database management system. Its support for MySQL, excellent performance, and security features make it an appealing option.

It was necessary to carry out a whole data modeling process, in which the entities that represent the main objects that interact in the system (PSAP information, Event, Category, Tweet, etc.) were defined, as well as their attributes and the relationships between them. This process is fully detailed in section 4.3 of Chapter 4.

3.2.4 Correlation between PSAP and Twitter information

Firstly, it is important to explain what we mean by correlation. The correlations we aim to find must be related to the same incidents, i.e., the same situations occurring during the same MCI. Therefore, it is important to identify if the texts are similar, whether they refer to the same temporal space, and if the same entities are involved.

This work module allows correlating the captured and categorized tweets with the information that comes from PSAP relative to emergency calls. Therefore, the information that is already known by decision-makers and other professionals who are acting on the field during an MCI can be complemented or updated with information from social media.

The main data relating to tweets and PSAP information for the operation of this process are the date and the text (text of the tweets and text of the PSAP occurrence description). This information is extracted from the database and then forwarded to the metrics that were used to recognize a possible correlation.

The three metrics used were the following: Temporal similarity, Cosine similarity, and a metric related to NER. The temporal similarity consists only of identifying the difference between the creation date of a given tweet and the occurrence date of a PSAP incident. The smaller this difference, the higher this metric score.

On the other hand, the Cosine similarity metric uses the texts for its calculation. It consists of measuring the angle between the vectors that represent the texts in a multi-dimensional space. Vectors with a lower angle between them lead to a higher score on the cosine similarity. This allows us to identify similarities between Tweets texts and the texts of the PSAP incident descriptions, which can help identify a possible correlation between them.

In addition to the temporal distance and the similarity between the text vectors, it is important to understand if there are common entities (locations, people's names, names of organizations, etc.) in both texts because it can be a very good indicator of a possible correlation. Therefore, a metric related to NER was used to identify these entities in both texts. The more common entities between the texts, the higher the score returned by this metric, however, some entities have more

value than others (which will be explained in the next chapter).

Finally, the scores of all metrics are used for the final calculation of the correlation score. If this score is higher than or equal to 0.5 it means that a new correlation has been found, otherwise, there is no correlation between the Tweet and the PSAP information.

Full details of the calculations of the scores for each metric, the calculation of the final score, and other developments related to this process can be found in section 4.4 of Chapter 4. Figure 3.4 shows a diagram demonstrating the overview of how this process works.

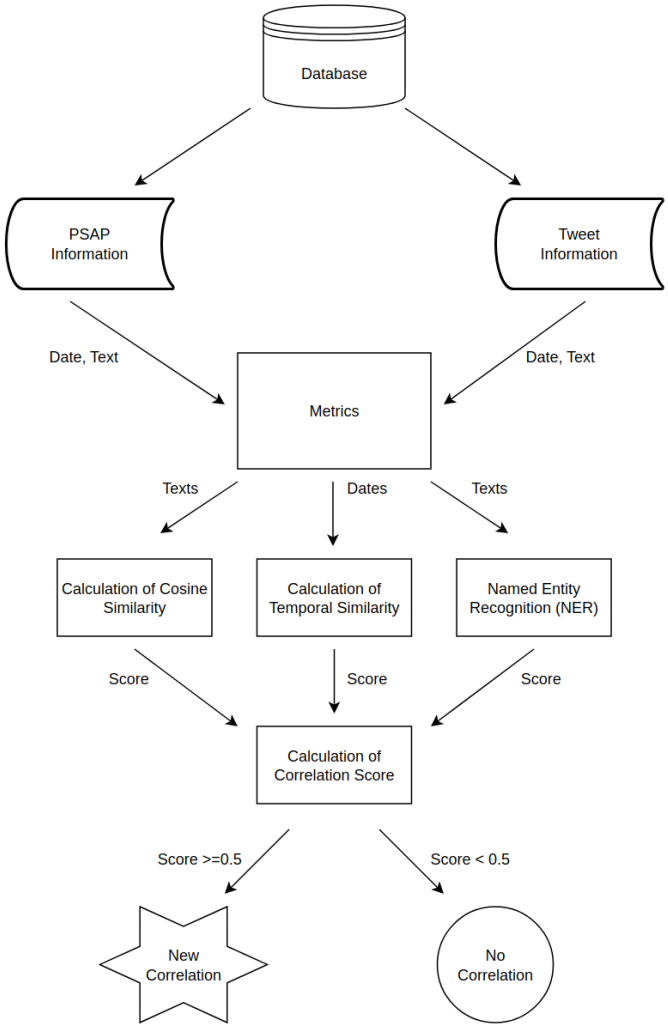

Figure 3.4: Correlation Process Diagram

Chapter 4

Module Development and Implementation

In the previous chapter, the architecture of the system was explained as well as the functioning of the modules that are part of it. In this chapter, all the details of the module implementations are explained. Section 4.1 is about the whole process of searching tweets related to the MCI events that are intended to be monitored, section 4.2 presents the full implementation of the categorization process, section 4.3 shows the developed data model and details about the relational database implementation, and section 4.4 demonstrates all the development required for the correlation between tweets and PSAP information. Finally, in section 4.5, the entire operationalization of the system was explained.

4.1 Tweet Search Process

The first step of this process involves creating a developer account on Tweepy (Twitter API), which was the method used for extracting the tweets. It was necessary to obtain a Bearer Token (Authentication Token) to access the tweets. The *client* function of the *tweepy* Python's library, which receives the obtained Bearer Token as input, was used to create the agent responsible for making requests to the API.

Then, the queries that will be used for the search are defined. Each query is given as input to the *search_recent_tweets* function of the created *client*. This function returns the information of the most recent tweets that match the specifications of the query. In addition, a maximum of 100 tweets was set as the limit for the number of tweets that can be extracted through each call of this function (the 100 most recent).

The creation of these queries involves several rules depending on the type of query that is intended to use. Table 4.1 shows some examples with their meanings. If only the keywords are specified in the query without any other flag, all tweets containing all those keywords are extracted. On the other hand, with the use of the *OR* flag, it is only necessary that the tweets have only one of those keywords to match the query specification and be extracted.

In the third example of the table 4.1 it can be seen the insertion of the *-is:retweet* flag, which

Queries Examples	Explanation
MCI storms destruction	Extract all the tweets that contain the words MCI, storms, and destruction
MCI storms OR destruction	Extract all the tweets that contain the words MCI, storms or destruction
MCI -is:retweet	Extract all the tweets that contain the word MCI and are not retweets
MCI lang:en	Extract all the tweets that contain he word MCI and are written in English
MCI has:geo	Extract all the tweets that contain the word MCI and have geo data associated
MCI has:images	Extract all the tweets that contain the word MCI and have images

Table 4.1: Examples of search queries that can be used by the *tweepy* Python's library to extract tweets and their respective meanings

ensures that retweets are not extracted. The symbol "-" has a negation meaning, so with the missing of this symbol, only the retweets would be extracted. The fourth example of the table is related to the language in which the tweets that are intended to be extracted are written. The identifier of the language in which the tweets are written comes after the *lang* statement, for instance, the flag *lang:en* causes only tweets written in English to be extracted, and the flag *lang:pt* only extracts tweets written in Portuguese.

The last examples involve the *has* statement. With the *has:images* flag, only tweets that contain images can be extracted, but with the *has:geo* flag, just tweets that have geo data associated are valid to be extracted.

In this work's process of searching for tweets, only two flags were used and they were the same for all queries:

1. *-is:retweet* - in order to avoid repeated and outdated tweets, since the retweet can be about an already extracted tweet, and also the retweet may have been made recently, but the original tweet can be outdated

2. *lang:en* - in order to obtain tweets only written in English since the dataset used to train the models contain only tweets written in English, it is important that the tweets that are supposed to be categorized also have this characteristic, otherwise there would be many classification errors.

The creation of each query used in this work always involves 3 strings as shown in Figure 4.1. The first one is related to the name of the event (MCI) that is being monitored since it is important that there are no tweets related to other events to not prejudice the analysis of the system. The second one concerns a keyword that should be useful for finding relevant tweets. Each event is associated with a set of keywords and the number of queries used in monitoring that event is equal to the number of keywords associated with it. The last string involves the already-mentioned flags that are used in all search queries in this process. All strings are separated by spaces.

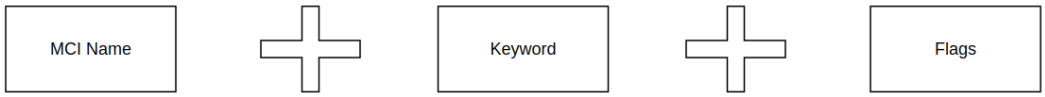

Figure 4.1: Diagram illustrating how the queries were created

In this process, each tweet's data record extracted is composed of 3 fields: 1) *Tweet_text* - corresponds to the text of the tweet in string format; 2) *Tweet_id* - corresponds to the unique identifier of a tweet in string format; 3) *Creation_date* - corresponds to the date on which the tweet was created, which is also in string format.

Algorithm 1 shows all the details of the search request of the tweets. This algorithm contains a list of keywords (associated with the event) and the name of the event that will be monitored. As previously mentioned, queries are then created (one for each keyword) and added to a list of queries. Subsequently, for each query, the *search_recent_tweets* function is invoked, which extracts the most recent tweets that match the query specifications. At the beginning of the function, a *DataFrame* with three columns that correspond to the fields of the data records of the tweets (Tweet text, Tweet ID, and Creation Date) was created. The information of the tweets that are extracted through the queries is only added to this *DataFrame* if they pass a similarity test. To develop this similarity test we used the *similarity* function of the *spaCy* python's library [46], which measures how close two texts are, semantically. This function returns a value between 0 and 1. After some tests, we defined that if a tweet has 85% or more (the value returned is equal or higher than 0.85) similarity with any of the tweets already present in the *DataFrame*, it is discarded. At the end of this process, the *DataFrame* with the information of the new tweets is returned.

4.2 Categorization Process

To start this categorization process, it is necessary to obtain and use data that can be utilized by machine-learning models for learning, so that they can make predictions about future tweets. Sub-section 4.2.1 is about the whole data preparation process, i.e., the datasets used to obtain the data, the data merging tasks, the definition of the labeling scheme (which defines in which categories the tweets will be classified), and how we dealt with unbalanced data.

Subsequently, since most ML algorithms do not receive raw texts as input, it is necessary to transform them into other types of information that are also able to represent the texts in the best way (e.g. vectors). This process is named data processing. All this process and the different techniques used in it are fully detailed in sub-section 4.2.2.

After all the data treatment, different algorithms are then tested, so that the best one can be identified and used in the developed system. All the models tested in this work and the whole

Algorithm 1 Search Tweets Algorithm

Input: $keywords$ - list of MCI event keywords , $eventName$ - name of the MCI event
Output: $dfSearch$ - list of tweet information

```
procedure SEARCHING REQUEST(keywords, eventName)
    querys ← NewList()
    dfSearch ← DataFrame(columns = [tweetText, tweetId, creationDate])
    n ← 0
    for keyword in keywords do
        newQuery ← eventName +' ' + keyword +' ' +' −is : retweet' +' lang : en'
        AddItem(querys, newQuery)
    end for
    for query in querys do
        tweets ← searchRecentTweets(query, 100)
        for tweet in tweets do
            if tweetSimilarity(tweet, dfSearch) == False then
                dfSearch[n] ← [tweet.text, tweet.id, tweet.created_at]
                n ← n + 1
            end if
        end for
    end for
    return dfSearch
```

process of identifying the best algorithm are detailed in sub-section 4.2.3.

