

1.0 Research Description

This chapter introduces the research which will be undertaken in the field of information mining for disaster management. It is divided into four sections. The first section explains the motivation for the research, discussing the limitations of existing systems as well as the underlying problem that must be addressed. The second section presents the general and specific objectives that the research aims to accomplish. The third section describes the scope and limitations of the presented research objectives. The fourth section provides the rationale behind performing the research. And lastly, the fifth section outlines the research methodology to be employed.

1.1. Overview of the Current State of Technology

In the Asia-Pacific Disaster Report 2010, a document prepared by the United Nations International Strategy for Disaster Reduction (UN/ISDR) and the United Nations Economic and Social Commission for Asia and the Pacific (ESCAP), it was noted that the Asia-Pacific region is four and 25 times more vulnerable to natural disasters as compared to the African and European or North American regions, respectively. Also in the same report, it is stated that despite generating more or less only one quarter of the world's gross domestic product (GDP), the Asia-Pacific region has accounted for up to 85 per cent of deaths and 30 per cent of global economic losses due to natural disasters during the year 1980 to 2009.

Disasters are events that may be natural or human-induced and sudden in nature which cause disruption on the normal functions of a community on a large scale leaving the it no time and capacity to respond (Virtual University for Small States of the Commonwealth [VUSSC], n.d.). These unfortunate events are often unavoidable and may happen anytime and anywhere. However, the damages and losses of such events can be reduced through effective and efficient disaster management procedures.

Disaster management is defined as the proper organization, management, and allocation of available resources and responsibilities in terms of preparedness, mitigation, response, and recovery for disaster situations with the goal of lessening the impacts of such incidents (International Federation of Red Cross [IFRC], n.d.). With this challenge of disaster management in damage reduction, disaster managers now face the daunting task of calculating and comparing costs and benefits of each step and decision made and correctly managing available resources during disaster situations (Rao, et al., 2007). In addition to being a task of a large scale, these managers must also be able to calculate these analyses at a very fast pace for these kinds of information are often perishable. Timely, accurate and effective use of available information is critical in disaster management scenarios as it allows better visualization of the disaster at hand and facilitates better and faster decision making for the authorities involved. However, due to the large amount of data and information that must be taken into consideration, manual procedures in facilitating disaster management often become herculean tasks for authorities. In order to address this problem, numerous disaster management systems have been developed. The purpose of these systems include, but are not limited to, creating and maintaining large repositories of information related to disaster management, automating the process of disseminating immediate and relevant information to the correct people, and allowing a visualization of the extent of the disaster at hand that creates an avenue for disaster managers to efficiently organize possible plans of action in response to the situation.

Information extraction plays a vital role in disaster management. It is a process with the goal of automatically retrieving structured information from unstructured machine-readable documents (Turmo, Ageno, & Català, 2006). With the popularity of the Internet, information that is potentially useful in disaster response becomes available online in the hours and days immediately following a disaster. The information may be in the form of online news reports, community blogs, posts in social networking sites and specialized web-sites put-up for various tasks such as locating resources, disseminating current situation reports, and re-establishing communications. Aside from these, information sent through SMS during disaster situations is also crucial for SMS is a basic component of all mobile phones and is more accessible to affected people as compared to the Internet. With information extracted from SMS, disruption to Internet services in an affected community will not stop the retrieval of information needed by responders. The use of information extraction in retrieving all these crowdsourced data and extracting important information from them would provide more and both professional, as in coming from official reports, and on-hand-experience-based, as in coming from the affected people themselves, information associated to a disaster which, in turn, will facilitate better decision making environments for disaster managers. However, it must be taken into consideration that the structure of information retrieved from Internet sources such as social networking sites and news sites is inconsistent, thus only information extraction systems that are able to adapt to these changes in structure can handle input coming from these sources. In addition, verification of crowd sourced information is also an issue as this type of information often comes from undefined and unspecialized groups of people. And lastly, information in these tense situations are often perishable, meaning they expire, and thus must be retrieved and analyzed as soon as possible. With the outlined reasons, a disaster management system coupled with adaptive information extraction and validation components that can function in near real time with or without the need to access the system's physical website or software to provide input is a community's best ally during a disaster situation (Fahland, et al., 2007).

SAHANA is among the different disaster management systems currently available. SAHANA serves as a large repository of information associated on-going disasters manually provided by victims through means of logging reports on the system's website. Information obtained will then be made available to responders and volunteers involved. Despite providing a well-structured and usable interface and data, SAHANA lacks a data collection module, does not implement a validation scheme, and requires people to access its website in order to log reports (Careem, et al., 2006).

In order to address SAHANA's limitations, a number of disaster management systems that make use of IE components emerged and HUODINI is one example. It is one of the first disaster management systems that made use of information extraction to offer the capability of extracting information from online sources (Fahland, et al., 2007). However, HUODINI did not allow users to push data into the system's repository of information, thus making valuable information from affected people irretrievable. In addition, the system also fails to implement a data validation scheme upon the information it extracts from its sources. This proves to be crucial for this unreliability of information may cost lives.

The RESCUE Disaster Portal is another and more recent system that makes use of an IE component. Like HUODINI, it makes use of information extraction techniques to identify and obtain information from multiple online sources (Ashish et al., 2008). In addition, the RESCUE project also allows viewers of the portal to push information similar to the functionalities provided by SAHANA but to a more limited extent. However,

like HUODINI, information provided by this system may not always be reliable as they do not implement any validation techniques upon information gathered.

Though information validation is a huge issue when it comes to disaster management and all previously mentioned systems fail to resolve it, there are particular disaster management systems that try to address this problem. Ushahidi (Meier, 2009a) is a free and open source platform that allows users to collect data via text messages, email, twitter, and web-forms. It is equipped with the Swift River (Blow & Jhalla, 2009) platform which performs monitoring, filtering, and ranking of information being fed to the system based on the user-specified domain. Though it addresses some of the issues in information validation, the sources where it extracts its information are limited. In addition, Ushahidi is only available in English.

In year 2010, the Center for Research and Epidemiology Disasters (CRED) ranked the Philippines as the world's most disaster prone country for the year 2009 (See, 2009). The Philippines was hit with a total number of 25 calamities, followed by China with only a total of 16, making the Philippines the most frequently hit by calamities posing serious threats to life and economy and the busy metropolis of Metro Manila is no exception to these threats. With the continuous migration of people to urban areas, natural calamities are more likely to leave destruction of a larger scale to these areas as compared to rural areas (Masing, 1994).

Being the capital the Philippines, Metro Manila should be able to quickly bounce back to normal conditions whenever disasters strike and this could be achieved with proper on-going disaster mitigation and response. There is a need for a disaster management system that will be able to provide near real-time and validated information immediately ready for use given a sudden disaster situation that will guide responders on rummaged community. Though efforts and initiatives to adapt and deploy SAHANA in the Philippines have already started, its lack of an information extraction module fails to make use of information readily available over the internet (Careem et al., 2006). In addition to this, the need to access the system's website in order to avail of its functionalities is not appropriate in the Philippine setting as most hardly hit areas are those that do not have access to the Internet. Currently, there is no disaster management system in the country that implements an information extraction component that will allow the extraction of information for disaster management from different online sources and SMS reports. And also, no existing system is taking advantage of the information made available through social networking sites. Furthermore, no existing system in the country can provide reliable crowdsourced information for the reason that no robust and proper data validation technique in terms of data gathering for this domain has been implemented yet.

1.2. Research Objectives

This section presents the general and the specific objectives of the proposed research.

1.2.1. General Objective

To develop a disaster management system that extracts and validates data from various sources in near real-time and provides visualization of the information to authorities concerned.

1.2.2. Specific Objective

The following are the specific objectives for the research.

1. To study and review existing information extraction and disaster management systems;
2. To study information extraction techniques to be used for gathering data for disaster management;
3. To evaluate existing tools and resources which could be incorporated in the information extraction components of the system;
4. To identify relevant information and possible sources of data in disaster management;
5. To analyze different techniques in information validation and;
6. To develop a validation scheme for the evaluation of the credibility of gathered information

1.3. Scope and Limitations of the Research

The research aims to design a disaster management system for on-going disaster mitigation and response focusing on providing the individuals concerned with validated information relevant to disaster management. The system will be integrated with an information extraction system as its means of gathering data.

Existing domain-dependent and domain-independent information extraction systems and existing disaster management systems will be reviewed in order to facilitate a better understanding of the core components of these systems and to determine how possible data sources for disaster management are structured. The architectures, approaches, and algorithms of these systems will be identified and studied.

Existing tools that can be incorporated to the information extraction component of the system will also be studied and evaluated.

The system to be developed will be community-based and will cover the whole stretch of Metro Manila. Its main focus will be natural disasters, particularly typhoons, floods, and droughts, human-induced disasters, particularly armed conflict, and a combination of both natural and human-induced disasters, precisely fires and landslides, as these have been consistently identified as among top 5 disasters that have affected the country both in terms of frequency and affected population of each year during the last four years, 2007 to 2010 (Citizens' Disaster Response Center [CDRC], 2007, 2008, 2009, 2010). Ad-hoc disaster situations will also be handled by the system.

In addition, the research also aims to study and to implement an adaptive information extraction system that will be able to perform near real-time information extraction from online sources such as blogs, online news sites, and social networking sites, and also from SMS reports.

Relevant information in disaster management includes the place of the event, the time and/or duration of the event, the persons involved in the event, the type of event, and the description of the event. Information will be gathered from sources such as local news sites, Twitter, Facebook, and SMS reports with the use of information extraction and mining techniques. The system will be capable of handling the English text and, to some extent, the Filipino text. Information extraction on text written in Filipino or Taglish will be handled with keyword-based searching only as problems in parsing of Filipino text may be encountered for there are no stable tools as of yet that can handle the Filipino, Taglish, and "text speak" languages.

Different techniques in validating information will also be studied. The approaches and algorithms used in these techniques will be identified, studied, and evaluated. Observations and conclusions from this experimentation will serve as basis for the validation scheme to be designed. The output will then be integrated to the disaster management system as its means of validating gathered information.

1.4. Significance of the Research

Metro Manila, also known as the National Capital Region, is composed of 13 cities and 4 municipalities, a total of 17 local government units. It houses Makati City, which is the financial center of the country, Quezon City, where main offices of the national government agencies are found, most of the country's business districts, wealth extremes, as well as major shopping centers and accounts for the 33 per cent of the nation's GDP (Pacific Disaster Center [PDC], 2005). Occupying 637 km² of land, it has become home to a rapidly growing population of approximately 23 million.