4.2.1 Data Preparation

Datasets

Initially, only the dataset used in [29] was utilized. This dataset has 17 383 labeled disaster-related tweets and has already been explained in Chapter 2, however, it is essential to do a slightly more specific analysis. Figure 4.2 shows the number of tweets belonging to each label within the annotation scheme of that dataset. The label names are represented with tags that have the following meanings:

1. **OTHER_INFO** - Other useful information
2. **DONATE** - Donation needs or offers or volunteering services
3. **INJURED** - Injured or dead people
4. **N_REL** - Not related or irrelevant
5. **SYMPATHY** - Sympathy and emotional support
6. **DAMAGE** - Infrastructure and utilities damage
7. **ADVICE** - Caution and advice
8. **EVACUATIONS** - Displaced people and evacuations

9. **MISSING** - Missing trapped or found people

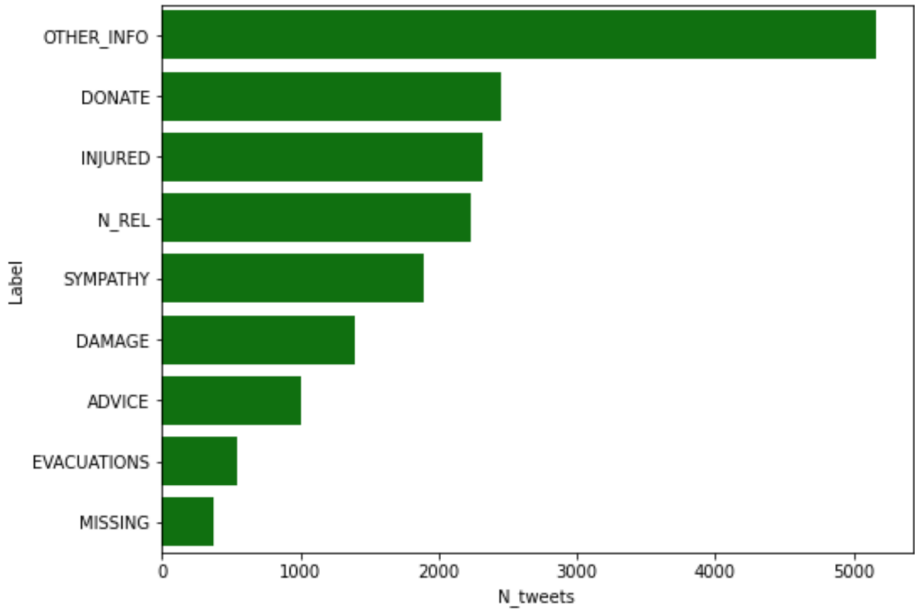

Figure 4.2: Diagram illustrating the number of tweets belonging to each label in the dataset used in [29]

The classes 'other useful information', 'not related or irrelevant', 'sympathy and emotional support', and 'donation needs or offers or volunteering services' represent 67,507% of the dataset, which indicates that a large part of the tweets may not report urgent situations to be sent to first responders. On the other hand, the class 'missing trapped or found people' has very low representation, despite being a class that is considered to be important.

After some tests, it was decided that the dataset would be increased in order to improve the results of the ML models. Therefore, the dataset used in [5], which has 36 959 labeled disaster-related tweets, was added to the existing data.

As in Figure 4.2, in Figure 4.3 the labels were also represented by tags that have the following meanings:

1. **DONATE** - Rescue volunteering or donation effort
2. **OTHER_INFO** - Other relevant information
3. **SYMPATHY** - Sympathy and Support
4. **DAMAGE** - Infrastructure and utility damage
5. **INJURED** - Injured or dead people
6. **ADVICE** - Caution and advice
7. **URG_NEEDS** - Requests or urgent needs

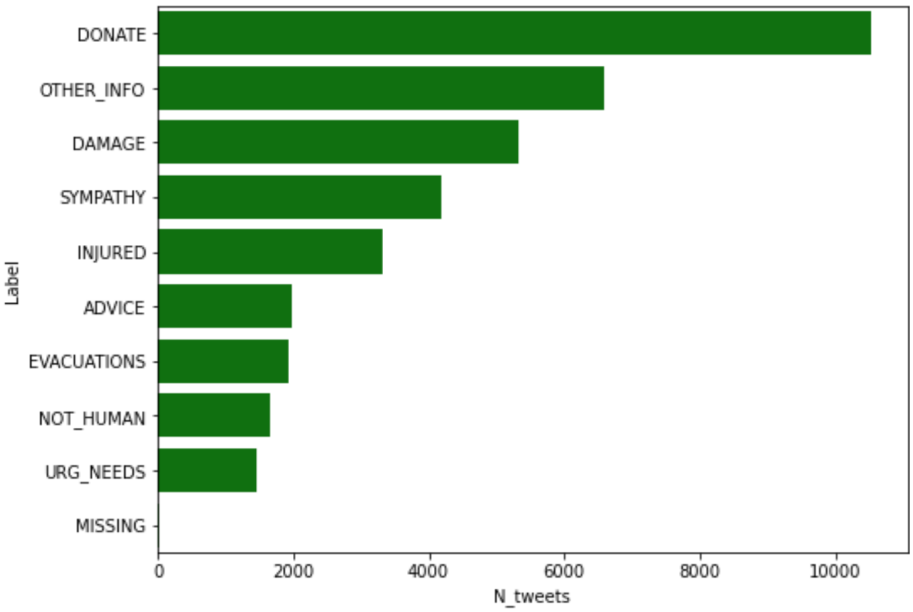

Figure 4.3: Diagram illustrating the number of tweets belonging to each label in the dataset used in [5]

8. **EVACUATIONS** - Displaced people and evacuations
9. **NOT_HUMAN** - Not humanitarian
10. **MISSING** - Missing or found people

As can be seen from Figure 4.3, the added dataset presents similar data unbalance problems to the one presented above, i.e., classes that do not represent critical requests are very representative (for instance, classes 'rescue volunteering or donation effort', 'other relevant information', 'sympathy and support', and 'not humanitarian' represent approximately 62.05 % of the data) and classes that may contain important requests have low representation, the most flagrant example, as in the first dataset, is the class 'missing or found people' which only has 17 associated tweets. The existence of few data belonging to this class may lead to ML models not having enough examples of it for training, which may subsequently cause the models not to recognize the correct label for these tweets. The manner in which we dealt with these types of problems will be explained later in this sub-section.

By joining both datasets, we obtained a total of 54 342 labeled tweets, however, the annotation schemes used by the datasets are different from each other, so it will be necessary to transform the data so that the labels used are the same.

Data Merging Process and Labeling Scheme

Considering this work's context, we conclude that the three most useful classes for decision-makers and first responders are: **1) Injured or dead people** - this class may contain information about injured people who need urgent medical assistance, so it is extremely important to send this

Twitter_as_Lifeline	HumAID	New Dataset
INJURED	INJURED	INJURED
MISSING	MISSING	MISSING
OTHER_INFO	OTHER_INFO	OTHER_INFO
DONATE	DONATE	NON URGENT
DAMAGE	DAMAGE	
ADVICE	ADVICE	
EVACUATIONS	EVACUATIONS	
SYMPATHY	SYMPATHY	
N_REL	NOT_HUMAN	
———————-	URG_NEEDS	———————-

Table 4.2: Table of correspondences between the different labels of the different datasets

type of information to first responders; **2) Missing trapped or found people** - a class that contains information that may be very useful for rescuing missing people; **3) Other useful information** - a class corresponding to other information that may be unknown and useful in order to influence certain decisions of decision-makers. Therefore, these will be the urgent labels of our labeling scheme. The other class that will also be part of the schema is called **Non-urgent** and should contain all the tweets that don't belong to any of the 3 urgent classes described above.

The annotation scheme used in [5] has one more class than the one used in [29]. This class is called 'requests or urgent needs' and it was decided that this class would not be part of our labeling scheme (all tweets related to this class were deleted) because it contains information related to requests for supplies and other resources, i.e., it is not related to medical assistance or rescues, so it does not fit the context of our problem.

Table 4.2 shows the corresponding labels of each dataset (through the same tags of Figures 4.2 and 4.3), for instance, the label 'not relevant or irrelevant' of the dataset used in [29] corresponds to the label 'not humanitarian' used in [5], which is part of the 'non-urgent' label that is used in this work's labeling scheme.

To understand the types of tweets that belong to each class of our labeling scheme, Table 4.3 shows some examples of tweets present in the new dataset and their respective class. The first one, classified as 'injured or dead people', gives important information about injured people who may be on rooftops or clinging to trees in the Buzi district of Mozambique. This information may be important for the actions of the first responders. The second one, classified as 'missing trapped or found people', reveals information that can be useful about a missing person that was in a specific city in Eastern Zimbabwe. The third one, classified as 'other useful information', shows a tweet saying that during cyclone Idai, rural areas were not being given priority, despite the focus of the cyclone passing through there a lot. This type of information can be useful for decision-makers to improve the strategies used. The last example, only reveals sympathy and support for all those affected by Cyclone Idai, so it is classified as 'non-urgent'.

After this transformation step, our dataset has a total of 52 879 labeled tweets (change resulting from deleting the tweets with the label 'requests or urgent needs'). Figure 4.4 shows the number

Examples	Label
This is the Buzi district in Mozambique after #CycloneIdai. Thousands of people have had to cling to trees and climbed onto roofs in an effort to try stay afloat. Hundreds of people are feared dead. #PrayForMozambique	INJURED
Im crying blood for my missing children. Even though #CycloneIdai hit Mozambique over a week ago, aid agencies are warning that the disaster is getting worse. @shingainyoka was in Chimanimani in Eastern Zimbabwe,one of the worst affected areas.	MISSING
@genetmalawi True though I feel like people affected by #cycloneIdai in rural areas havent been given priority coz they are also some suffering in the outskirts of #Blantyre city whose houses have fallen. I feel like much focus is only going to those rural areas	OTHER_INFO
Our hearts go out to those affected by Cyclone Idai.Now is the time for all of us to come together. Wherever you are,make an effort to do something to help those affected. #CycloneIdai	NON URGENT

Table 4.3: Examples of labeled tweets present in the new dataset

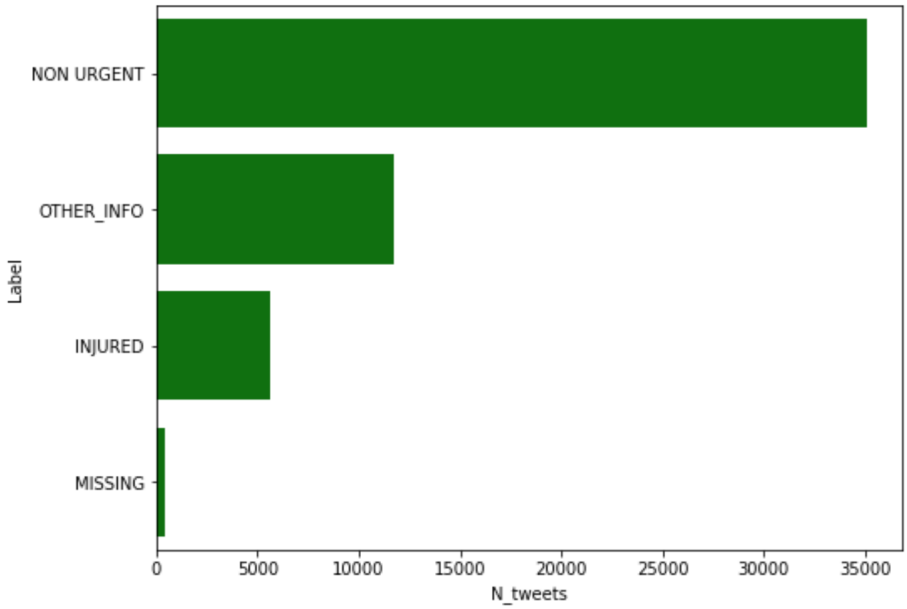

Figure 4.4: Distribution of tweets by each class in the new dataset before balancing

of tweets belonging to each label in this dataset. As we can see, the dataset is quite unbalanced, so there is a need to adopt strategies to deal with this problem.

Dealing with Unbalanced Data

ML algorithms need data to learn to categorize tweets according to their methodologies, so if there is not enough representation of a particular class in the dataset, the algorithms will not have enough data to be able to identify future tweets from that class. After some tests, which are fully detailed in section 5.1 of Chapter 5, we realized that most of the models miscategorized the tweets with the 'missing trapped or found people' label, which must have been caused by the small number of existing tweets related to this class.

The best-known ways to face a data unbalance problem are Undersampling and Oversampling. Undersampling keeps all the data from the minority class and decreases the size of the majority data, which does not seem to be the most appropriate approach to address this problem because the remaining classes would inherit the same problem as the 'missing trapped or found people' class due to the limited number of existing data for training. Oversampling is the opposite, the strategy is to keep all the data from the majority class and increase the size of the minority data, which is more reasonable given this type of problem. There are several techniques to generate new data from the minority class, such as: repetition, bootstrapping, Synthetic Minority Over-Sampling Technique (SMOTE) [10], etc.

We used Synthesized's Scientific Data Kit [20] to generate a synthetic dataset of tweets with the 'missing trapped or found people' label. This Data Kit creates a generative model to bootstrap data from any type of dataset. The Large Language Model decoder used in this process is not specified in the documentation of the Data Kit due to privacy concerns.

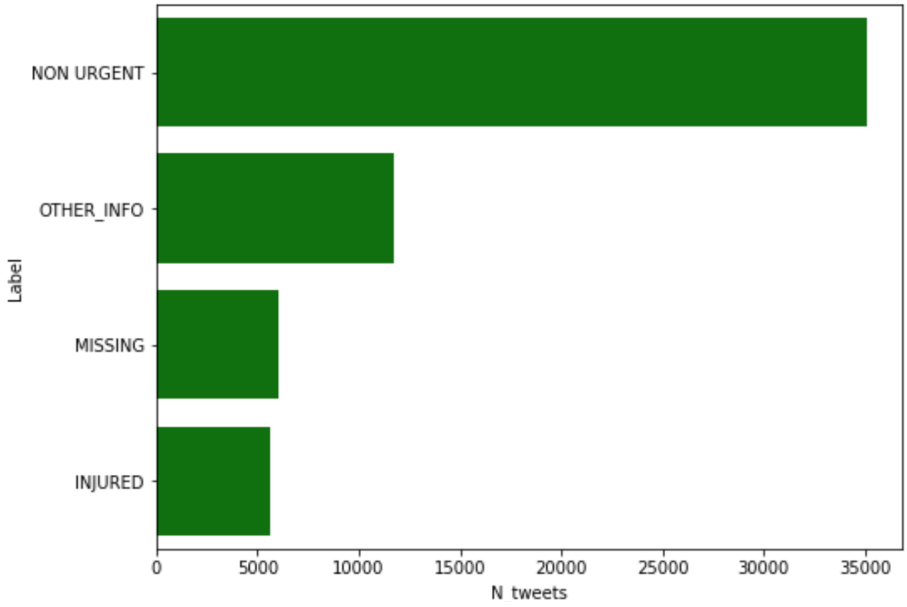

Figure 4.5: Distribution of tweets by each class in the new dataset after balancing

First, we created a data frame (from the *Pandas* library [32]) with all the 'missing trapped or found people' tweets, then we extracted all meta-information from the data frame by calling the *MetaExtractor.extract* function, which created a *DataFrameMeta* object. This meta-information is given as input to *HighDimSynthesizer* (Synthesized's Scientific Data Kit's library), which constructs the generative model. Then, we used the *synthesizer.learn()* function to learn the patterns in the data, and finally, we used the *synthesizer.synthesize()* to generate the dataset. In this final step, is necessary to specify the number of rows of the synthetic dataset, so it was decided that this number should be equal to the number of 'injured or dead people' tweets (5 629) because, after some tests, it was realized that the tweets related to this class are, generally, well classified by the ML models, which may indicate that the number of tweets is sufficient for training. It is also important that the majority of the dataset is composed of 'non-urgent' and 'other useful information' tweets, because, in reality, most of the disaster-related tweets do not contain information about 'injured or dead people' or 'missing trapped or found people', so it is important that the algorithms are able to distinguish them among the majority of the tweets.

This new synthetic dataset was merged with the existing dataset to make the final dataset. Figure 4.5 shows the new distribution of tweets by each class in this final dataset that was used to train and evaluate the ML algorithms.

In section 5.1 of Chapter 5, the demonstration of the quality of the synthetic tweets is fully detailed.

4.2.2 Data Processing

As mentioned before, data must be processed before integrating the ML models. The texts of the tweets have to be encoded, usually in the form of a numerical vector, so that models can "under-

Original Labels	Encoded Labels
INJURED	0
MISSING	1
NON URGENT	2
OTHER_INFO	3

Table 4.4: Correspondences between the original and the encoded labels with *LabelEncoder*

stand" the content of the texts. Similar vectors correspond to similar meanings. This encoding process is usually named Word Embedding. The labels in which the tweets are categorized also have to be encoded.

The encoding strategies and techniques used in this work to integrate the data in Classical ML models (non-deep learning models) were different from those used for Deep Learning models.

Classical Algorithms

Regarding non-deep learning models (classical models), the labels were transformed into numerical values, i.e., the names 'injured or dead people', 'missing trapped or found people', 'non-urgent', and 'other useful information' were encoded with values between 0 and 3. This encoding was done using the *LabelEncoder* module from *Sklearn* python's library [36].

Table 4.4 shows the correspondences between the original and the encoded labels.

After encoding the labels, it is necessary to transform the text of the tweets into numerical vectors. This step was done using *TfidVectorizer* (also from *Sklearn* library), which uses the Term Frequency-Inverse Document Frequency (TF-IDF) statistical metric. TF-IDF is a measure that compares the number of tweets a word appears in a document with the number of documents the word appears in ([14]). In the context of this problem, documents correspond to tweets.

All the words that appeared in a tweet, except the stop-words and words with low frequency (a threshold of 5 was set, i.e., if in the whole dataset, a word appeared less than 5 times, it is discarded) are represented, i.e., a tweet is represented by a vector and each position of this vector represents one of those words. For each word, a TF-IDF score is calculated and placed at its own position in the vector.

TF-IDF score corresponds to the multiplication of the Term Frequency (TF) with the Inverse Document Frequency (IDF) score. Formulas 4.1, 4.2, and 4.3 show how these scores are calculated.

$$TF(T,D) = \frac{NumberOfTimesTheTermTAppearsInD}{TotalNumberOfTermsInD} \tag{4.1}$$

$$IDF(T) = \log(\frac{TotalNumberOfTweets}{NumberOfTweetsThatContainTheTermT}) \tag{4.2}$$

$$TF-IDF(T,D) = TF(T,D) \times IDF(T) \tag{4.3}$$

id	INJURED	MISSING	OTHER_INFO	N URGENT
1	0	0	1	0
2	1	0	0	0
3	0	1	0	0
4	0	0	0	1

Table 4.5: Example of *OneHotEncoder* applied to the labeling scheme used in this work

The main advantage of this processing technique is that it contains information on the more important words and less important ones as well, however, it does not take into account the semantic meaning or context of the words. There are other types of embeddings that are common for classical algorithms, such as Word2Vec and BOW, however, TF-IDF generally performs better in these models. For instance, Denis Eka Cahyani et al. [12] realized that TF-IDF modeling had better performance than Word2Vec modeling in emotion text classification. TF-IDF also performs better than BOW in ML models, as reported in [26].

Deep Learning Algorithms

The data processing for data used by deep learning algorithms was completely different. We used *OneHotEncoder* from *Sklearn* library ([36]), which converts the categorical labels into binary features that contain the value 1 if the tweet belongs to that class and the value 0 otherwise. Table 4.5 shows how the labels of each tweet will be represented, e.g., the tweet with id = 1 is labeled as 'other useful information', so the column related to that class has the value 1 and the other columns have the value 0.

The tweet text encoding process was slightly more complex. The *Tokenizer* class from *Keras* library [15] was used to clean and transform each text into sequences of integers. First, all punctuation was removed, the OOV words were replaced by a '[UNK]' tag, and all texts were converted into lowercase. Then, through the function *fit_on_texts()*, an internal vocabulary was created based on the list of existing tweets in which each word present in these tweets has a respective integer value based on its frequency in the texts (lower values represent more frequent words). By calling the *texts_to_sequences()* function, the texts are transformed into a vector of integers, in which each position represents a different word and is filled with the word's integer value that was defined in the vocabulary created in *fit_on_texts()* function.

At this stage, the tweets were already transformed into sequences of integers, however, it is necessary that all the vectors that represent the tweets have the same size. The maximum number of words that a tweet can have is 147, so, through the *pad_sequences()* function from *Keras*, the tweets were padded with zeros until the length of 147.

Table 4.6 shows an example of applying this entire embedding process to a simple sentence.

BERT

In this work, also a BERT algorithm was used, which requires a specific type of embedding (BERT embedding). This type of embedding has shown excellent results in the research work made by

Sentence	Vocabulary	To Sequences	Padding
”The cat is prettier than the dog!”	”the” ->1 ”cat” ->2 ”is” ->3 ”prettier” ->4 ”than” ->5 ”dog” ->6	[1,2,3,4,5,1,6]	[1,2,3,4,5,6, 0,0,0,0,0,0, 0,0,0,0,0,0, 0,0,0,0,0,0, ...]

Table 4.6: Embedding process for data used by Deep Learning Algorithms applied to a simple sentence

Ashis Kumar Chanda [13], due to the use of contextual information.

Regarding the target labels, the encoding process was the same as the one used for classical algorithms, i.e., the *LabelEncoder* was used as shown in Table 4.4.

Texts needed a more complex embedding process before being used by BERT, which will categorize them. The first step is a simple *Tokenization* task, in which the texts are converted into tokens. This task was done through the function *tokenize* of the *BertTokenizer* class from *transformers* python's library ([48]). Then, it is necessary to inform the model where the sentence begins and where the sentence ends, because BERT receives a single vector representing the whole input sentence. So we added a '[CLS]' token at the beginning and a '[SEP]' token at the end of each input text. Next, it is necessary to truncate the sentences to the maximum length allowed, since the maximum number of words in a tweet is 147, we defined that the vectors would have a length of 149 (147 words + '[CLS]' and '[SEP]' tokens). To ensure that all inputs have the same size, it was necessary to pad all sentences with the token '[PAD]'.

When a BERT model is trained, a unique ID is given to each token, in order to transform all inputs into numerical vectors. Thus, when a new sentence comes into the system for classification, the *convert_tokens_to_ids* function of *BertTokenizer* class is used.

At this stage, the sentences are already transformed into numerical vectors, however, it is important to use an ”attention mask” that clarifies to the model which tokens should be attended to and which should not ('[PAD]' tokens). The ”attention mask” is a binary array of size 149 (equal to the vector representing the sentence) that is also given as input to the model, in which each position represents a certain token and contains the number 1 if that token should be attended by the model and the number 0 otherwise.

Finally, the two vectors (the one representing the sentence and the attention mask vector) are transformed into *torch* tensors (from *PyTorch* python's library [27]), which are multi-dimensional matrices containing elements of a single data type. This type of tensor has more built-in capabilities than normal *Numpy* arrays do. These capabilities (like GPU acceleration) are targeted for Deep Learning applications and are very important in transformer-based models since their training is often very slow for long sequences.

Table 4.7 shows an example of the application of this embedding process to a simple sentence.

Sentence	”This is an example of the use of Bert embedding.”
Sentence to Tokens	[’This’, ’is’, ’an’, ’example’, ’of’, ’the’, ’use’, ’of’, ’Bert’, ’embedding’, ’.’]
’[CLS]’ and ’[SEP]’ Tokens	[’[CLS]’,This’, ’is’, ’an’, ’example’, ’of’, ’the’, ’use’, ’of’, ’Bert’, ’embedding’, ’.’, ’[SEP]’]
’[PAD]’ token and Truncation Process	[’[CLS]’,This’, ’is’, ’an’, ’example’, ’of’, ’the’, ’use’, ’of’, ’Bert’, ’embedding’, ’.’, ’[SEP]’, ’[PAD]’, ’[PAD]’, ’[PAD]’]
Tokens to IDs	[101, 2421, 112, 188, 3858, 233, 3776, 106, 233, 3567,4317 ,206,102,0,0,0]
Attention Mask	[101, 2421, 112, 188, 3858, 233, 3776, 106, 233, 3567,4317 ,206,102,0,0,0] Attention Mask: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1,1,1,1,0,0,0]
Vectors to Torch Tensors	torch.tensor([101, 2421, 112, 188, 3858, 233, 3776, 106, 233, 3567,4317,206,102,0,0,0]) torch.tensor([1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0])

Table 4.7: BERT Embedding process applied to a simple sentence

4.2.3 Selection of the best Machine-Learning Algorithm (Test Process)

After all the data treatment, data can now be used by the Machine-Learning models for categorization. Several algorithms have been tested within this work and a process has been outlined to select the best algorithm to address the problem.