With the rapid growth in population and industrial development in the metropolis, inadequate infrastructural developments and environmental problems have become common (Masing, 1994). With these factors, a large scale disaster will undoubtedly leave Metro Manila in chaos. And the high vulnerability of the country to natural calamities does not make things any better. With a big bulk of the country's economic stability depending on the daily functions of Metro Manila, disaster managers should be able to devise a way to prevent more people from entering known disaster inflicted areas and to allow quick and effective relief operations in order to facilitate faster recovery.

Though the country is rich with responders and volunteers with lots of working experience in handling disaster management, on-going disaster mitigation and response still proves to be difficult as perennial constraints and problems will never cease to exist especially in metropolitan areas. Lourdes Masing, a coordinator of the Philippine Nation Red Cross, states that lack of proper coordination among agencies responding to disaster situation is a problem often encountered by volunteers when participating in relief operations. This problem is often attributed to the lack of adequate information with regards to the extent of the disaster or the actual size of the affected population. And because of this, problems in relief operations arise which result to some segments being overly bombarded with relief supplies while others are left unserved. She also states that there is a lack of disaster information materials in the barangay level, leaving authorities nearest to the disaster site with no material to organize plans of action.

The problem is not the lack of information regarding affected communities for these are usually the first people wanting to seek help, doing every possible way to be able to relay their messages to responders. As evidenced by the tsunami that devastated Japan, the social networking giant Facebook was able to beat television and newspapers in delivering the information regarding the disaster to the world as numerous Facebook users instantly posted first-hand accounts of the incident seconds after the tsunami struck (Mohan, 2011). Vast amounts of information, particularly those that the victims themselves make publicly available with the use of social networking sites, are scattered all over the internet. In the event that the affected community loses access to the Internet, victims turn to their mobile phones for help by sending SMS to people with the capability to assist. It is just that no existing system, particularly in the Philippines, are gathering, extracting and validating these information for the use of responders and volunteers.

In addition, it must also be noted that parts of an affected community are not equally affected by a crisis. Therefore making these people, the standby volunteers, the real first responders for they are always there. Unlike paid search and rescue teams and salaried emergency responders, these people are the ones who are at the site the second a disaster occurs. As of now, there is no avenue for these people to learn of the whereabouts, needs, and situation of victims needing their assistance just a few meters away.

The proposed system will alleviate the problems presented for it is capable of extracting relevant information from articles and other sources of data made available by people online through means such as social networking sites and community blogs, and mobile technology, may these people be belonging to the affected population or not in near real time. The system will also apply validation schemes on these crowd sourced information in order to ensure its validity and accuracy. Validated information will then be synthesized and displayed in a manner that is most relevant to responders from both government and non-government organizations. The visualization will also be made available to the public or standby volunteers thus establishing a link between those who are most affected and those who are less affected at least until external help arrives.

1.5. Research Methodology

This section identifies the different activities that will be performed throughout research. Scrum-based methodology, an iterative agile software development life cycle process, is to be adopted in developing the system in order to facilitate and maintain the flexibility and robustness of the research when responding to the continuous changes and updates.

The phases that the research will undergo include investigation and research analysis, system design, system development, system integration and testing, system evaluation, and documentation. Regular consultation with the thesis adviser will also be done throughout the whole period of the research.

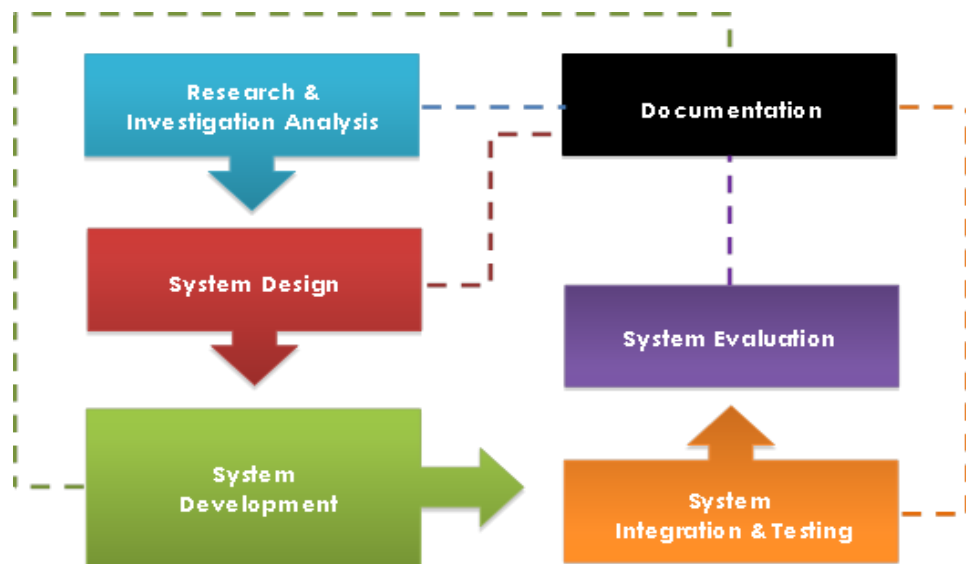


Figure 1.1: Research Phases

1.5.1. Investigation and Research Analysis

In this phase, fundamental knowledge of the different modules and requirements of a disaster management system, and as well as study the different techniques and algorithms can be used in implementing an information extraction system are developed. The resulting research of this phase serves as a preliminary guide during the system design and system development phases. Existing tools related to information extraction that can be used in developing the system are also studied and evaluated during this phase.

1.5.2. System Design

Findings from the investigation and research analysis phase are used in choosing the appropriate algorithms, architecture, and approach to be used in coming up with a feasible research output that coincides with the proposed research goals.

1.5.3. System Development

In this phase, actual coding and implementation of the system with the use of the proposed design is done in an iterative manner. Data gathering for the system is also done during this phase. Daily and weekly meetings are conducted in order to assess the progress of each member and the rese as a whole and to plan for succeeding tasks.

1.5.3.1. Sprints

Instead of using the original 30-day timeframe of scrum for sprints, a one week timeframe is used. Each sprint will have the goal of finishing a part of the research until all phases of the research are complete. Each member is expected to produce a working output based on the tasks assigned to him during planning meetings. The tasks may vary between developing a system component and doing further research work.

1.5.3.2. Sprint Planning Meetings

These are weekly meetings wherein the tasks that must be accomplished for a given period of time are discussed. The tasks can be in the form of coding functions for the program itself or doing further research work. The assignment and division of tasks for each individual for the following week is decided during this meeting.

During these meetings, the output of each individual based on the task allocation during previous sprint planning is evaluated. Unmet tasks will be discussed and will be carried on to the next sprint.

1.5.3.3. Scrum Meetings

10-15 minute meetings are conducted everyday with the purpose of updating one another of the progress of each individual in their assigned tasks. This activity is to ensure that there is daily progress for all members and to discuss problems that may be preventing them from finishing assigned tasks.

1.5.4. System Integration and Testing

In this phase, all modules are to be integrated. Aside from unit tests done during sprints, another testing phase will be done after the modules have been integrated. System components found to be faulty are fixed during this phase in order to prepare the system for actual user testing.

1.5.5. System Evaluation

In this phase, the system will be evaluated by testing the accuracy of the information extraction module of the system.

1.5.6. Documentation

Each activity done throughout the entire process of development is to be documented. The purpose of this activity is to keep track of modifications done in the system and also to provide reference for future modifications.

1.6. Calendar of Activities

Table 1.1, 1.2, 1.3 and 1.4 shows a Gantt chart of the activities for the thesis period. Each bullet represents approximately one week worth of activity.

Table 1.1: Timetable of Activities (May 2011 to August 2011)

Activities (2011)	May	Jun	Jul	Aug
Investigation & Research Analysis	•	••••	••••	••••
System Design				
System Development				
System Integration & Testing				
System Evaluation				
Documentation	•	••••	••••	••••

Table 1.2: Timetable of Activities (September 2011 to December 2011)

Activities (2011)	Sept	Oct	Nov	Dec
Investigation & Research Analysis	•••	••		
System Design	••••	••••	••	••
System Development			••	••
System Integration & Testing				
System Evaluation				
Documentation	••	••	••	•••

Table 1.3: Timetable of Activities (January 2012 to April 2012)

Activities (2011)	Jan	Feb	Mar	Apr
Investigation & Research Analysis				
System Design				
System Development	••••	••••	•••	••
System Integration & Testing	••	••	•••	••••
System Evaluation				•
Documentation	•••	•••	•••	•••

Table 1.4: Timetable of Activities (May 2012 to August 2012)

Activities (2012)	May	Jun	Jul	Aug
Investigation & Research Analysis				
System Design				
System Development				
System Integration & Testing				
System Evaluation	••••	••••	••••	••
Documentation	•••	••••	••••	••••

2.0. Review of Related Literature

This chapter discusses the features capabilities, and limitations of existing research, algorithms, or software that are related or similar to the thesis.

2.1. Disaster Management

The International Federation of Red Cross has defined disaster management as the proper organization, management, and allocation of available resources and responsibilities in terms of preparedness, mitigation, response, and recovery for disaster situations. The main goal of disaster management is to reduce, or avoid, the potential losses from hazards, assure prompt and appropriate assistance to victims of disaster, and achieve rapid and effective recovery.

Disaster management is considered as an enormous task due to the reason that disasters are not usually confined to a particular location. Adding to this, disasters are not known to disappear quickly and are known to leave long-term traces of devastation on the places they ravage. Because of these reasons, there is a need to optimize the efficiency of planning and response which can be achieved with proper management. Collaboration among governmental, private and community levels is often necessary for there is only a limited amount of resources. This level of collaboration requires a coordinated and organized effort to mitigate against, prepare for, respond to, and recover from emergencies and their effects in the shortest possible time (VUSSC, n.d.).

The disaster management cycle is composed of four phases: (1) mitigation, (2) preparedness, (3) response, and (4) recovery. Disaster management is a cyclical process; however, the phase of one cycle does not necessarily have to end before another phase can begin. Most of the time, phases of disaster management take place concurrently. Timely decision making during each phase results in greater preparedness, better warnings, reduced vulnerability and/or the prevention of future disasters.