Algorithms tested

This work tested classical and deep learning algorithms (including BERT, which is a transformer-based model). The 9 different classical algorithms examined in this study were the following: Complement Naive Bayes, Multi Naive Bayes, Random Forest, Logistic Regression, Decision Tree, Stochastic Gradient Descendent (SGD) with *’hinge’* as loss function, SGD with *’epsilon_insensitive’* as loss function, Linear Support Vector Classifier (SVC) and K-Nearest Neighbors (KNN). To find the best hyperparameters of each model, we used the *GridSearchCV* class from the *Sklearn* library ([36]), which does an exhaustive search over specified parameterizations to find the best one.

Regarding the deep learning algorithms, the 5 models tested in this work were the following: LSTM, Gated Recurrent Units (GRU), Bidirectional Gated Recurrent Units (BI-GRU), Convolutional Neural Network Bidirectional Gated Recurrent Units (CNN-BI-GRU), and BERT.

The architectures used for the LSTM, GRU, and BI-GRU models were quite simple. All of them contained only 3 layers, with the first layer corresponding to an embedding layer which constructs an embedding matrix with the calculated weights (vector representations of each word) based on learning the vectors given as input and explained in the section 4.2.2. This embedding matrix is then given as input to the second layer, which is a model-specific layer, i.e., the LSTM

has an LSTM layer, the GRU has a GRU layer, etc. The last layer is the output layer, which receives as input the second layer, the activation function, and the number of different existing labels.

On the opposite side, the CNN-BI-GRU model's architecture is much more complex. It has two channels. The first one is composed of an embedding layer (with the same purpose as used in the other 3 models), a convolutional layer with 100 filters and a kernel size that sets the number of words to be read at once, a MaxPooling layer to consolidate the output from the convolutional layer, a Flatten layer to reduce the three-dimensional output to two dimensional for concatenation, and some dropout layers to tackle the over-fitting problem and to introduce generalization to the model. The second and more simple channel is composed of an embedding layer as well, a BI-GRU layer, a dropout layer, and the output layer.

The two activation functions tested in these 4 Deep Learning models were *relu* and *softmax*.

Regarding the BERT algorithm, we used a *transformers* pre-trained model on a large corpus of English data. This model is called *bert-base-uncased* and as the name suggests, it's an uncased model, i.e., it doesn't recognize the difference between uppercase and lowercase letters. This model was intended to jointly condition on both left and right context to pre-train deep bidirectional representations from unlabeled text. Instead of processing one word at a time, it processes the entire sequence of words at once, which helps in maintaining the context of the sentence. The *bert-base-uncased* model has 12 layers and 768 hidden nodes.

Pre-trained models must be fine-tuned in order to achieve a model with satisfactory performance and good generalization capability. In order to fine-tune this model, it was only necessary to add an output layer, however, we decided to make an architecture slightly more complex. In addition to the BERT layer, the architecture used contains 1 dropout layer, an output layer, and two activation functions (*relu* and *softmax*).

After some tests carried out to parameterize the model for fine-tuning, it was decided to use a learning rate of 0.001, 0 warm-up steps, and a batch size of 100.

Was also decided to use only 5 epochs for this fine-tuning task, in which we monitored over-fitting and saved the weights that form the best model (those that obtained the lowest *loss function* value). Finally, the BERT model is used with the weights that were saved during the fine-tuning task.

Selection process

In order to select the best possible model to address the problem, it was important to define a specific selection method. Firstly, the data was divided into a training set, a testing set, and an independent validation set (70% was used for training, 15% for validation, and 15% for testing). The training set was used to train all models (including all the parameterizations considered in the algorithms already described above). Then, the performance of each model was measured through certain statistics using the validation set. The three metrics considered to analyze the performance of the classical algorithms and the BERT model were the following:

$$Precision = \frac{TP}{TP + FP} \tag{4.4}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.5}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4.6}$$

These metrics were calculated for each of the classes of our labeling scheme in order to understand whether all types of tweets are being well classified. *TP (True Positives)* corresponds to the number of tweets well classified by the algorithm belonging to the class that is being analyzed. *FP (False Positives)* is the number of tweets that have been classified as belonging to the class that is being analyzed, but which in fact do not belong to that class (they have been misclassified). Regarding *TN (True Negatives)*, corresponds to the number of tweets that the algorithm was able to understand that did not belong to the class being analyzed. Finally, *FN (False Negatives)* is the number of tweets that actually belong to the class being analyzed but which the algorithm categorized incorrectly.

Regarding the remaining Deep Learning algorithms, the *Accuracy* was the only metric used to measure their performance:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.7}$$

This statistic consists merely of dividing the well-classified tweets by the total number of tweets. For these algorithms, the results were evaluated across all classes rather than for each class.

This distinction in how the different types of algorithms are evaluated derives from the data processing used in each of them.

Once the performance of all models had been analyzed, the one considered to have the best performance was selected. This analysis was based on a comparison of the average F1 score obtained in the classes in Classical and BERT models, and the general Accuracy obtained in the other Deep Learning models tested.

Finally, the model went through a final evaluation process with the test set, i.e., the model was tested with the data from that set to understand if the results were similar to those obtained with the validation set. After this process, the model was found to be appropriate for the problem.

The results of all the models tested as well as the algorithm chosen to address the problem, are fully detailed in section 5.2 of Chapter 5.

4.3 Data Model and Relational Database

In order to store all the information that is exchanged and generated during the operation of the system, it was necessary to create a database. As already mentioned in section 3.2.3 of Chapter 3, a relational database was used, more specifically MariaDB [24].

Building a database involves a whole data modeling process, i.e., it was necessary to create the tables that would make up the database, as well as their attributes and their relations with other tables, in order to represent the system's data and how it flows.

To monitor a specific event (MCI), it's necessary to take into account the type of event that is being monitored, i.e., whether it's a flood, an earthquake, or a gun attack, for example. So the first table to be created was the *Event Type*, which has the following attributes: *id*, *name*, *description*, and *creation date*. The *id* is only a unique numeric identifier for each type of event, and it is auto-incremented whenever a new event type is created. All the tables within our data model have this identifier as their primary key. Then, we also used the *name* and *description* of the event type as attributes, for instance, the event type with the *name* "Tornado" has the description "Violently rotating column of air that is in contact with both the surface of the Earth and a cumulonimbus cloud". The last attribute corresponds to the creation date of the event type in our database. The table 4.8 shows the detailed format of the *Event Type* table.

The *Event Type* is an important attribute of a particular event being monitored. The table *Event*, fully detailed in table 4.9, represents a specific event (MCI), for instance, Hurricane Harvey in 2017. This table contains the same numerical identifier as all the other tables (*id*), the *Event Type* id (explained above), the *name* that specifically identifies the event (e.g. "Hurricane Harvey 2017"), the *country* where the event took place, the start and the end date of the event. Each event is associated with only one event type, however, each event type can be associated with several different events.

In our data model, a given event can be associated with multiple keywords, PSAP information, and Social Media Information (tweets). The *Keyword* table represents the keywords that will be used to search for tweets about a particular event. In addition to the *id* and *event id*, this table contains the name of the keyword, its *creation date* in the database, and two boolean variables: *is active* - True if the keyword should still be used in the search, False otherwise (as events go on, sometimes certain keywords become outdated and it can be advantageous not to use them in the search); *is generic* - True if the keyword is generic and can be applied to several events, False it is a keyword specific to a certain event (e.g. "Help" is a generic keyword that can be applied to several events, however, the name of a specific street where there may be damage related to an MCI is a specific keyword). Table 4.10 shows the details and attributes of this table.

PSAP Information table represents the data that comes from PSAP concerning emergency calls. This information is sent to us in JSON format with several fields that reveal detailed information about the incident being reported. The most important fields of each incident are stored in the database. In addition to the associated *id* and *event id* (which identifies the MCI to which a specific PSAP information refers), the *PSAP Information* table is also composed of: an *external id* - string given by PSAP that uniquely identifies the PSAP incident; a *description* - a string that describes the incident; *links* - a string that contains the external ids of all other incidents related to the incident that is being reported; *entity agency* - a string that contains the name of the agency that is reporting the incident (normally for PSAP incidents, the agency is called "PEMEA-ID", however,

Event Type
id INT NOT NULL PK
name STRING
description STRING
creation date DATETIME

Table 4.8: Event Type Table

there may be other entities reporting the incident); *entity role* - a string with the department of the agency that is reporting the incident (e.g., "Police", "Fire Department", "Medical", etc.); *timestamp* - date and time when the incident was reported; *is mass* - a boolean variable that is True if the incident being reported is an MCI itself, and it is False if the incident is a specific incident that happened during an MCI (e.g., a missing child near a specific street during a hurricane); *status* - also a boolean variable that is set to True if the incident is still open and has not yet been resolved, if it has already been resolved, it is set to False. Table 4.11 shows the details of this table.

Social Media Information table represents all the tweets that are obtained through the Tweet Search Process reported in the section 4.1 of this Chapter. This table is quite simple and contains only essential information about each tweet. In addition to the associated *id* and *event id*, this table is composed of an *external id* (tweet id provided by the API), the tweet text (*text*), and the date and time of tweet creation (*creation date*). Table 4.12 shows the details of this table.

As mentioned above in this dissertation, during the categorization process, each extracted tweet is associated with one category (that is predicted by the ML algorithm that was selected to address the problem) in order to determine the nature of the tweets. The *Category* table present in our data model consists of only 3 fields: *id*, *label* (name of the category), and *description* (which describes the category). Table 4.13 shows the format of this table, which will have only 4 entries, one for each category belonging to our labeling scheme that has been explained above in section 4.2 of this Chapter.

The association of a tweet with a category is also stored in the database via the *Prediction* table. This table, fully detailed in table 4.14, in addition to the *id* contains the following attributes: *social media information id* - database id (different from the external id that is obtained through the Tweepy API) of the categorized tweet; *category id* - database id of the category in which the tweet was classified; *creation date* - date and time when the prediction was made.

Finally, in order to store the correlations between the tweets and the PSAP information, the *Correlation* table was also created. This table, detailed in table 4.15, stores every correlation between a tweet and a PSAP incident. One tweet can be correlated with several PSAP incidents, and one PSAP incident can be correlated with several tweets. Apart from the *id*, this table contains only the ids of the PSAP incident and the tweet that are part of the correlation, along with the *confidence score* that is calculated in the correlation process described in section 4.4 of this Chapter.

Figure 4.6 shows a diagram that details the entire data model of this system.