Figure 2.1: Disaster Management Cycle

2.1.1. Mitigation Phase

Mitigation activities aim to eliminate or reduce the probability of a disaster to occur (Warfield, 2005). If a disaster is inevitable, the goal shifts to reducing the disaster's effects. Appropriate measures in both national and regional development planning are determining factors in the outcome of disaster mitigation processes. The effectiveness

of mitigation activities is largely dependent on the availability of information on hazards, emergency risks, and the knowledge in countermeasures to be taken.

2.1.2. Preparedness Phase

During the preparedness phase, programs aiming to strengthen the technical and managerial capacity of government organizations, and communities are implemented in order to ensure that disaster responders are ready to act given a sudden emergency situation (Warfield, 2005). Aside from having well prepared disaster responders, the preparedness phase also involves ensuring that essentials such as reserves of food, medicines, and survival equipment are maintained in cases of a sudden onset of a catastrophe, may it be local or national.

2.1.3. Response Phase

Emergency response aims to provide immediate assistance and support to disaster stricken communities (Warfield, 2005). Assistance may range from providing aid such as assisting refugees with transport, temporary shelter, and food, to establishing semi-permanent settlement in camps and other locations. The main focus of this phase is to provide disaster stricken communities with basic necessities until the situation becomes under control.

2.1.4. Recovery Phase

Once an emergency gets under control, a number of programs and activities are implemented in order to help affected communities restore their lives prior the emergency (Warfield, 2005). These activities continue to operate within the affected community at least until everything gets back to normal.

2.2. Existing Disaster Management Systems

Disaster management systems (DMS) are technological aids that help in facilitating effective management of information during disaster situations (VUWSC, n.d.). With the proper usage of DMS technology, improvements in terms of the effectiveness taken on the different phases of disaster management can be achieved. Existing disaster management systems are presented in the following sections.

2.2.1. SAHANA (Careem, et al, 2006)

Sahana is a Free and Open Source Software (FOSS) application that aims to facilitate effective information management in the processes of relief operations, recovery and rehabilitation in disaster situations.

The Sahana Application Framework makes use of a set of APIs in order to provide core and optional capabilities to its modules. It uses the Location API for its location hierarchy selection and storage functionalities, GIS API for its GIS mapping interface and spatial capability, Reporting API for its automated and custom reporting capability, and as well as the Synchronization API in order to facilitate database synchronization among Sahana instances.

The Sahana core modules are built on top of the application framework and libraries. The core modules are as follows:

- Organization Registry (OR): This module keeps track of all organizations of responders assigned to specific disaster regions. Information regarding these organizations such as contact information, personnel, areas where each is active, and the range of services each can provide are stored in the registry using this module;
- Request Management System (RMS): This module allows victims to post requests of aids, and responders to provide pledges of support to posted requests. Fulfillment of requests is also being tracked in this module.
- Shelter Registry (SR): This module maintains a registry of evacuation centers. Information such as the centers' location and facilities provided are stored.
- Missing Persons Registry (MPR): This module allows victims and responders to report information about missing and found persons. Information regarding the reported missing persons, the people who report them to be missing, along with the information of the person making the search are captured and are stored in the registry. The information gathered will then be used by the system to generate reports that can help people in their search.

As what has already been mentioned, Sahana also provides optional modules. The modules take up the outermost layer of the framework and are only installed upon the request of clients. Among the optional modules currently available are the following:

- Volunteer Coordination System (VCS)
- Child Protection System (CPS)
- Inventory Control and Catalog System (ICS & CS)
- Situation Mapping (SM)
- Data Import (DI)
- Mobile Messaging (MM)

2.2.2. HUODINI (Fahland, et al., 2007)

HUODINI (HUMbOldt Disaster MaNagement Interface) provides visualization of disaster-related heterogeneous information. In this disaster management system, information is collected via a number of freely available data sources on the web, such as news feeds, personal blogs, tagged images, and seismographic information.

In order for the system to be able to integrate and understand information gathered, information is first converted into schema less resource description framework (RDF). This is to allow flexibility in the type of information that will be integrated to the system. Information is then tagged with time and location in order to facilitate queries regarding events reflecting only the information needed by the user who requested the query. In order to handle textual data, the system has also implemented an information extraction module. The visualization of information is presented with the use of Google Maps.

Currently, the system is limited to handle information associated to earthquakes only. In addition, it can only integrate poll-based data sources. Information is carried out as a one-shot process making it unsuitable for continuous information supply.

2.2.3. RESCUE (Lickfett, et al., 2008)

The RESCUE Disaster Portal is an easily customizable disaster web portal that provides a set of component applications to be used by responders to provide access to information related to disasters and emergency situations. It features situation overview with maps, announcements, press information, emergency shelter states and tools for

family reunification, with the goal of equipping first respondents with vital information in a short span of time.

Currently, the RESCUE Disaster Portal is deployed on Apache Tomcat 5.5 servlet container with MySQL as its relational database. Information, such as announcements, press releases, and information from websites, are stored in the relational database and is inputted into the system in multiple ways. Some information are manually inputted to the system by the administrator, while other information are obtained with the use of crawlers. The information gathered are presented in a website.

RESCUE Disaster Portal is implemented using JAVA technologies, the Spring Framework, and the Hibernate OR.

Online information aggregation and situation mapping are the core capabilities of the system as most of the modules of the system are dependent on the features provided by these modules. Aside from this, JSP templates are used for the navigation and layouts in visualizing information.

2.2.4. Ushahidi Platform (Meier, 2009a)

Ushahidi is an open source platform developed for information gathering, and visualization and interactive mapping of gathered information. The platform can handle all kinds of disasters; clients just have to specify what disasters the system needs to focus on. Its main goal is to help in data visualization that will be used for response and recovery. It does this by combining crisis information coming from reports generated by citizens, media, and NGOs and plots the results on a map with the use of geographical mapping tools.

One of the key components of Ushahidi is its ability to make use of mobile phones as a means of sending crisis updates to subscribers and receiving crisis incidents from reporters. This was implemented in response to instances wherein the Internet becomes hard to access because of disruptions in the network and brownouts. Another key component of the Ushahidi platform is the information validation scheme which is implemented with the integration of the SwiftRiver platform. During the first 24 hours of a disaster, massive amounts of data overwhelm the repository of disaster management systems, and as an answer to this problem, the SwiftRiver platform was created. Its goal is to offer an easy to use, free, and open source alternative to expensive proprietary MIS systems that is able to process large amounts of data in a short period of time. Upon gathering information from different sources, the Ushahidi passes these information to the SwiftRiver platform for validation. Real-time data from sources such as Twitter, SMS, Email, RSS feeds, and web-forms are filtered and verified by SwiftRiver with the use of crowdsourcing techniques. It compares information gathered to already verified sources with higher authority and trust ratings. It then gives the unverified information scores based on the comparison made.

After filtering out unimportant information, Ushahidi tabulates the data into formalized structures by using ready-made templates in the system's database. It extracts information by using templates that is assumed used by the input data, which is achieved by searching for patterns and structures in the writing styles of the input. After processing the gathered information, events are then plotted on a map, including the location and the type of disaster, with the use of geo mapping tools.

2.3. Information Extraction

Information extraction is the process of extracting valuable information with the use of templates. Templates are premade database labels which should be filled out in order to create a formalized and structured textual document that will be entered into the database. There are five steps in information extraction: segmentation, classification, association, normalization, and deduplication.

Segmentation, which is the first step, is the process of collecting text snippets that are candidates to fill a specific field in the template. This determines the start and end boundaries of the text to be filled. After segmenting the different text snippets, they are then classified according to their appropriate field in the template. After classifying the text snippets, texts belonging to same records are grouped together. Relationships between text snippets that correspond to one document or event are established. But in order to establish formalized and structured textual document with consistent structure, normalization needs to take place. This process standardizes text snippets in such a way that it can be compared to each other. The last step is deduplication. This process is concerned with the removal of duplicates in order to remove redundancies and inconsistencies in the database.

Information extraction can be implemented in various ways from simple information extraction to more complex processes. For simple extraction, regular expressions can be used to handle, filter, and sort out information that will be extracted. On the other hand, for complex processes, combination of expressions, user-defined algorithms, and other mediums can be used to fit in to the needs of the planned system. An example application here is to use machine-learning methods such as building decision tree models, if-else rules, and other models to maximize the efficiency and performance of extracting information. With machine-learning, the rules do not need to be hand tuned since the machine will be taught how to handle, and improve its extraction module (McCallum, 2005).

Another implementation is by using the distillation process. This method can be done by retrieving information as a response to a query or by extracting snippets of texts. These sentences are weighed with respect to their relevance and then removing inconsistencies and redundancies in the database. It is also important to reduce the information being gathered to its maximum in order to get the full efficiency while processing extracted information. Techniques such as link analysis, expansion of query, and keyword search can reduce the size of its target. Also, semantic search is a process where it searches information in the database and complements it with semantic information to make the results more information. After the database is fully reduced to its maximum, co-reference analysis will relate records of each other to be part of searched queries when searching one of its co-references.

2.4. Existing Information Extraction Systems

Existing information extraction systems are presented in the following sections.

2.4.1. Domain-Dependent Information Extraction Systems

Domain-dependent IE systems extract information from a structured body of text with a specific or defined structure. These IE systems are only capable of extracting information from a single domain. This is due to the fact that most domain dependent IE

Systems use rules that only apply to a specific domain making these systems' migration unfeasible.

Resume Information Extraction System (Resume IES) (Karamath & Akyokus, n.d.)

Resume information extraction with named entity clustering focuses on extracting valuable information from resumes in order to simplify and speed up the process of finding job position candidates. The system is summarized into 4 phases that completes the information extraction process namely text segmentation, named entity recognition, named entity clustering and text normalization.

In the first phase, a resume is segmented into blocks according to their information types, which is called text segmentation. The system stores blocks of related information for every common heading found in most resumes. These common heading are stored in a dictionary and are used to differentiate one segment from another. All of the text between the heading and the start of the next heading is accepted as a segment. The second phase is named entity which makes use of chunkers that recognizes the entities throughout the resume. Each chunker is designed to search for a specific information in the resume and runs independently from one another. The first chunker searches for known names through dictionaries of well-known institutions, companies, and academic degrees. The second chunker looks for prefixes and suffixes that are commonly used such as "University of", "College", "Corp", etc. The third chunker tries to find clue words like prepositions that is nearby the target word to be recognized. For example, the "at" preposition commonly signifies the workplace in a work experience header. The last chunker searches for known patterns. For example in names, known patterns are the capitalization of letters per each start of a word. The third phase enables the independent named entities be formalized by clustering related information with one another to form a block of information. The system associates related entities into a group depending on their type and their proximity in the resume. The last phase transforms some of the named entities to make them consistent. This process is important in order for the information be well organized and consistent making it easier to use. For example, in BS Computer Science, BS is expanded to Bachelor of Science to make all of the information uniform with each other.

Information Extraction for e-Legislation (Lim, Miranda, Trogo, & Yap, 2010)

Information Extraction for eLegislation (IEfeL) is an information extraction tool that is used to extract valuable information from different types of documents found in the Blue Ribbon Committee (BRC), an independent and exclusive commission of nonpartisan statesmen and experts formed to investigate some important governmental issue (Princeton WorldNet).

The system's architecture consists of the following modules: the pre-processor, filter, pre-parser, semantic tagger, coreference resolution, and template filler. The pre-processor module cleans the input accepted and removes the noise in the data. It is made up of a Unicode tokeniser, which splits a sentence into tokens; a sentence splitter, which outputs a list of sentences; a cross reference, which is the database of synonymous informal words; a POS tagger, which tags tokens according to their part-of-speech label; an unknown word module, which attempts to label unknown words using context; and a named entity recognition module, using a statistical method for classifying entities (specifically proper nouns) produced by POS taggers. The filter accepts tokens from the semantic tagger and removes tokens that are needed by succeeding modules. The pre-parser groups tokens tagged by POS according to noun phrases and verb phrases. The semantic tagger removes ambiguities and identifies the

type of template where the relevant phrases, words, or tokens will be used for. The coreference resolution is used to solve referencing issues for pronouns and noun phrases, using the Ruslan Mitkov approach. The last module of the IE system, the template filler, fills up the fields of the templates to be used.

2.4.2. Domain-Independent Information Extraction Systems

Unlike domain dependent IE systems, Domain-independent, or adaptive, information extraction systems do not use rules that are specific to a domain. IE systems belonging to this category do not rely on a user-specified domain as its main purpose is to build a system can be easily migrated to other systems and will be able to adapt to different domains. Even if a system is trained to handle inputs from a specific domain, it will still be able to extract information from different domains, with minimum or no effort for retraining.

AVATAR (Jayram, et al., 2006)

AVATAR is a domain independent information extraction system that uses rule based techniques to extract information from text documents. It also uses probabilistic database techniques to do the extraction process. These techniques involve creating frameworks that would be used to map queries in the database.

Text analytic programs used in the information extraction process are called annotators and the objects that they extract are called annotations. AVATAR identifies two types of annotators, namely base annotators and derived annotators. Base annotator uses the basic rules to identify an annotation; whereas derives annotator uses annotations from both base and derives annotations. Each annotator are composed of a set of rules called meta-rules, an example is a meta-rule is for the annotator of Names, the rule identifies a word as a name if the word starts with a capital letter then followed by a series of small letters. Another example of a meta-rule but of a derived annotator is a more advanced rule for identifying a name, the conditions for this rule is a presence of a title (e.g. Dr. Mr.) then followed by a name (the rule previously mentioned).

SALEM (Biafioli, et al., 2005)

SALEM (Semantic Annotation for Legal Management) is an NLP-based system for classification and semantic annotation of Italian law paragraphs. SALEM is a domain-independent system that uses NLP and information extraction techniques in tagging the law documents. This system addresses the problem of automatically enriching legal texts with semantic tags.

SALEM performs two main tasks. First, it assigns each law paragraph to a given legislative provision type. A legislative provision type is the part of a law section where it expresses permission or an obligation for some actor to perform or not perform a certain action. The second task is to tag parts of the paragraph with domain-specific semantic roles which identifies the legal entities pointed out in the legislative provision. Legal entities would refer to actors, actions and properties.

SALEM takes in single law paragraphs and outputs a semantic structure by tagging. This is achieved by following a two-stage strategy. The first step involves a parsing system which pre-processes each law paragraph to give a somehow shallow syntactic analysis. This strategy is called syntactic pre-processing. The second and the last strategy called the semantic annotation, feeds in the syntactically pre-processed text from the first strategy. These steps are implemented as finite state automata.

TextRunner (Etzioni, et al., 2008)

TextRunner is a domain independent Open IE. It identifies tuples (related entities) in the database and the relation between these entities. The system first identifies all the positive tuples in the input. It does this by first passing the input into a parser, it then compare each pair of noun phrase in the document. All the positive noun phrases are stored together with their identified attributed. Dependencies between entities are then identified using the initial parsing. After all of this, TextRunner uses the selected attributes to derive a classifier model using the Naive Bayes approach. TextRunner passes the input into a Single-Pass Extractor. In this step all the noun phrases from the positive tuples are passed into a noun phrase chunker. The chunker then identifies which words in the noun phrase are likely to be part of the entity. Tuples with noun phrases that have no identified entity are discarded. It also eliminates useless tuples, tuples for over specifying. After this step, the system then runs the intermediate output into a redundancy based assessor. The task of the redundancy check assessor is to check of the information generated overlaps. Overlapping information are then discarded.

IEKA (Wong & Lam, 2007)

IEKA (Information Extraction Knowledge Adaption) uses a wrapper machine learning technique. IEKA is a wrapper technique that resolves the limitation of wrappers to one site, its main objective it to be able to train itself to a new site with no or minimal human intervention. It does this by identifying two crucial information, site-invariant data and site-dependent data. Site-invariant data are Elements of a page that reflects that of another site of the same domain. Site-dependent data are Elements of a page that reflects that of another page of the same site.

IEKA works by first obtaining pages to train on. The pages taken by IEKA are main example page and an auxiliary example page. The main example page is the page of the same domain as the other pages that IEKA has been extracting information from. The auxiliary page is that of a different domain. Since websites uses the same format in different web pages the information that are similar between the two example pages are commonly used for formatting. IEKA then identifies the site-dependent data. It does this be first building a DOM (Document Object Model), and identifying every path to a text-fragments. The text-fragments are then compared using a modified k-nearest neighbor to their counterpart (located at the same path) text-fragments from the auxiliary page.

Text-fragments that are not similar are labeled potential text-fragments. IEKA then uses the site-invariant data to identify, which of the potential text-fragment are desired text-fragments. Since site-invariant data are common from different sites of the same domain, it is used to scan for recognized pattern from the potential text-fragment to identify the desired text-fragment. An example given in the text is the identification number from two different web sites that pertain to the same movie. The identification number of a movie is dependent on the specific site, but both of the identification number still shares similar features which could help identify the data. The information generated by these steps is then used to build a new model or rules to be used for the new web site.

ELIE (Finn, 2005)

ELIE is a supervised text-classification approach to information extraction. Text classification is usually used to classify the entirety of the text according to its nature, type or genre; it commonly uses all the information in the text as attributes to properly

classify it, whereas information extraction identifies the desired information by learning the relationship of a given word to the word surrounding it. In the case of ELIE, it uses text-classification to classify the word to either the start of the desired information, end, or neither. It does this by identifying attributes of each word in the text. The attributes used by ELIE are POS (part of speech), Gaz, Orthographic, and the attributes of the words that precede and follow the given word. Gaz is a dictionary containing first, last names of persons, names of streets and other proper nouns. Orthographic is the relation of the given word to the surrounding words. It uses these attributes to build a model that would be used to label to words. The output produced by the model is a high prediction model but will a low recall. ELIE improves the output by running the output through another layer of classifier. The second layer, unlike the first is more focused, rather than classifying the words of an entire document, it classifies only a specified number of words that follows the start boundary or precedes the end boundary identified by the first layer. The second layer of ELIE improved the recall of the overall output without sacrificing the prediction.

3.0. Theoretical Framework

This chapter presents a discussion on the different theoretical concepts associated to disaster management and information extraction systems, and as well as common architectures, approaches, modules, and resources needed in developing such systems.

3.1. Information Extraction Architectures

This section discusses the main modules of three different IE architectures and some systems that made use of them.

3.1.1. Generic Information Extraction Architecture (Hobbs, 1993)

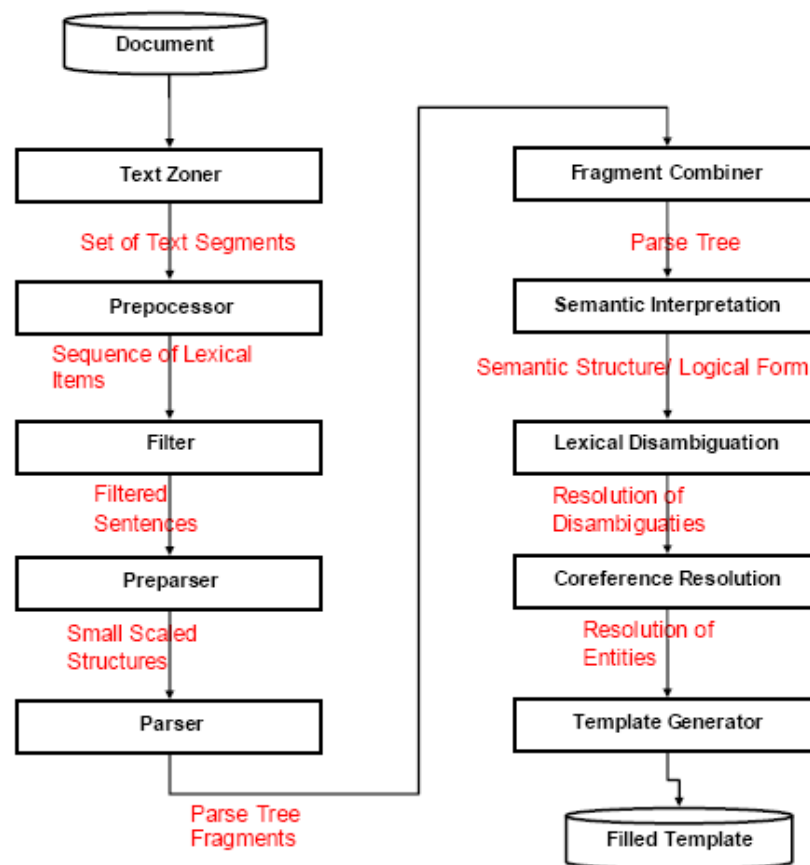


Figure 3.1: Hobb's Architecture

Hobbs has provided a general architecture for most information extraction systems that are in existence (Hobbs, 1993). He argues that though information extraction systems may have their own architecture they would still implement the functions of the following modules: Text Zoner, Preprocessors, Filter, Parser, Fragment Combiner, Semantic Interpreter, Lexical Disambiguation, Coreference Resolution, and a Template Generator (see Figure 3.1). The Text Zoner module turns a text into a set of text segments that is passed to the Preprocessor. The preprocessor turns the text segments

into a sequence of sentences by locating the sentence boundaries and produces a sequence of lexical items, a word that comes with its lexical attributes, for each sentence. The filter module removes sentences that are identified as irrelevant to the extraction process. Sentences are identified as irrelevant when there are no words that signal relevant events to the domain. The preparser identifies small-scale structures from a series of lexical items, which helps simplify sentence parsing. The parser takes in the series of lexical items or small-scale structures from the preparser and produces a parse tree that may be not be complete. The set of parse trees generated by the parser is then combined to form one whole parse tree for the sentence by the Fragment combiner. The Semantic interpreter translates the parse trees into a semantic structure or logical form. The Lexical Disambiguation module turns a semantic structure that contains ambiguous predicates, generated by the semantic interpreter, into a structure that contains unambiguous predicates. Coreference resolution or discourse processing turns a tree-like structure in a network-like structure by merging different descriptions of the same entity into a single node. Template generation creates the template of the form required by the evaluation from the structures generated by the previous modules (Hobbs, 1993).