Event
id INT NOT NULL PK
event type id INT NOT NULL FK
name STRING
country STRING
start date DATETIME
end date DATETIME

Table 4.9: Event Table

Keyword
id INT NOT NULL PK
event id INT NOT NULL FK
name STRING
creation date DATETIME
is active BOOLEAN
is generic BOOLEAN

Table 4.10: Keyword Table

PSAP Information
id INT NOT NULL PK
event id INT NOT NULL FK
external id STRING NOT NULL
description STRING
links STRING
entity agency STRING
entity role STRING
timestamp DATETIME
is mass BOOLEAN
status BOOLEAN

Table 4.11: PSAP Information Table

Social Media Information
id INT NOT NULL PK
event id INT NOT NULL FK
external id STRING NOT NULL
text STRING
creation date DATETIME

Table 4.12: Social Media Information Table

Category
id INT NOT NULL PK
label STRING
description STRING

Table 4.13: Category Table

Prediction
id INT NOT NULL PK
social media information id INT NOT NULL FK
category id INT NOT NULL FK
creation date DATETIME

Table 4.14: Prediction Table

Correlation
id INT NOT NULL PK
psap id INT NOT NULL FK
social media information id INT NOT NULL FK
confidence score FLOAT NOT NULL

Table 4.15: Correlation Table

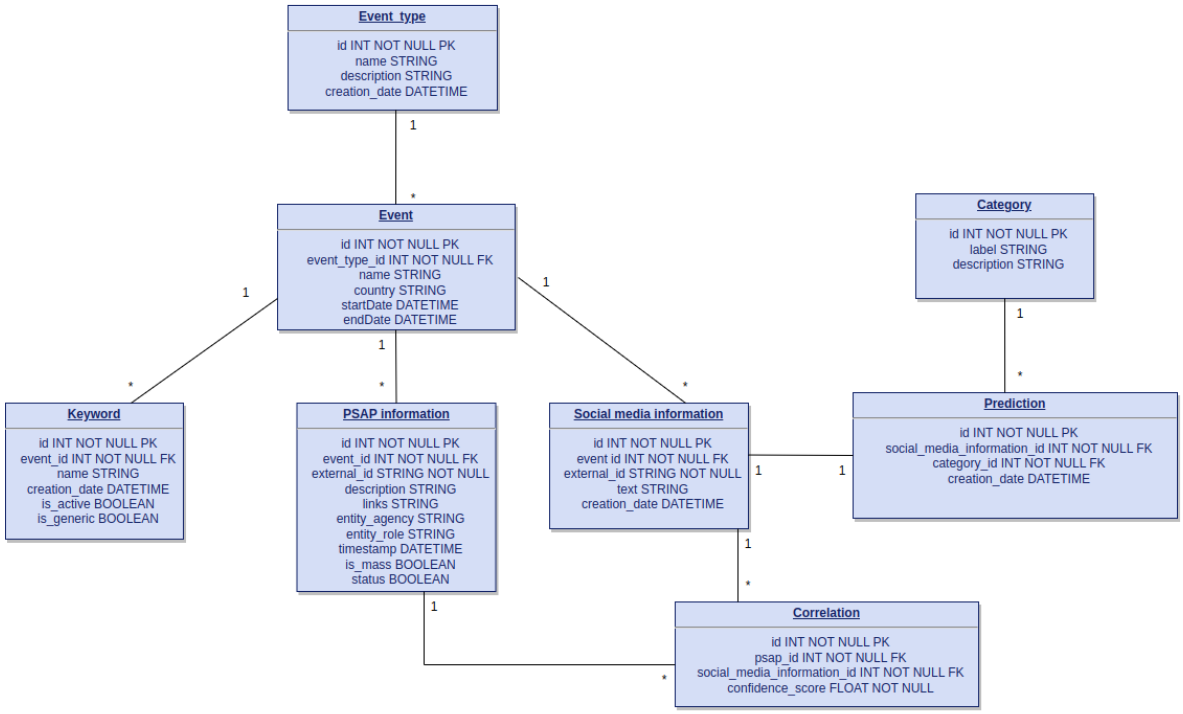

Figure 4.6: Data Model

4.4 Correlation between PSAP and Social Media information

To find links between PSAP incidents and tweets, it was necessary to develop a process for measuring the similarity between them. As mentioned above in section 3.2.4 of Chapter 3, only the texts and dates were used to measure this similarity. Regarding tweets, we used the *creation date* and *text* attributes from the *Social Media Information* table belonging to our database. When it comes to PSAP information, the *description* and the *timestamp* from *PSAP Information* table were the only data used in this process.

Additionally, it was also explained in section 3.2.4 that three types of similarity - Text similarity (Cosine similarity), Temporal similarity, and NER similarity - were calculated in order to determine the final similarity.

Text Similarity (Cosine Similarity)

This statistic was calculated with the use of the spaCy python's library [46]. First, the language model that would be used by spaCy needed to be downloaded. There were three possible English language models that differed in the amount of data: small, medium, and large. Since the medium model already has a significant amount of data for NLP jobs and the cost of processing and loading data isn't too high, we ultimately chose it. The medium model is typically the one used by developers.

This model was then used to create the *DOC* objects. In distinct interactions, the tweet text and the PSAP incident description that are involved in the similarity calculation are provided as input in string format, and a *DOC* object is formed for each text. A *DOC* object is a sequence of *Token* objects. Each *Token* object contains details on a specific textual component, usually a single word. For instance, the sentence "This is a dissertation!" is represented by a *DOC* object with the following format: ['This', 'is', 'a', 'dissertation', '!']. This type of object has a similarity function, which measures the cosine similarity between two texts. We utilized this function, which produces a value between 0 and 1 (with 0 representing a non-existent correlation and 1 representing a very strong one). This text similarity metric returns this value.

Temporal Similarity

This statistic measures the period of time that passed between the creation of a specific tweet and the creation of a PSAP incident, allowing us to understand the relevance of a possible correlation. For instance, suppose a tweet was posted a long time after a PSAP incident had been reported. In that case, the information contained therein is likely useless to the professionals on the scene (as they should already be aware of the situation), making this potential correlation of little importance.

The discrepancy value used to calculate the temporal similarity between a tweet and a PSAP incident corresponds to a value with a granularity that includes the days and minutes of difference between them. For example, if the distance between two dates is 1 day and 12 hours, then this

value will be 1.5 because 12 hours is half a day. If the distance is 1 day and 3 hours, this value will be 1.125.

Then, this discrepancy value must be used by a decreasing function so that the value returned by this statistic decreases as the discrepancy value mentioned above increases (the further apart the dates, the lower this metric value). Thus, we used the function $f(x) = (1/2)^x$ for this purpose. The value returned by this function is between 0 and 1 and is the final value returned by this statistic. To demonstrate with an example, if the temporal distance were three hours, this statistic would return 0.917, but if this distance were 17 hours, this metric would return 0.612.

NER Similarity

This statistic uses the NER model from the SpaCy Python library [46], in order to identify common entities in the texts (words that are categorized into locations, people's names, organizations, etc.). The result produced by this metric is also a value between 0 and 1, just like the other two metrics stated before.

The text of the tweet and the text of the PSAP incident description were compared, and each common entity found was awarded 10 points. However, if the entity was a location or a person's name (which we considered to be more relevant for identifying a possible correlation), 20 points were awarded in place of the 10 points. In the end, the division of these points by 100 is returned, but if there are more points than 100, the highest value for this NER similarity measure is returned instead (1).

Final Correlation Score Calculation

After computing all the similarity measurements, it was essential to determine the final confidence score for each potential correlation. Understanding the weight that each statistic should have in the final calculation was crucial. We decided that the temporal similarity metric would only represent 25% of the final similarity because it is only crucial for the usefulness of the correlation and not the correlation itself, i.e., two texts with a very large temporal distance do not cease to be correlated, but it may be too late to improve the decision-makers information. The other two measures, textual and NER similarity, which are calculated by comparing texts and can be essential for identifying links between PSAP information and tweets, make up the remaining 75% (37.5% each).

The formula for determining this final confidence score is the following:

$$Final = (Text * 0.375) + (NER * 0.375) + (Temporal * 0.25) \tag{4.8}$$

It was decided that a new correlation had been discovered if the final confidence score was greater than or equal to 0.5.

4.5 Operationalization

The system developed in this dissertation requires an operationalization step in order to be used. Certain system's jobs must run periodically, an API had to be created to share tweets and correlations with PSAP so that stakeholders and decision-makers could utilize it in the future, and the services must be deployed.

Periodic jobs

When it is decided to monitor a particular MCI, which corresponds to a certain type of event, keywords are established before the tweets are searched. This information (Event, Event Type, and Keywords) is defined once in the initial monitoring phase of each Event.

From that moment on, there were two jobs that had to be run periodically. Our database has to be updated often with new tweets that arise during the MCI and also need to be classified by the ML algorithm selected to address the problem. Therefore, the first job that needed to be run periodically was the entire process of extracting tweets (Tweet Search Process fully detailed in section 4.1 of this Chapter), classifying them (Categorization Process fully detailed in section 4.2), adding them to the database, and find correlations between them and all the PSAP information present in the database (Correlation Process fully detailed in section 4.4).

The PSAP incidents that emerged during the MCI, had also to be entered into the database. The second job involves making a call to the API provided by the Nightingale project partners, who are in charge of providing the PSAP information. Using this information, we verify if the incidents returned by the API already existed in the *PSAP Information* table of our database (this verification is done by inspecting all the external IDs of the table), and if not, we add it to the database.

To create these periodic jobs, we used *Crontab*, which is a Unix command that creates a command or list of commands that the operating system will execute one by one at predetermined intervals [43].

In this case, our cron tab command instructions specified that our jobs would run every 30 minutes.

API Creation

Since our system involves several work modules with different types of data, we developed an API that could retrieve the required data in several ways. Therefore, our API has the following endpoints:

- *GET ALL TWEETS* - returns the information concerning all tweets present in the database.
- *GET TWEET BY EVENT AND DATE (event_id, date)* - returns the information on all tweets present in the database belonging to the event whose ID is specified in the input and created from the date specified in the input as well (the date can be precise down to the minute and second, but it can also merely contain the day and hours).

<table>
<tr><th>Related Incident</th><th>Tweet List Item</th><th>FINAL JSON FORMAT</th></tr>
<tr><td rowspan="2">Existing: boolean</td><td>Tweet ID: string</td><td rowspan="3">List Tweets: [TweetListItem]</td></tr>
<tr><td>Timestamp: string</td></tr>
<tr><td rowspan="2">Confidence Score: float</td><td>Text: string</td></tr>
<tr><td>Predicted Category ID: integer</td><td rowspan="3">Last Tweet Timestap: string</td></tr>
<tr><td rowspan="2">Incident ID : string</td><td>Predicted Category: string</td></tr>
<tr><td>Incident: Related Incident</td></tr>
</table>

Table 4.16: Data used in each JSON file that is returned in our API

- *GET TWEET BY PSAP INCIDENT (psap_id)* - returns the information of all tweets present in the database correlated with the PSAP incident whose ID is specified in the input.

- *GET INJURED OR DEAD PEOPLE WORD CLOUD (event_id)* - returns an image with the most used words in the tweets present in the database classified as ”injured or dead people” of the event whose ID is specified in the input. A word occupies more space in an image the more times it is used in the tweets.

- *GET MISSING TRAPPED OR FOUND PEOPLE WORD CLOUD (event_id)* - returns an image with the most used words in the tweets present in the database classified as ”missing trapped or found people” of the event whose ID is specified in the input.

- *GET OTHER USEFUL INFORMATION WORD CLOUD (event_id)* - returns an image with the most used words in the tweets present in the database classified as ”other useful information” of the event whose ID is specified in the input.

This API’s endpoints are all GET requests. The word clouds are made available in image format and aim to understand the most used words in tweets classified in classes considered non-urgent, in order to update the keywords used for searching the tweets. These images have the particularity of representing the words with a size proportional to the number of times they are used.

The remaining endpoints return information in JSON format. Table 4.16 shows the format of each JSON file returned. Each JSON consists of a list of tweets and the timestamp of the most recent tweet in the list. Each tweet’s information consists of its ID, creation date (in string format), text, the ID, and name of the category in which it was categorized, and information about the incident correlated with the tweet that obtained the highest confidence score. The incident ID, the confidence score of the correlation between the incident and the tweet, and a Boolean variable that indicates whether the incident exists or not (if the tweet has not been correlated with any incidents, the variable is set to False) are all included in each information relating to the incident.

Listing 1 shows a small example of the information returned by a GET request to one of our endpoints that returns information in JSON format.

Listing 1 Example of a result after a GET request to one of our endpoints that returns information in JSON format


```
{
{
  "TweetList": [
    {
      "TweetId": "1653579814849093632",
      "Timestamp": "2023-05-03 01:58:44",
      "Text": "@MalBraveheart Not in Australia,
      but Ohio in the US. We had a swarm of tornadoes
      on a single night.  The Red Coss did absolutely
      nothing.Local businesses set up donation centers
      and Samaritan's Purse sent people in to clean up.
      They stayed until every last homeowner got help
      with clean up.&gt;&gt;",
      "PredictedCategoryId": 3,
      "PredictedCategory": "non-urgent",
      "RelatedIncident": {
        "Existing": false,
        "ConfidenceScore": 0,
        "IncidentId": "0"
      }
    },
    {
      "TweetId": "1653521588178886658",
      "Timestamp": "2023-05-02 22:07:22",
      "Text": "@isaiahrmartin @stablegeniusinc Can you help
      us in Oklahoma? It’s a disaster &amp; not fm
      tornadoes! Superintendent &amp;
      Governor! Send HELP",
      "PredictedCategoryId": 3,
      "PredictedCategory": "non-urgent",
      "RelatedIncident": {
        "Existing": false,
        "ConfidenceScore": 0,
        "IncidentId": "0"
      }
    }
    ],
  "LastTweetTimestamp": "2023-08-30 10:17:57"
}
```

Service Deployment

To deploy the system developed in this dissertation, we used *Docker*. We stored different types of information in containers, which consist of software images that have everything needed to run a particular service. In this way, the services can run in parallel, execute only the part they are responsible for, and communicate with the other containers.