3.1.2. FASTUS (Appelt et al., 2004).

FASTUS (Finite State Automation Text Understanding System) is a system that extracts information from a free text source for entry into a database and for other applications. FASTUS performs the extraction in successive stages such as processing of input, pattern matching and composite structure building. FASTUS revolves around the central idea of using a set of cascaded, non-deterministic finite-state transducers to separate the processing into several stages (see Figure 3.1).

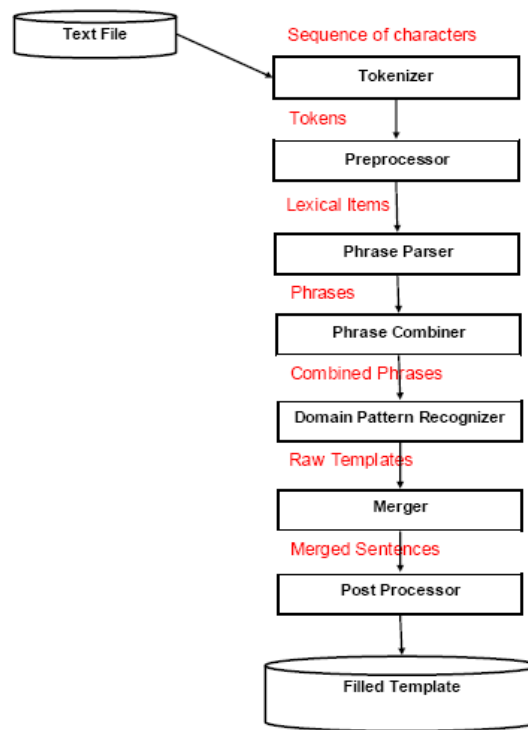


Figure 3.2: FASTUS Architecture

The first transducer is the tokenizer. It produces symbolic and numeric tokens as output. This is then passed on to the preprocessor. The preprocessor is tasked to recognize multiword lexical items, and some company and personal names. It produces lexical-items as output. The output is then passed on to the phrase parser as input. The phrase parser breaks down the input stream into noun groups, verb groups and particles. Aside from this, the phrase parser also identifies the head of each constituent to some extent. The output of the phrase parser is then passed on to the phrase combiner where they are combined into larger phrases of the same type. The combined phrases are then passed as input to the domain pattern recognizer which matches the phrases to patterns relevant to the information to be extracted. The output of the domain pattern recognizer is partially instantiated raw templates which will then be merged by a merger. Lastly, the raw templates are structured to its final form by the post processor.

The architecture of FASTUS can be summarized as consisting of text scanner and finite state machines. This simple and straightforward architecture of FASTUS allows it to provide fast and reliable results.

3.1.3. GATE (Cunningham et al, 2002)

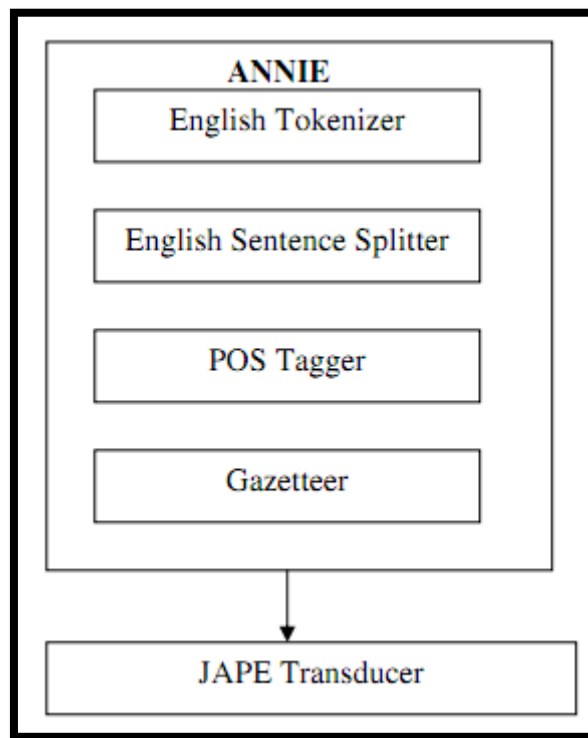


Figure 3.3: ANNIE's Architecture

GATE is an open-source language engineering tool that is capable of information extraction. The components of GATE can be categorized into three resources: Language Resources, Processing Resources, and Visual Resources. Language Resources are resources that are used to describe or define a language; examples of this type of resource are lexicons, corpora, and ontologies. Processing Resources are modules that are used to generate information from a source; examples of this resource are parsers, generators, and n-gram modelers. Visual Resources are module of GATE

that is used to visualize the data into a more user readable format. All modules within these three resources are called CREOLE (Collection for Reusable Object for Language Engineering) and are described in an XML file for easy import into GATE. Software that makes use of this architecture is ANNIE, basically ANNIE can be considered as packaged GATE software with NLP tools included with the resources.

The NLP tools provided by ANNIE are tokenizer, sentence splitter, POS tagger, gazetter, finite state transducers orthomatcher, and coreference resolver (see Figure 3.3). Besides the included NLP tools of ANNIE, the GATE architecture also enable adding and modifying existing NLP tools. The GATE architecture uses these tools to process and ultimately extract information from the sources. These tools are used sequentially by the architecture, with the tools and the order specified by the user. For each module the inputs are also specified by the user, the inputs can be the source document, a file within the language resource, or the output of the previous module (Cunningham et al, 2010).

3.2. Information Extraction Modules

The following sections describe Document Retrieval, Tokenization, POS Tagging, Word Sense Tagging, Named Entity Recognition, Coreference Resolution, which are common IE modules.

3.2.1. Document Retrieval

Document retrieval is vital in knowledge acquisition. This collection of documents is then divided in two sets – the training data set and the test data set. The information extraction system must learn as many extraction rules as possible for it to obtain relevant information from a text. However, that can only be possible if the system is trained on a diverse training data set.

A conventional Web crawler is used to fetch a high rate stream of documents from the Web. Depending on the needs of the extensible crawler's applications, this crawl can be broad, focused, or both. For example, to provide applications with real-time information, the crawler might focus on real-time sources such as Twitter, Facebook, and popular news sites.

Crawler4j is an open source web crawler developed in java that provides the user simple interface for crawling the web (Ganjisaffar, 2010). All that's needed to be configured is defined by two functions that should be overridden by the programmer. These two functions are:

`shouldVisit(webURL)` – this function decides whether the URL in the parameter should be crawled or not.

`visit(page)` – this function is called after the content of a URL is downloaded successfully and stored in a page variable. This function lets you collect the text, links, url, and docid of the downloaded page.

After configuring the two functions required by the system, it is now up to the programmer to design a controller that will contain the number of seeds of the crawl, the destination folder of all crawled data, and number of concurrent threads.

Within the `crawler4j.properties`, you can configure more advanced options helpful to control the crawling process. One of the key properties is the crawl depth. This allows

the programmer to define how deep the crawler should go. For example page “A” links to page “B” and then links to page “C” and so on, this property will be able to limit the depth of what it crawls for.

With this tool, programmers does not need to keep track of web pages it previously crawled for since the tool will automatically manage this type of conflict.

3.2.2. Tokenization

Tokenizers divide the character sequence into sentences and the sentences into tokens (Webster,1992). Not only words are considered as tokens, but also numbers, punctuation marks, parentheses and quotation marks. Tokenization is quite similar to the breaking down of a sentence into words. The difference of the two is that a token is not necessarily a word. As an example, let us consider the two words “you’re” and “O’neill”. “O’neil” can be seperated as “O” and ”neil” or “O’neil” but “you’re” cannot become “you”-”re”.

In languages such as English, which are alphabetic in nature, words are often separated by blanks. Tokenizers simply have to replace whitespaces with word boundaries and remove other delimiters at both ends of a word such as punctuation marks, parentheses, and quotation marks to provide accurate results. Tokenization however is not as simple as using delimiters to identify the tokens because there still exist ambiguities that arise from that method of tokenization. Examples of such are multi word expressions like idioms, names, and dates, dehyphenation, missing white spaces and ambiguity of periods (Webster, 1992).

OpenNLP tokenizer segments an input character into tokens. This is being implemented using three types of tokenizers, whitespace tokenizer, simple tokenizer, and learnable tokenizer. The whitespace tokenizer identifies non-whitespace sequences as tokens. The simple tokenizer identifies sequences of the same charcter class as tokens. The learnable tokenizer detects token boundaries based on a maximum entropy probability model (The Apache Software Foundation, n.d.).

Example:

Input: Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.

The tokens would be “Pierre”, “Vinken”, “,”, “61”, “years”, “old”, “will”, “join”, “the”, “board”, “as”, “a”, “nonexecutive”, “director”, “Nov.”, “29”, “.”.

The tokenizer outputs either an array containing the words tokenized in this form “A”, “sample”, “input”, or an array that contains each offsets of the tokens in an input string.

Also, the tokenizer offers an option to view the probabilities for the detected tokens using getTokenProabilities. The users are also able to train data by the method TokenizerMe.train.

3.2.3. Part-of-Speech Tagging

Part-of-Speech (POS) Tagging is the process of assigning each word or token of the text to a corresponding part of speech (Marquez et al, 2000). POS gives information about the role of a word in its context and its inflections. It also adds the information contained within words by explicitly indicating some of the structure inherent in language. The POS tagger uses a set of tags to determine its corresponding part of

speech (Brants, 2000). Although in the traditional English grammar there are only 9 parts of speech namely noun, verb, article, adjective, preposition, pronoun, adverb, conjunction, and interjection, it is not uncommon in POS tagging to distinguish up to 50 or more tags. A list of tags used by the Penn Treebank Project can be seen in Appendix A. A simplified version of this process can be compared to the identification nouns, verbs and adjectives taught to schoolchildren. OpenNLP POS tagger is a tagging tool that designates tokens with their corresponding part-of-speech based on the meaning and context of the token (The Apache Software Foundation, n.d.). It uses a probability model in predicting the corresponding tags for the word. It only accepts tokens as input so preprocessing of text is required.

The tool provides APIs that can be easily integrated to an application. It provides APIs for both part-of-speech tagging and model training. To use the API, the POS model must first be loaded into memory from a disk or from another source. After that, the class is now readily available for new instances.

POSModel – A model needed to instantiate a POSTagger.

POSTaggerME – A class that instantiates a new POSTagger.

The tagging process requires an input array of strings, which represents a token in each string object. The output will be an array that contains the corresponding tags found at the same index as the token in the input array. The user can also see the breakdown of confidence scores for the returned tags by calling POSTaggerME.probs(); method from the POSTaggerME object.

OpenNLP also offers APIs for training the tool. With this, three requirements are needed. First, the application must open a sample data stream, which includes details on where the sample file is found. Next POSTagger.train method is called to start training. The output will be a model file, which in turn can be used as a model in the tagging process. The input training data should follow the syntax below.

About_IN 10_CD Euro_NNP ,_, I_PRP reckon_VBP ._.

That_DT sounds_VBZ good_JJ ._.

Each line represents one sentence and each word follows the syntax (Word)_(POS). The token and tag pairs are combined with the symbol “_”.

3.2.4. Named Entity Recognition

Named entity recognition (NER) identifies words or phrases in a text that refer such as names of people, organizations, locations, expressions of times, quantities, monetary values, percentages, etc and classifies them in predefined categories (people, place, location, etc.).

Recognizing named entities can be done in multiple ways. A manual approach of using a list of named entities which the system can search for instances of the entities in the given text is usually done however this approach is not only time-consuming but it is prone to errors as well (Kosseim & Thiery, 2011). This is because of the effort of adding all companies, organizations and names of persons that may appear in the text. Dictionary matching or pattern matching with regular expressions or other ad hoc methods can be used to recognize named entities but working on a formal grammar that

can obtain good results is a long process which is worth a long time of working with experienced computational linguists.

Training statistical Models can be used for more advanced entity extraction. This is done by manually labeling a training data, which is similar to the data that the system will process, of all entities of interest and their types. This is because of a system that is trained to run on a formatted document is used on free text such as blogs, the performance of the system will degrade. Hidden Markov Models is an example of a machine learning technique that is often implemented in named entity recognition.

The goal of a named entity recognition system is to take an unannotated block of text, such as this one:

Andrew visited 10 sites on his vacation to Mexico in 2010.

And produce an annotated block of text, such as this one:

*<person>Andrew</person> visited <quantity>10</quantity> sites on his vacation to
<place>Acme Corp.</place> in <date>2010</date>.*

LingPipe's named entity recognizer is a tool created to find mentions of specified things in running text. It tries to recognize an entity based on the domain of the trained data (Alias-I, 2003). For example in a news article, common entities are name of people, location, time, and organizations. Provided an input news data, the system will try to find and categorize words based on what entity they are. They aim to find the entities above (people, location, time, and organization) in the running text.

The tool depends on the training data that will be fed to it. The ready model of LingPipe involves the supervised training of a statistical model and other methods including dictionary matching and regular expression matching. When creating a statistical model, the training data must be labeled with all the entities and their types. Also, the user should be strict on the domain of its training. For example, the tool is trained to recognize news articles and its entities. If suddenly the tool is used on blogs and other informal articles, the accuracy may come low since the style of formalization will be different.

The tool follows a rule-based named entity detection. It tries to match certain regular expression that captures the intended pattern of entities. For example, we try to write a regular expression for an email address. We will find that a well-formed email address consists of two parts, the local part and the domain name separated by the "@" symbol.

Local->abc123@xyz.com<-Domain

The RE for example that will be formed is as follows:

`[A-Za-z0-9]([_\.\\-]?[a-zA-Z0-9]+)*@([A-Za-z0-9]+)([_\.\\-]?[a-zA-Z0-9]+)*\.[A-Za-z]{2,}`

The tool then will use this expression to pattern match a text in an article. If the expression matches what is find, then it will recognize it as an email address.

3.2.5. Semantic Tagging

Semantic tagging is the annotation of each content word with a semantic category. Semantic tagging classifies words based on their intensions and meanings in the

sentence. POS tagging may sometimes be considered semantic but is usually seen as syntactic tagging.

Semantic tagging can be classified into 5 different categories namely: sense-tagging, feature tagging, compositional semantic representation, dialogue act-tagging and document tagging.

Sense tagging is the process of tagging words in a text with its corresponding sense to solve the issue of word sense ambiguity. Most words are ambiguous to some degree. These words are called polysemous words. They are words with the same spelling but differ in meaning based on its context in a sentence. . An example where sense-tagging could be used is with the word “support”. “Support” in these sentences: “My support goes to the bulls” and “The support of the building is strong”, is not used in the same sense.

Feature tagging is using more abstract categories or features for tagging. It is used to identify semantic relations between lexicalised words (Ekeklint, 2001).

Compositional semantic representation is one of the most common methods in representing the semantics of sentences. It is the process of tagging an entity or a token reflecting their compositional properties. Dialogue act-tagging is the tagging of the dialogue act in the series of dialogue. It tags them according to which type of classification it belongs. An example of a classification could be sorted among the types of sentence that the dialogue act fits.

Document-tagging is classifying the document into a category such as business and sports.

3.2.6. Coreference Resolution

Coreference resolution is the process in which noun phrases referring to a same real-world entity are identified. For acronyms, the usual approach would be taking the initials of multi-word names; for pronouns, their dependence on a noun is being backtracked by finding an “earlier term occurrence”.

Ruslan Mitkov Algorithm

The Ruslan Mitkov algorithm is an anaphoric resolution algorithm that only requires little knowledge base. This is unlike most approaches to anaphora resolution wherein algorithms heavily rely on a lot of linguistic and domain knowledge making the process time consuming. The Ruslan Mitkov approach makes use of preprocessed inputs (part-of-speech and noun-phrase levels) instead of parsing on its own. Candidates for the resolution are located and are assigned scores. Scores are based on the comparison of the input against noun phrase rules and antecedent indicators. The candidate with the highest score is selected as the antecedent. In the case of a tie, the candidate to be used is determined by immediate reference, collocation patterns, indicating verbs, or recent occurrence (Mitkov, 1998).

3.2.7. Pattern Matching in Strings

Pattern matching is a technique in data analysis in which a group of characteristic properties that defines a pattern is compared to the properties of a set of characteristics of an object or text to identify if the patterns of both objects are the same (Lang, 2001).

This technique is commonly applicable to information retrieval systems that answers questions about a large text collection by using a pattern matching language to identify syntactic relations and other information about the text. Most of the systems, especially information retrieval and extraction, require search engines however this technology is currently limited mostly into keyword searches. With pattern based indexing, it can improve this sort of system not only by keywords, but also syntactic relations of the objects of the text (Katz et al, n.d).

The idea of Boyer Moore algorithm is to compare the pattern with the text reading from right to left. If in the case that the symbol being compared with the pattern doesn't match at all, the whole pattern can be shifted by n positions after the compared text. n can be computed using two pre-computed functions to shift the pattern to the right. These two are Bad Character Heuristics and Good Suffix Heuristics.

Bad Character Heuristics

This heuristic can be applied if the text symbol causing a mismatch occurs somewhere in the pattern. If this occurs, the latest occurrence of the mismatched text can be aligned to this text symbol.

Illustration:

Table 3.1: Bad Character Heuristics

0	1	2	3	4	5	6	7	8	9
A	B	B	A	B	A	B	A	C	B
B	A	B	A	C					
		B	A	B	A	C			

In the given sample, "B" in the text being searched on causes a mismatch with the "C" in the pattern. Since "B" occurs somewhere in the pattern, the latest occurrence of "B" in the pattern will be aligned with the mismatched pattern. In this case both "B"s that occur in the pattern is found at index 0 and 2. Since the rightmost occurrence of "B" in the pattern is two, it will then be aligned to the text symbol's "B."

Bad Suffix Heuristics

Bad character heuristics is not applicable at all times. Sometimes the heuristic can cause a negative shift for the pattern.

Table 3.2: Failed Bad Character Heuristics

0	1	2	3	4	5	6	7	8	9
A	B	A	A	B	A	B	A	C	B
C	A	B	A	B					

In this example, the text symbol's "A" conflicted with the pattern's "B" on index 2. If bad heuristics is applied, the latest occurrence of "A" in the pattern is at index 3. If it is aligned with the mismatched text symbol, it will cause a negative shift to the left. This is where good suffix heuristics is applied. This heuristics computes the maximum possible shift distance based on the structure of the pattern

Table 3.3: Good Suffix Heuristics

0	1	2	3	4	5	6	7	8	9
A	B	A	A	B	A	B	A	C	B
C	A	B	A	B					
		C	A	B	A	B			

In this example, the text symbol's "A" conflicted with the pattern's "B" on index 2. Since we know that the combined symbols "AB" of index 3 and 4 matches with the pattern, the pattern can be shifted until the next occurrence of "AB" in the pattern is aligned to the text. The system will always compute for both the bad character and good suffix

3.2.8. Lexicon

The lexicon contains all the important terms and keywords in the system's domain. The gazetteer is also included in the lexicon since it also contains important terms in the domain. As for database of terms, there are available resources for this, such as WordNet, Gazetteer, and the JAPE Rules.

Wordnet (Princeton, 2011)

WordNet is an open source system that contains a large lexical database of the English language containing nouns, verbs, adjectives, and adverbs. These parts of speech are grouped into sets of cognitive synonyms called synsets, each expressing a distinct concept. These synsets are interlinked with one another by means of conceptual-semantic and lexical relations. The result of these links is a network of meaningfully related words and concepts that is easily navigated through the interfaced browser.