The system has been divided into the following containers:

- *DATABASE CONTAINER* - covers all components required to create and support the database.
- *BACKEND CONTAINER* - responsible for deploying the API
- *PERIODIC CONTAINER* - responsible for executing and ensuring the software required for the aforementioned periodic jobs
- *ADMINER CONTAINER* - provides a framework that makes it possible to access the database's contents in a very simple way, which makes it easier to control and monitor the database.

Chapter 5

Evaluation and Demonstration

In this chapter, we intend to demonstrate the functionalities and results of the system developed. In section 5.1, we demonstrate how the data balancing process described in 4.2.1 was crucial for the outcomes, and we also show the caliber of the synthetic data generated following this balancing process. In section 5.2, we show the results of all the models tested for the categorization process, as well as reveal which model was chosen to address the problem. Finally, we demonstrate in detail the monitoring of a particular event using our system.

5.1 Data balancing

As mentioned above in section 4.2.1 of Chapter 4, a procedure of data balancing has to be performed, in order to ensure better results in the categorization of ”missing trapped or found people” tweets. This section aims to show how this class results improved after the addition of the synthetic data and then provide some examples to highlight the quality of this data.

Table 5.1 shows the results of the SVC linear model before the use of the synthetic data.

As we can see, the results for the ”missing trapped or found people” class are much poorer than the results for the other classes, although the ”other useful information” class does not show excellent results either. The *Precision* value is not extremely bad because the amount of false positives is small, i.e., there are few tweets that do not belong to that class being classified as ”missing trapped or found people”, however, the recall value is very low due to a large number of false negatives, i.e., there are a lot of tweets that belong to the ”missing trapped or found people” class being classified as other classes. Models can’t ”understand” when it is a tweet from the ”missing trapped or found people” class, because they are failing to find patterns in the data

	Precision	Recall	F1-Score	Total Support
INJURED	0.91	0.91	0.91	5 629
MISSING	**0.77**	**0.25**	**0.38**	**392**
NON URGENT	0.84	0.88	0.86	35 093
OTHER INFO	0.60	0.53	0.56	11 765

Table 5.1: Linear SVC’s performance before the use of synthetic data

	Precision	Recall	F1-Score	Total Support
INJURED	0.90	0.92	0.91	5 629
MISSING	**0.98**	**1.0**	**0.99**	**6 021**
NON URGENT	0.84	0.89	0.86	35 093
OTHER INFO	0.63	0.50	0.56	11 765

Table 5.2: Linear SVC's performance after the use of synthetic data

RT @ShakingEarth: #Earthquake #Chile - 10 to 15 children missing - 80% of the fishing boats in La Caleta damaged due to the tsunami
@USAID This is a database with the location and needs of people trapped in Nepal. http://t.co/SEG4iu67gs Many are in need of evac & supplies
Nepal earthquake: Nine New Zealanders still unaccounted for http://t.co/W44ZgV4oAn via @nzherald

Table 5.3: Examples of synthetic 'missing trapped or found people' tweets generated through the Synthesized's Scientific Data Kit [20]

that distinguish these types of tweets from others (training mistakes that prevent the model from generalizing effectively to data). This phenomenon is called *underfitting* and it is clearly caused by the limited data available for training.

After the procedure of adding the synthetic data to the dataset, the results of all the models significantly improved. The results of the same SVC model (whose results obtained before this process are illustrated in table 5.1) after this procedure are illustrated in table 5.2. As we can see, the "missing trapped or found people" class moved from having the lowest to having the best categorization performance.

Analyzing the quality of the synthetic data is just as crucial as determining if the new data has contributed to better results. After an analysis, we realized that the generated data was of the same type as the "missing trapped or found people" tweets that already existed. We also realized that the texts had the same structure as texts written on social media (use of #'s, @'s, etc.) and that there were differences in the construction of sentences from text to text (which may be important for models to be able to detect different types of "missing trapped or found people" tweets written in different ways).

Table 5.3 shows some examples of these synthetic "missing trapped or found people" tweets.

5.2 Machine-Learning Algorithms Performance and Selection of the best model

This section starts by showing the testing set results for the top parameterizations for each model tested in this dissertation. We evaluated several parameterizations of the various algorithms and

we were able to identify those that obtained the best results for each model.

A given model is considered to have a good performance when it performs well across all of the evaluation metrics used in the model evaluation process, i.e., if a model performs exceptionally well in one evaluation measure but horribly in another, the model is considered to have poor performance.

After the results, we demonstrate which model was chosen and why based on them. Finally, we show the results of the selected model with the validation set.

5.2.1 Classical Algorithms Results

Table 5.4 shows the performances of the classical models in the testing set.

As we can see, only a few models are still unable to accurately categorize "missing trapped or found people" tweets. It should be emphasized, nonetheless, that the classification of this class remains extremely challenging for the Logistic Regression, SGD Epsilon, and SGD Hinge models.

It also becomes clear that all the models had difficulty categorizing "other useful information" tweets, and this ended up being the class with the worst results overall. On the other hand, the "non-urgent" and "injured or dead people" classes were the ones that produced the greatest results, maybe as a consequence of the special characteristics of the texts in these categories that make it easier for the models to recognize them.

Complement Naive Bayes, Multi Naive Bayes, and Linear SVC were the only classical algorithms to achieve consistent results across all measures for all classes. The latter model distinguished itself from the others with an average above 82% across all metrics, proving to be an excellent candidate for addressing the problem.

5.2.2 Deep-Learning Algorithms Results

Table 5.5 shows the performance of the deep-learning models tested in this work in the testing set.

The results of these models did not change much with the variation of their hyperparameters. CNN-BI-GRU and BI-GRU models obtained good performances (the first one obtained an Accuracy of 81%, and the latter had an Accuracy rate of 80%), unlike the LSTM and GRU which had disastrous performances (both showed an Accuracy rate of 59%).

Since these algorithms are more complex and require extensive computing needs, and do not obtain better results than certain classical models, we decided that were not the right models to address the problem.

5.2.3 BERT Results

Table 5.6 shows the performance of the BERT model implemented in this work in the testing set.

The results demonstrated that this model outperformed all the other models examined throughout the selection process. This algorithm was the best at categorizing the "injured or dead people" class, obtaining a Precision of 91%, a Recall of 96%, and an F1-score of 93%. These metrics values are slightly higher than those obtained by Linear SVC, which obtained the best results among

Algorithms	Labels	Precision	Recall	F1
Decision Tree	INJURED	0.87	0.87	0.87
	MISSING	0.96	1.0	0.98
	NON URGENT	0.82	0.82	0.82
	OTHER_INFO	0.51	0.50	0.51
	AVERAGE	**0.79**	**0.80**	**0.80**
KNN	INJURED	0.91	0.57	0.70
	MISSING	0.98	0.99	0.99
	NON URGENT	0.73	0.98	0.84
	OTHER_INFO	0.71	0.10	0.17
	AVERAGE	**0.83**	**0.66**	**0.67**
Logistic Regression	INJURED	0.87	0.84	0.86
	MISSING	0.81	0.53	0.64
	NON URGENT	0.72	0.95	0.82
	OTHER_INFO	0.63	0.17	0.27
	AVERAGE	**0.76**	**0.62**	**0.65**
Complement Naive Bayes	INJURED	0.75	0.89	0.81
	MISSING	0.82	1.0	0.90
	NON URGENT	0.83	0.86	0.84
	OTHER_INFO	0.60	0.41	0.49
	AVERAGE	**0.75**	**0.79**	**0.76**
Multi Naive Bayes	INJURED	0.85	0.83	0.84
	MISSING	0.94	1.0	0.97
	NON URGENT	0.81	0.91	0.85
	OTHER_INFO	0.63	0.39	0.48
	AVERAGE	**0.81**	**0.78**	**0.79**
Random Forest	INJURED	0.89	0.84	0.87
	MISSING	0.97	0.84	0.90
	NON URGENT	0.74	0.97	0.84
	OTHER_INFO	0.73	0.11	0.19
	AVERAGE	**0.83**	**0.69**	**0.7**
SGD Epsilon	INJURED	0.92	0.66	0.76
	MISSING	0.89	0.59	0.71
	NON URGENT	0.69	0.99	0.81
	OTHER_INFO	0.79	0.04	0.07
	AVERAGE	**0.82**	**0.57**	**0.59**
SGD Hinge	INJURED	0.91	0.68	0.78
	MISSING	0.84	0.63	0.72
	NON URGENT	0.70	0.98	0.82
	OTHER_INFO	0.80	0.04	0.08
	AVERAGE	**0.81**	**0.58**	**0.6**
Linear SVC	INJURED	0.90	0.92	0.91
	MISSING	0.98	1.0	0.99
	NON URGENT	0.84	0.89	0.86
	OTHER_INFO	0.63	0.50	0.56
	AVERAGE	**0.84**	**0.83**	**0.83**

Table 5.4: Results of the best parameter settings for each classical ML model in the testing set

Algorithms	Activation Function	Accuracy
CNN-BI-GRU	Relu	0.81
BI-GRU	Softmax	0.80
GRU	Relu/Softmax (Same results)	0.59
LSTM	Relu/Softmax (Same results)	0.59

Table 5.5: Performances of the best parameter settings for each deep learning algorithm in the testing set

	Precision	Recall	F1-Score
INJURED	0.91	0.96	0.93
MISSING	0.96	1.0	0.98
NON URGENT	0.89	0.87	0.88
OTHER INFO	0.66	0.67	0.66
AVERAGE	**0.85**	**0.88**	**0.86**

Table 5.6: Results of the BERT algorithm in the testing set

the classical algorithms, with a Precision of 90%, a Recall of 92%, and an F1-score of 91% in that class.

Although the ”missing trapped or found people” class was also reasonably well categorized, this finding didn't surprise us because, after balancing the data, the algorithms that showed positively consistent results also performed perfectly when it came to categorizing this class.

Regarding the ”non-urgent” class, this model performed similarly to Linear SVC, Complement Naive Bayes, and Multi Naive Bayes, although slightly better, showing excellent results.

This algorithm's ability to classify ”other useful information” tweets was one of the aspects that made it stand out from the others. As mentioned above, this class was the one that the algorithms had the most difficulties detecting. For instance, Linear SVC, which had the best performance in this class among the classical algorithms, only had 63% Precision, 50% Recall, and 56% F1-score. BERT obtained 63% Precision, 66% Recall, and 65% F1-score, which shows a reasonable difference in performance to Linear SVC, and even greater for the other algorithms.

BERT showed excellent results, which may be due to the fact that it takes into account the semantics and context of the words in the text.

5.2.4 Selection of the best algorithm and Validation Process

The model chosen to address this problem was BERT. As shown above, this algorithm obtained the best performance in the evaluation with the testing set and contains characteristics (perception of the semantics and context of the words in the sentences) that could be very important for this work, and set it apart from the other algorithms.

As a consequence, this model progressed to the validation stage, where it was evaluated using the validation set and the same metrics as previously. Table 5.7 shows the results with the valida-

	Precision	Recall	F1-Score
INJURED	0.89	0.95	0.93
MISSING	0.97	1.0	0.99
NON URGENT	0.88	0.88	0.86
OTHER INFO	0.63	0.66	0.65
AVERAGE	**0.84**	**0.87**	**0.86**

Table 5.7: Results of the BERT algorithm in the validation set

Event Type ID	Name	Description	Creation Date
1	Tornado	Violently rotating column of air that is in contact with both the surface of the Earth and a cumulonimbus cloud	2023-03-18 10:34:09

Table 5.8: Information about the *Tornado* event type included in the system

tion set. This table reveals that there was no significant difference in the results compared to those obtained with the testing set, so the model was validated and considered appropriate for use in the system.