Most of the time, WordNet is mistakenly thought as a thesaurus. However, there are some important distinctions between the two. First, WordNet interlinks each synsets not just according to word forms or string of letters, but through the specific senses of words. It finds words with close proximity to one another in terms of their lexical meaning and is semantically disambiguated. Secondly, semantic relations among words are properly labeled following patterns whereas in Thesaurus, the grouping of words is not explicitly labeled.

The main relation among words is defined by their synonymy. Synonyms are words that denote the same concept and are interchangeable in many contexts. Using this concept, words are grouped to form synsets. Each of the 117,000 available synsets in WordNet is linked by means of a small number of conceptual relations. These synsets usually contains one or more sentences ilustrasion the use of the synset members.

Most words that construct the network relations connect words from the same part of speech (POS). WordNet consists of four sub-nets to categorize words based on their part of speech, namely nouns, verbs, adjectives, and adverbs, with few cross-POS pointers. Cross-POS relations include the morphosemantic links that distinguishes similar words sharing the same root word. Observe (verb), observant (adjective),

observer (noun) is a sample of a morphosemantic link which makes WordNet more powerful.

ANNIE Gazetteer

A gazetteer consists of a set of lists containing names of entities. These lists are used to find occurrences of these entities in a text. An example of this is the ANNIE Gazetteer.

Ecu
European Currency Units
FFr
Fr
German mark
German marks
New Taiwan dollar
New Taiwan dollars
NT dollar
NT dollars

An individual gazetteer list is a plain text file with one entry per line. This list can be modified using any text editor. There is also the index file with is used to describe all the gazetteer list files that belong together.

JAPE Rules (Lakin et al, 2009)

In JAPE, rule specification language is made up of two vital parts. A Left Hand Side (LHS) and a Right Hand Side (RHS). The LHS is where the pattern that makes up the rule is located. The RHS is where the manipulation of the annotation is placed. This is also where calls to external references or Java Codes can be placed. The LHS and the RHS is separated with a string "-->" the text that precedes the string for the current rule is part of the LHS. The text that follows the string is the RHS. A simple example is given below:

```
1      Phase: Jobtitle
2      Input: Lookup
3      Options: control = appelt debug = true
4
5      Rule: Jobtitle1
6      (
7      {Lookup.majorType == jobtitle}
8      (
9      {Lookup.majorType == jobtitle}
10     )?
11     ): jobtitle
```

```

12      -->
13      :jobtitle.JobTitle = {rule = "JobTitle1"}

```

In the example, Lines 5-11 are part of the LHS and Line 13 is the RHS.

The LHS of the rule is mainly used as the pattern that the engine will try to identify from the text; this implies that the LHS is capable of making simple comparisons between the text or its annotation with keywords or input. The LHS is capable of matching a string or a value to the annotated text; this follows the format {Token.string == "string"}. The LHS is also capable of using Meta-properties of the annotated text. Meta-properties supported by default are length, string, and cleanString. Meta-properties are used with the following syntax {X@length > 5} where X is an annotation type name. The LHS is also capable of using templates. Templates are strings that can be reused throughout the rule set. These strings can have parts where a text can be inserted. These templates follow the syntax: Template: url = "http://gate.ac.uk/\${path}" and placed before the rule is declared. The template is accessed using the syntax: [X Y = "string"] where X is the template name (in the example syntax: url) Y is a template parameter (in the example syntax: Y). The LHS also supports macros. These macros enable the rules to be bound to a name which can be accessed on the LHS of rules. Macros follow the syntax Macro: X(Y) where X is the macro name and Y is the pattern. Macros can be used in the LHS of the rule by calling the name enclosed in a parenthesis. The LHS of the rules is also capable of accessing context. Context can be from tokenization, gazetteer, or from other module in preprocessing. In the case of the gazetteer, it annotates the tokens with Lookup with some feature set. So accessing the feature set majorType and comparing it to a string follows the syntax {Lookup.majorType = "string"}. The LHS is also capable of negation. In this case rather than identifying if it has a pattern, in the negated state, it identifies if it lacks a pattern. In the LHS Regular expressions can also be placed like *, +, ?, and |.

The RHS is used for manipulating the annotated feature set. The RHS is started off with ":" followed by the base that links it with the LHS (in the example: jobtitle) followed by a '.' then its annotation type, following this is the feature set of the current annotation. The values from these feature sets can be a static string or can be derived from the LHS. In the latter case, the value is accessed by using the format :base.type.feature. In the example, Lookup majorType is accessed by :jobtitle.Lookup.majorType. In the RHS Java codes can also be placed, in this case, the code is placed within a parenthesis right after the "-->" symbol. A Java code and a regular RHS annotation manipulation

PlantPhos Windowing Technique (Lee et al, 2011)

Windowing is a technique that creates rules through recognizing possible patterns around a target object. The target object maybe in the form of words, elements, or anything as long as it can be represented and is part of a series, text, or sequences. Windowing makes use of windows that are presented as a numerical value. Windows are the extent of the pattern that rules are being built around of.

The cat in the mat is sleeping.

Window size = 5 Center = mat Left = 2 Right = 2

The example above has a pattern with window size 5. With the size of the window, center (mat) is set as the target object to determine, which takes up one size of the window. With 4 slots left remaining in the window, it is determined whether the pattern should come from the right hand side, or left hand side of the center. In the sample above, the remaining window slots are allocated equally to the left and right of the target object. So from the center, 2 sequences at the left ("in", "the") and right ("is", "sleeping") are

treated as candidates to be a rule.

<String:"in"><String:"the"><NER:location><String:"is"><String:"sleeping">

<String:"in"><String:"the"><NER:location>

<NER:location><String:"is"><String:"sleeping">

<String:"the"><NER:location><String:"is">

After getting all possible combinations, every rule then is tested through own means of the developers on whether the rule is effective or not.

3.3. Adaptive Information Extraction

Recent developments in Information Extraction Systems are geared towards to portability and adaptability to new situations. This is due to the potential application of the diverse, widely available, and rapidly growing online text sources in knowledge acquisition. Current IE systems are unable to utilize these sources because they are created only to handle specific knowledge and formats that may not appear in these new sources. Encountering documents of a new domain or style of formatting may lead to very inaccurate results. These problems prompted the research in applying machine learning techniques to IE systems to develop Adaptive IE systems.

Applying inductive learning methods in acquiring knowledge of a specific domain to create IE rules from training documents has been the focus of much of the research effort in this area. IE rules can be classified into two categories: single-slot or multislot. Given a template to be filled, single-slot rules are able to fill one slot with a template by extracting document fragment related to the slot while multislot rules are able to fill a set of slots by extracting multiple documents related to the slots (Turmo et al, 2006).

3.3.1. NEXUS (Atkinson et al, 2008)

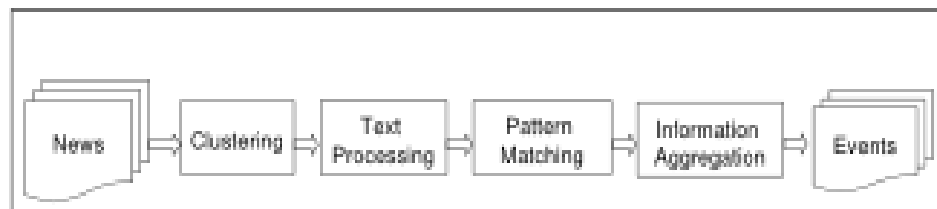


Figure 3.4: NEXUS' Architecture

News cluster Event eXtraction Using language Structures (NEXUS) is an event extraction system used to populate violent incident knowledge bases. The system automatically extracts security-related facts from online news articles. In order to facilitate real-time data collection and information extraction, the system's architecture has been designed in such a way that it performs quick and accurate event extraction from online news articles (see Figure 3.4). Before the event extraction process of the system occurs, news articles are first gathered by European Media Monitor, a media monitoring software. This software delivers news clusters for each topic gathered. From the pool of extracted articles, NEXUS selects security related events via keyword based heuristics. The documents in each cluster are then linguistically preprocessed and are composed of the following steps: sentence boundary disambiguation, named entity recognition, chunking, labeling of action words, and unnamed person groups.

After the preprocessing step, a set of hand-coded extraction rules are then applied by the pattern engine on each of the document within a cluster. Core templates are then learned from annotated data. Based on the resulting template, a bootstrapping machine learning approach is applied in order to guarantee better precision of the learning patterns.

Finally, they select only the patterns in which they have a local maximum of the context entropy. The set of pattern is expanded by synonyms and hyponyms from WordNet and by syntactic variants.

3.3.2. PROTEUS (Grishman & Yangerbar, 1998)

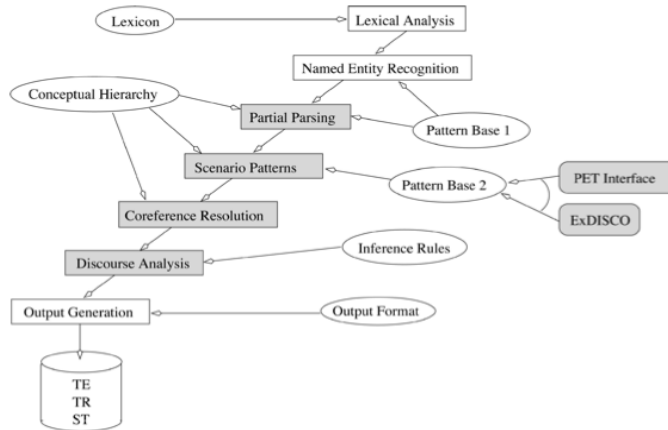


Figure 3.5: PROTEUS' Architecture

Cascaded pattern matching is the basis of PROTEUS' system architecture. The first module of the system is the Lexical Analysis module that tokenizes each sentence using a lexicon that consists of a general syntactic dictionary and lists of domain-specific words and POS tags the resulting tokens. The next module is the Named Entity Recognizer, which identifies named entities such as proper names using a set of rules (Pattern Base 1). The Partial Parsing module finds small syntactic phrases within sentences, such as basic NPs and VPs, and marks them with the semantic category of their class. This module also finds appositions, prepositional phrase attachments, and certain conjuncts by using special rules (Pattern Base 1) and creates logical form representations of the relations found between entities. The Scenario Patterns module applies rules, which create the logical forms related to events represented in the

clauses, for clausal identification (Pattern Base 2). This module is the only module that makes use of domain-dependent knowledge due to events most likely depending on information related to a specific domain. They make use of the PET Interface or the ExDISCO learning approach to create these rules because of the difficulty of creating these rules manually. The Coreference Resolution module links anaphoric expressions to their antecedent. Sequentially seeking the antecedent in the current sentence and its preceeding sentences until the antecedent is found is how this module links anaphoric expressions. The Discourse Analysis module uses a set of inference rules to build more complex event logical forms from those explicitly described in the document. Finally, the Output Generation module uses a set of rules (Output Format) to translate the resulting forms into a specific template structure.