5.3 Monitoring a specific event

This section outlines all the steps used to monitor a specific MCI, in order to highlight how this service operates as a whole. We used the tornadoes that hit the United States in March 2023 as an example. Firstly, the event type *Tornado* was created and added to our database along with its name, description, and creation date. The properties of this newly constructed event type are shown in table 5.8.

Then, we created the *US Tornadoes* event so that all the information resulting from the operation of this system is associated with this event. This event was then added to the database along with the name of the country where the event is taking place, the event's start date, and its end date (we entered a far-off date because, when an event is happening, it's impossible to know when it will end; when the event is over we change the date, which will only be used for historical data). The properties of this event are shown in table 5.9.

Afterward, it was necessary to define the keywords used in the search. Table 5.10 shows the properties of the keywords used in this example. Using these keywords, the search queries, which are fully described in table 5.11, were generated.

Using Tweepy API, tweets related to the search queries are extracted, then categorized by the Machine-Learning model chosen to address the problem (BERT model), and then the information from the tweets and their categorization is stored in the database. 35 tweets were retrieved in the

Event ID	Name	Start Date	End Date
1	US Tornadoes	2023-03-18 10:54:36	2024-03-18 10:54:36

Table 5.9: Information about the *US Tornadoes 2023* event included in the system

Keyword ID	Name	Creation Date	Is Active	Is Generic
1	Help	2023-03-18 10:57:42	True	False
2	Missing	2023-03-18 10:57:42	True	False
3	Injured	2023-03-18 10:57:42	True	False
4	Destruction	2023-03-18 10:57:42	True	False
5	MCI	2023-03-18 10:57:42	True	True
6	Wind	2023-03-18 10:57:42	True	False

Table 5.10: Information about the keywords used in the search

Tags
”US Tornadoes Help -is:retweet lang:en”
”US Tornadoes Missing -is:retweet lang:en”
”US Tornadoes Injured -is:retweet lang:en”
”US Tornadoes Destruction -is:retweet lang:en”
”US Tornadoes MCI -is:retweet lang:en”
”US Tornadoes Wind -is:retweet lang:en”

Table 5.11: Tags used in the search

first instance of this search. 51.4% (18) were classified as non-urgent, 25.7% (9) as ”other useful information”, and 22.9% (8) as ”injured or dead people”. Since ”missing trapped or found people” is a category with difficult-to-find qualities, no tweets were categorized under it.

Three samples of tweets categorized as ”non-urgent” are shown in table 5.12. As we can see, these merely reveal information about donations, encouraging remarks, or requests for the tornadoes to stop. Regarding ”injured or dead people” tweets, some examples are shown in table 5.13. These tweets reveal information about people who are injured or have died due to the tornadoes, so it can be considered that these tweets have been well categorized.

Finally, the examples of tweets classified as ”other useful information”, which are illustrated in table 5.14, show information about policies that lead to certain problems, wind statistics during tornadoes, or discussion by people about measures to mitigate the impact of tornadoes and other natural disasters. This information can also be useful in helping to improve the way these events are dealt with.

She’s provided support during natural disasters, donating $1 million to the victims of Louisiana floods and $500,000 to the Nashville flood relief, and raising $750,000 through a Speak Now Help Now benefit concert for victims of tornadoes in the southern US in 2011. https://t.co/wBUyEJruld
@IntelPointAlert @kywxgal @kywxgal Oh my goodness! Vanessa - wow! Does it feel like it’s every day sometimes? With earthquakes and tornadoes? Don’t know what to say . . . God keep us safe in Jesus’ Name. Help the victims.
@EdKrassen Just watch us come together and help each other after disasters such as our infamous tornadoes level communities.

Table 5.12: Examples of not urgent extracted tweets related to the *US Tornadoes*

At least 23 people have been killed in Mississippi after tornadoes and severe weather swept through the state. State officials say at least 11 fatalities were caused by a tornado, while 12 people died from severe weather conditions, including wind \https://t.co/PZ4fPHv1wj
7 injured in Texas tornadoes, storms; blizzard warnings issued in 6 states as massive winter storm rocks US https://t.co/8BHkWrLVtV via @usatoday
outbreak of 79 tornadoes struck the Central and Southern US. 20 tornadoes were significant . Oklahoma, and areas around Racine & Neosho, Missouri. 24 people were killed and 408 were injured.

Table 5.13: Examples of injured or dead people extracted tweets related to the *US Tornadoes*

This happens all the time.I remember R's voting " no" on giving more funds to Jersey,New York after Hurricane Sandy because socialism. Until less than a month,their states wracked by Tornadoes. Then Oh help us Federal government.we like socialism. https://t.co/I8cDd84d8v"
#HeyTexas our @NWSSouthern partners have one #Hail of a forecast for us! #GIANT Hail (2-3/4" or larger), +75 mph wind gust & 10% (or greater) chance of EF-2 (or greater) #tornadoes from ∼2:00pm today thru tomorrow morning #txwx #AreYouReady https://t.co/GZfq5bTCj
Join us tomorrow from 2-3:30PM at Blue Ash Public Library (4911 Cooper Rd) to review our Hazard Mitigation Plan & provide input on mitigation measures that will help reduce the impact from hazards like flooding & tornadoes/wind. Info & other meeting times: https://t.co/TRE4StQP02 https://t.co/Q8Oxl8nri0

Table 5.14: Examples of other useful information extracted tweets to the *US Tornadoes*

Text Incident PSAP: "Tornadoes killed people in Tennessee and there are many structures damaged across the state"	**Temporal Distance**	**Textual Similarity**	**Temporal Similarity**	**Final Score**
Tweet: "Tornadoes are killing people, i'm so afraid #tornadoes"	3 hours (0.125 days)	0.15	0.91	0.33
Tweet: "Tornadoes in Tennessee, everything is damaged !!!"	3 hours (0.125 days)	0.57	0.91	0.65
Tweet: "@david67 more than 500 structures damaged in Tennessee"	3 hours (0.125 days)	0.62	0.91	0.69
Tweet: "Tornadoes are killing people, i'm so afraid #tornadoes"	9 hours (0.375 days)	0.15	0.77	0.29
Tweet: "Tornadoes in Tennessee, everything is damaged !!!"	9 hours (0.375 days)	0.57	0.77	0.61
Tweet: "@david67 more than 500 structures damaged in Tennessee	9 hours (0.375 days)	0.62	0.77	0.65
Tweet: "Tornadoes are killing people, i'm so afraid #tornadoes"	1 day and 3 hours (1.125 days)	0.15	0.45	0.22
Tweet: "Tornadoes in Tennessee, everything is damaged !!!"	1 day and 3 hours (1.125 days)	0.57	0.45	0.54
Tweet: "@david67 more than 500 structures damaged in Tennessee	1 day and 3 hours (1.125 days)	0.62	0.45	0.58

Table 5.15: Examples of correlation score results with unsourced texts and dates

After all tweets and categorizations are stored in the database, the PSAP information is retrieved through the endpoint provided by the project partners, and the incidents are stored in the DB. Since we don't have information about the PSAP incidents from the United States, some examples of correlation score results returned by our correlation algorithm with some texts and dates assumed by us are shown in table 5.15.

For these instances, we considered the following sentence as the description of a particular PSAP incident: *"Tornadoes killed people in Tennessee and there are many structures damaged across the state"*. The first example tweet used had the following text: *"Tornadoes are killing people, i'm so afraid #tornadoes*. Since it is a very general text and does not have any common entity with the description of the PSAP incident, the Textual Similarity value (Cosine Similarity + NER Similarity) was very small (0.15). This text lacks the names of any cities, streets, or countries that might establish the location of the tornadoes. The final correlation score's value then changes

depending on the temporal distance between the creation date of both texts. If the two are separated by 3 hours (corresponding to a Temporal Similarity value of 0.91, which is a fairly short time gap), the final value of the correlation score is 0.33, which is not enough to establish a correlation. This value decreases to 0.29 with 9 hours of temporal distance and to 0.22 with 1 day and 3 hours of separation (resulting in a very weak correlation score value).

The second example tweet involves the following text: *"Tornadoes in Tennessee, everything is damaged !!!"*. Due to the fact that both texts are about tornado damage and have Tennessee as a common localization entity, this text was given a Textual Similarity score of 0.57. With the common location explicit in both texts, it is clear that they are both related to the same event. The value of the final score changes with the temporal distance, but this text always obtained confidence scores greater than 0.5 with the temporal distances used as examples, which indicates that even if the date of creation of the two texts were separated by 1 day and 3 hours, a correlation between the PSAP incident and the tweet would still be recorded.

The third example tweet involves the following text: *"@david67 more than 500 structures damaged in Tennessee"*. The greatest textual similarity value was found in this example. The text expressly mentions damaged structures, as the text of the PSAP incident description used as example, in addition to having the Tennessee localization entity in common. This text obtained a Textual Similarity of 0.62. The final correlation score between the two texts was 0.69 for a temporal distance of 3 hours, 0.65 for a difference of 9 hours, and 0.58 if they were separated by 1 day and 3 hours.

After this correlation process, the API is deployed (*backend container*), and the entire process of searching for tweets, categorizing them, correlating them, and storing the information in the database is repeated every 30 minutes (through the *periodic container*).

Chapter 6

Conclusion

We start off this final chapter by summarizing the conclusions drawn from the system developed under the scope of this dissertation. Then we explain the limitations of this work, and at the end, we explain what could be improved in this work and leave some recommendations for future research on the subject.

6.1 Conclusions

In this work, we were able to develop a complete system for monitoring social networks during MCIs. This system begins with a data extraction process from social networks, in which we used Tweepy API to extract data from Twitter. However, this system can be adapted to extract data from other social networks, simply by using an API or other extraction source related to the social network intended to be monitored.

Then, a whole process of text categorization, which is the most important component of the system developed in this dissertation, was created. This process starts by combining two different datasets of disaster-related tweets, defining a labeling scheme in which the tweets could be classified, and mapping the categories of the datasets used to our own categories. This data preparation phase allowed us to approach the problem in the way that we considered most suitable to our purpose and also made it possible to use data with the quantity and quality needed for machine-learning models.

Different NLP techniques were then used to process the texts, several algorithms were evaluated and a process was devised to select the best algorithm to address the problem. The BERT model was chosen to be included in the system after analyzing the results shown in section 5.2 of Chapter 5.

Additionally, an algorithm that can identify correlations between social media texts and texts of information from emergency call centers (public safety answering points - PSAP) has been developed.

Finally, all the tasks necessary to operationalize this system have been carried out, allowing for the optimum utilization of it during an MCI.

The system developed fully answers the research questions mentioned in section 1.2 of Chapter

1. Regarding the first research question (*Can relevant information really be detected amidst a lot of non-urgent information?*), approximately 65% of tweets in the dataset used to train, test and validate the machine-learning models are considered "non-urgent", but BERT still achieved results above 90% in all evaluation metrics in the testing set when categorizing tweets belonging to the "missing trapped or found people" and "injured or dead people" classes. Although categorizing the tweets from the "other useful information" class was more challenging, BERT was still able to achieve scores over 65% in all evaluation metrics in the testing set. The results with the validation set were similar, and when a specific event was monitored (described in section 5.3 of Chapter 5), 22.9% of the tweets extracted were categorized as "injured or dead people" and 25.7% as "other useful information". Therefore, there are urgent tweets present on social networks, and the BERT algorithm can detect them, so it can be considered possible to detect urgent information amidst a lot of non-urgent information.

With the test set, the BERT model averaged 85% precision, 88% recall, and 86% F1-score. With the validation set, the average was 84% precision, 87% recall, and 86% F1-score. This information demonstrates that most tweets are correctly classified, ultimately answering the second research question (*Can the selected Machine-Learning Algorithm ensure that the vast majority of tweets are categorized without error?*).

Regarding the correlation algorithm, which is fully explained in section 4.4 of Chapter 4, some results are presented in section 5.3 of Chapter 5. Although we lacked the data to test our algorithm on particular MCIs, it was nevertheless able to identify correlations between texts and dates defined by us. With some in-depth testing, this algorithm would probably need to be adjusted for better performance. However, in answer to the third research question (*Is it possible to create an algorithm able to detect correlations between tweets and the information that is generated after emergency calls (PSAP information)?*), since we are able to obtain the dates and texts of tweets and PSAP incidents, then it is feasible to find correlations between them.

The aim of this work was to monitor social networks during MCIs in order to detect urgent information that could be used by stakeholders and decision-makers to better allocate resources and gain an even deeper understanding of the situation on the field. After analyzing the results obtained, we believe that this service is in accordance with these objectives and that it may be a very useful tool to help monitor MCIs. It is important to emphasize that this service is meant to supplement current offerings rather than to replace any existing monitoring methods.

6.2 Limitations

The present work contains some restrictions that are important to highlight in order to realize what can be improved in its context.

The entire system has been designed and built only for tweets written in English (only tweets written in English are extracted, the models are trained and evaluated with tweets written in English, PSAP incident texts are written in English, etc.). As is well known, there are mass disasters all over the world affecting people from many languages and cultures. People will express

themselves on social media in their own language if a certain MCI takes place in their country. Therefore, if an MCI takes place in a country where English is not widely spoken, this service needs to be adjusted.

Another major limitation of this work involves our correlation algorithm. The lack of PSAP data that we had for testing did not allow us to guarantee the best performance and optimization. Additionally, we need the PSAP data from each country where it is supposed to be implemented, and probably each country has a different way of handling the data, which makes the operationalization of this correlation service more complicated.

6.3 Future Work and Recommendations

There are many ways in which this work can be improved. Firstly, is important to address the first limitation of this work (the fact that the entire service is designed and built to only include tweets published in English). Since it doesn't seem feasible to build a dataset of disaster-related tweets for each language of the countries where disasters may occur (so that machine-learning models could learn from tweets made in that language), it would be a great improvement for the service to add a translation layer. In this way, tweets would be extracted in any language, then they would be translated and finally categorized. This would extend the usefulness of our service to several countries.

It would also be important to overcome the second limitation of this work by obtaining more PSAP incident data so that the correlation algorithm can undergo more tests in order to be adjusted and obtain better results.

Other transformed-based models could also be tested in future works, which could likewise provide outstanding results. For instance, the RoBERTa algorithm [30] may even guarantee better results and be more optimized than BERT, so it is possible that integrating this algorithm into future research work could lead to an increase in performance when categorizing texts.

Abbreviations

AI Artificial Intelligence. 5, 20

AIDR Artificial Intelligence for Disaster Response. 9

AMT Amazon Mechanical Turk. 9

API Application Programming Interface. 9, 12, 14, 18, 19, 23, 40, 45, 61

BERT Bidirectional Encoder Representations from Transformers. 11, 12, 34–37, 51, 53, 54, 61–63

BI-GRU Bidirectional Gated Recurrent Units. 36, 37, 51

BI-LSTM Bidirectional Long-Short Term Memory. 12

BOW Bag Of Words. 11, 12, 34

CBOW Continuos Bag Of Words. 10

CNN Convolutional Neural Network. 11, 12

CNN-BI-GRU Convolutional Neural Network Bidirectional Gated Recurrent Units. 36, 37, 51

FCUL Faculdade de Ciências da Universidade de Lisboa. 3, 7

GRU Gated Recurrent Units. 36, 51

HMMs Hidden Markov Models. 15

HTML HyperText Markup Language. 13, 14

IBM International Business Machines corporation. 15

IDF Inverse Document Frequency. 33

INOV-INESC Instituto de Engenharia de Sistemas e Computadores Inovação. 3

KNN K-Nearest Neighbors. 36

LSTM Long-Short Term Memory. 15, 36, 51

MCIs Mass Casualty Incidents. 1–5, 18, 19, 21, 61, 62

MEMMs Maximum Entropy Models. 15

ML Machine-Learning. 2, 4, 5, 12, 20, 25, 27, 28, 31, 32, 34, 45

MLP Multi Layer Perceptron. 11, 12

NER Named Entity Recognition. 15, 16, 21, 44

NG-ES Nightingale - Emergency System. 3, 17

NLP Natural Language Processing. 2, 5, 7, 14, 43, 61

NLTK Natural Language Toolkit. 10

OOV Out-Of-Vocabulary. 10, 34

POS Part-Of-Speech. 11

PRISMA Preferred Reporting Items for Systematic Reviews and Meta-Analyses. 7

PSAP Public Safety Answering Points. 3–5, 17, 18, 21–23, 39, 40, 43–45, 62, 63

SGD Stochastic Gradient Descendent. 36, 51

SMOTE Synthetic Minority Over-Sampling Technique. 31

SVC Support Vector Classifier. 36, 51, 53

SVM Support Vector Machine. 11, 12

SWAPP SoftWare mobile APPlication for citizens. 3

TF Term Frequency. 33

TF-IDF Term Frequency-Inverse Document Frequency. 33, 34

WWW World Wide Web. 13

XHTML eXtensible HyperText Markup Language. 13

Bibliography

[1] Twython—twython 3.8.0 documentation. https://twython.readthedocs.io/en/latest/, 2013. [Online; Accessed: 19 June 2023].

[2] Supervised vs Unsupervised Learning: What's the Difference? https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning, November 2022. [Online; Accessed 22 May 2023].

[3] What are distance metrics in vector search? https://weaviate.io/blog/distance-metrics-in-vector-search, Sep 2022. [Online; Accessed 23 June 2023].

[4] Welcome to apache opennlp. https://opennlp.apache.org/, 2023. [Online; Accessed 20 June 2023].

[5] Firoj Alam, Umair Qazi, Muhammad Imran, and Ferda Ofli. Humaid: Human-annotated disaster incidents data from twitter with deep learning benchmarks. In *Proceedings of the International AAAI Conference on Web and social media*, volume 15, pages 933–942, 2021.

[6] A Aswathy, Rekha Prabha, Lakshmi S Gopal, Divya Pullarkatt, and Maneesha Vinodini Ramesh. An efficient twitter data collection and analytics framework for effective disaster management. In *2022 IEEE Delhi Section Conference (DELCON)*, pages 1–6. IEEE, 2022.

[7] Steven Bird. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72, 2006.

[8] Phil Blunsom. Hidden markov models. *Lecture notes, August*, 15(18-19):48, 2004.

[9] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomás Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016.

[10] Kevin W. Bowyer, Nitesh V. Chawla, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *CoRR*, abs/1106.1813, 2011.

[11] Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[12] Denis Eka Cahyani and Irene Patasik. Performance comparison of tf-idf and word2vec models for emotion text classification. *Bulletin of Electrical Engineering and Informatics*, 10(5):2780–2788, 2021.

[13] Ashis Kumar Chanda. Efficacy of BERT embeddings on predicting disaster from twitter data. *CoRR*, abs/2108.10698, 2021.

[14] Mukesh Chaudhary. TF-IDF Vectorizer scikit-learn. https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a, jan 28 2021. [Online; Accessed 12 April 2023].

[15] François Chollet and others. Keras: The Python Deep Learning library. Astrophysics Source Code Library, record ascl:1806.022, June 2018.

[16] Agata Darmochwał. The euclidean space. *Formalized Mathematics*, 2(4):599–603, 1991.

[17] Kanika Dawar, Ashwanth J Samuel, and Raf Alvarado. Comparing topic modeling and named entity recognition techniques for the semantic indexing of a landscape architecture textbook. In *2019 Systems and Information Engineering Design Symposium (SIEDS)*, pages 1–6. IEEE, 2019.

[18] Ashwin Devaraj, Dhiraj Murthy, and Aman Dontula. Machine-learning methods for identifying social media-based requests for urgent help during hurricanes. *International Journal of Disaster Risk Reduction*, 51:101757, 2020.

[19] TensorFlow Developers. Tensorflow. https://www.tensorflow.org/. [Online; Accessed 12 May 2023].

[20] Synthesized SDK :: Synthesized Docs. https://www.tensorflow.org/. [Online; Accessed 10 April 2023].

[21] Irvin Dongo, Yudith Cadinale, Ana Aguilera, Fabiola Martínez, Yuni Quintero, and Sergio Barrios. Web scraping versus twitter api: a comparison for a credibility analysis. In *Proceedings of the 22nd International conference on information integration and web-based applications & services*, pages 263–273, 2020.

[22] Irvin Dongo, Yudith Cardinale, and Ana Aguilera. Credibility analysis for available information sources on the web: a review and a contribution. In *2019 4th International Conference on System Reliability and Safety (ICSRS)*, pages 116–125. IEEE, 2019.

[23] Jacob Eisenstein. *Introduction to natural language processing*. MIT press, 2019.

[24] MariaDB Foundation. https://mariadb.org/, Nov 2019. [Online; Accessed 13 April 2023].

[25] Wolf Garbe. SymSpell. https://github.com/wolfgarbe/SymSpell, 6 2012. [Online; accessed 28 May 2023].

[26] Purva Huilgol. Quick introduction to bag-of-words (bow) and tf-idf for creating features from text. https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/, Feb 2020. [Online; Accessed 12 April 2023].

[27] Sagar Imambi, Kolla Bhanu Prakash, and GR Kanagachidambaresan. Pytorch. *Programming with TensorFlow: Solution for Edge Computing Applications*, pages 87–104, 2021.

[28] Muhammad Imran, Carlos Castillo, Ji Lucas, Patrick Meier, and Sarah Vieweg. Aidr: Artificial intelligence for disaster response. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, page 159–162, New York, NY, USA, 2014. Association for Computing Machinery.

[29] Muhammad Imran, Prasenjit Mitra, and Carlos Castillo. Twitter as a lifeline: Human-annotated twitter corpora for nlp of crisis-related messages. *arXiv preprint arXiv:1605.05894*, 05 2016.

[30] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[31] Long Ma and Yanqing Zhang. Using word2vec to process big text data. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2895–2897. IEEE, 2015.

[32] Wes McKinney et al. pandas: a foundational python library for data analysis and statistics. *Python for high performance and scientific computing*, 14(9):1–9, 2011.

[33] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[34] Matthew J Page, David Moher, Patrick M Bossuyt, Isabelle Boutron, Tammy C Hoffmann, Cynthia D Mulrow, Larissa Shamseer, Jennifer M Tetzlaff, Elie A Akl, Sue E Brennan, et al. Prisma 2020 explanation and elaboration: updated guidance and exemplars for reporting systematic reviews. *British Medical Journal*, 372, 2021.

[35] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, 2010.

[36] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. *CoRR*, abs/1201.0490, 2012.

[37] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[38] Adwait Ratnaparkhi. A simple introduction to maximum entropy models for natural language processing. *IRCS Technical Reports Series*, page 81, 1997.

[39] Hannah Ritchie, Pablo Rosado, and Max Roser. Natural disasters. *Our World in Data*, 2022. https://ourworldindata.org/natural-disasters [Online; Acessed 20 February 2023].

[40] David Sellers and Jamie Ranse. The impact of mass casualty incidents on intensive care units. *Australian Critical Care*, 33(5):469–474, 2020.

[41] Hemlata Shelar, Gagandeep Kaur, Neha Heda, and Poorva Agrawal. Named entity recognition approaches and their comparison for custom ner model. *Science & Technology Libraries*, 39(3):324–337, 2020.

[42] Chenjie Song and Hiroyuki Fujishiro. Toward the automatic detection of rescue-request tweets: Analyzing the features of data verified by the press. In *2019 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*, pages 1–4. IEEE, 2019.

[43] TechTarget. What is the crontab command in unix? – techtarget definition. https://www.techtarget.com/searchdatacenter/definition/crontab, Feb 2023. [Online; Acessed 30 July 2023].

[44] Amazon Mechanical Turk. https://www.mturk.com/. [Online; Accessed 25 May 2023].

[45] Priyanka Tyagi and RC Tripathi. A review towards the sentiment analysis techniques for the analysis of twitter data. In *Proceedings of 2nd international conference on advanced computing and software engineering (ICACSE)*, 2019.

[46] Yuli Vasiliev. *Natural language processing with Python and spaCy: A practical introduction*. No Starch Press, 2020.

[47] Jiapeng Wang and Yihong Dong. Measurement of text similarity: a survey. *Information*, 11(9):421, 2020.

[48] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.

[49] Bo Zhao. Web scraping. *Encyclopedia of big data*, pages 1–3, 2017.