ExDISCO

The ExDISCO (Yangarber & et al., 2000) is a bootstrapping method wherein extraction patterns in the form of subject-verb-object (SVO) are learned from an initial set of manually built SVO patterns. By applying these initial patterns upon a text, it will be able to determine if the particular text is suitable for extracting a target event or a part of it. When the set of seed patterns are applied, the unannotated corpus is divided into two categories, relevant and irrelevant. Exploratory search for patterns of SVO statistically correlated with the set of relevant texts facilitates the guessing of new extraction patterns that can be used in terms of searching for new relevant documents that can further be used by the system on the current domain. The resulting patterns are in the form of basic syntactic chunks semantically annotated (depending on their heads). However, a human expert is required to indicate which slots of the output template are to be filled by each learned pattern.

3.4. Information Extraction Evaluation Metrics

This section will cover the discussiona regarding the evaluation metrics that will be used in evaluating the information extraction module of the system.

3.4.1. Precision

Precision indicates how many of the extracted items are correct. The higher the precision, the better the accuracy of the system (Makhoul, 1999).

$$\text{Precision} = \frac{|\{\text{relevant information}\} \cap \{\text{retrieved information}\}|}{|\{\text{retrieved information}\}|}$$

3.4.2. Recall

Recall indicates the number of items that have been actually extracted over the number of items that should have been extracted. The higher the recall, the higher the chance for the system to extract information that needs to be extracted (Makhoul, 1999).

The formula of recall is:

$$\text{Recall} = \frac{|\{\text{relevant information}\} \cap \{\text{retrieved information}\}|}{|\{\text{relevant information}\}|}$$

3.4.3. F-measure

F-measure is a harmonic mean, which provides a weight to determine the bias towards recall or precision. It helps ensure that the information extraction module of the system would reach 100% in both precision and recall.

The formula of F-measure is:

$$F_{\alpha} = \frac{(1 + \alpha) * \text{precision} * \text{recall}}{(\alpha * \text{precision}) + \text{recall}}$$

3.5. Crisis Mapping

Crisis mapping is defined as live mapping focused on crises that may range from slow-burn crises to sudden-onset disasters (Meier, 2011). Three key pillars have been identified as integral to crisis mapping, each constituting an important area of research for the field (Meier, 2009b): (1) Crisis Map Sourcing (CMS), (2) Crisis Mapping Analysis (CMA), and (3) Crisis Mapping Response (CMR).

3.5.1. Crisis Map Sourcing

Crisis Map Sourcing is all about data point collection. It defines the methodology to be employed in terms of collecting information in such a way that the information collected adds significant value to a pool of data when mapped. Four principal methodologies in crisis map sourcing have been defined: (1) Crisis Map Coding, (2) Participatory Crisis Mapping, (3) Mobile Crisis Mapping, and (4) Automated Crisis Mapping. The common thread between the four methodologies is that they each look to extract event-data for crisis mapping purposes.

Crisis Map Coding (CMC) draws on hand-coding geo-referenced event-data. In this approach, coding event data is done manually and focus is placed on geo-referencing this data.

Participatory Crisis Mapping (PCM) is participatory mapping with a focus on crises. Focus groups are formed and are asked to identify and map local knowledge on threats and risks at a community level.

Mobile Crisis Mapping (MCM) makes use of mobile technology in gathering data for crisis mapping. In this approach, mobile phones, geospatial technologies and unmanned aerial vehicles (UAV) are being used.

Automated Crisis Mapping (ACM) leverages natural language processing technologies and computational linguistics to extract event-data. In this approach, real-time and automated information collection mechanisms using natural language processing have been developed for the automated and dynamic mapping of disaster and health-related events.

3.5.2. Crisis Mapping Analysis

Crisis Mapping Analysis defines the methodology employed in order to analyze crisis mapping data to identify patterns over space and time. It entails the application of statistical techniques to spatial data for pattern or “signature” detection in real-time. Its

purpose is to provide in-the-moment decision-support to users of a given Crisis Mapping platform. Three principle approaches in crisis mapping analysis have been identified: (1) Crisis Mapping Visualization, (2) Crisis Mapping Analytics, and (3) Crisis Map Modeling.

Crisis Mapping Visualization (CMV) deals with rendering collected information on a dynamic, interactive map in such a way that rendering provides maximum insight on the data collected and any potential visual patterns.

Crisis Mapping Analytics (CMA) is GIS analysis applied to crisis data. This approach draws on applied geo-statistics and spatial econometrics to identify crisis patterns otherwise hidden to the human eye.

Crisis Map Modeling (CMM) combines GIS analysis with agent-based modelling. Its main purpose is to use empirical data to stimulate different scenarios using agent-based models.

3.5.3. Crisis Mapping Response

Crisis Mapping Response deals with the strategic and/or operational response that can be generated based on the information collected and represented on the chosen form of visualization. It has three principle components: (1) Crisis Map Dissemination, (2) Crisis Map Decision Support, and (3) Crisis Map Monitoring and Evaluation.

Crisis Map Dissemination (CMD) aims to disseminate the information provided by the crisis maps to entities that can provide assistance and response to the people affected.

Crisis Map Decision Support (CMDs) makes use of decision-support tools specifically for crisis mapping response. It entails the use of interactive mapping platforms that users can employ to query crisis data. Its main purpose is to identify patterns in order to amplify, mitigate or change them.

Crisis Map Monitoring and Evaluation (CMME) combines crisis mapping with monitoring and evaluation in order to produce basemaps, baselines mapped in space and time. It aims to identify project impact or lack thereof by comparing basemaps with new data being collected throughout the project cycle.

3.6. Mapping Tools

This section contains discussions on mapping tools that can potentially be used in the development of the crisis mapping software.

3.6.1. Google Maps

Google Maps is a map service offering powerful, user-friendly, and interactive mapping technology that can be accessed via a web browser. Viewers of the map can instantly view its adjacent sections with a simple click and drag. Zooming in and out of the map is also possible. Google Maps also makes use of satellite imagery, providing users with the option of viewing the map as a satellite image (bird's eye view of a location) instead of the traditional map view. Users can also view and navigate the map within street-level imagery. Aside from these, it is also integrated with local business information including business locations, contact information, and driving directions. Users have the capability of inputting search queries, and the Google service automatically plots locations that are related to the query. Alongside this, basic information regarding each plotted location such as address, contact numbers, and brief descriptions are also shown. Aside from

locating places with queries, users may also opt to find places by inputting its longitude and latitude coordinates.

Google Maps compiles information from a variety of sources. It combines information from web search results, as well as data submitted directly by business owners, and third-party sources such as publicly available Yellow Pages directories.

Google has provided a wide array of APIs that let website owners embed Google Maps into one's websites and applications. In addition to this, they are also given the capability of overlaying data on top of these maps. The Maps API is a free service and is available for use by consumers who agree to abide with Google's terms of service.

One of the APIs comprising the Maps API is the Maps JavaScript API. This API allows webpage owners to embed and manipulate Google Maps in their webpage with the use of JavaScript. It provides a number of utilities for manipulating and adding user-defined content to the map through a variety of services.

3.6.2. Open Street Map

OpenStreetMap is a free and editable web-based map of the whole world. Data used in the map were gathered with collaborative crowdsourcing. It provides free geographic data such as street maps to anyone who needs them. The motivation behind the project was the legal and/or technical restrictions or copyright issues on geographic data on map services such as Google Maps, holding back people from using these data in creative, productive or unexpected ways.

OpenStreetMap populates its data by following how Wikipedia does, crowdsourcing. People gather location data from a variety of sources such as recordings from GPS devices, from free satellite imagery or simply from knowing an area, and upload this data to OpenStreetMap. The uploaded data can be further modified, corrected and enriched by anyone. And because OpenStreetMaps collects data this way and not from existing maps, they have all the rights to claim all information presented.

It must also be noted that there will always be issues in accuracy as information in OpenStreetMap come from unspecified and unspecialized groups or individuals. The wiki-style process of gathering data has no guarantee of accuracy and thus relies on the community of mappers to validate and correct these data.

Currently, there are "OpenStreetMapping" efforts in the Philippines. This started sometime in 2006 when an individual marked EDSA in Metro Manila via GPS traces. Since then, the mapping efforts continued to rapidly increase throughout the years. When Yahoo! Permitted OpenStreetMap to use their satellite imagery for tracing in December 2006, the mapping of Metro Manila, one of the areas covered exploded and remains to be the most mapped area in the Philippines approaching around 90% of its roads mapped.

3.6.3. OpenLayers

OpenLayers, developed by OpenGeo, is a library written purely in JavaScript used for displaying maps in a web browser. It has been developed to further the use of geographic information of all kinds. It accesses data through industry standards, making it free of server-side dependencies. It is highly extensible and is used as the foundation of all of OpenGeo's mapping interfaces. It allows the display of map tiles and markers loaded from any source. It can overlay multiple standards-compliant map layers into a

single application. It allows vector feature rendering and styling and web-based editing, including feature snapping and splitting. And lastly, it is pluggable with any JavaScript toolkit such as JQuery.

As a framework, OpenLayers is intended to separate map tools from map data so that all the tools can operate on all the data sources. This separation breaks the proprietary silos that earlier GIS revolutions have taught civilization to avoid. The mapping revolution on the public Web should benefit from the experience of history.

3.7. Crowd Sourcing

This section provides discussions on the ideas surrounding crowdsourcing. The definition and basic concepts of the said term will be discussed, as well as its different types of implementation.

Crowdsourcing is a blend of two words: “crowd” and “outsourcing” (Howe, 2008). The basic idea behind crowdsourcing is to tap into the collective intelligence of the public or a broad audience in order to complete tasks that would normally be done by a single person or a group of individuals. Crowdsourcing is an open call to an undefined group of people and because of this, it attracts those who are most fit to perform tasks, solve problems, and most importantly, contribute with the most relevant, creative and fresh thoughts. Its main purpose is to enable organizations to expand the size of their talent pool while also gaining a deeper insight into what external participants really want, free of charge.

Four main types of crowdsourcing have been identified (Howe, 2008): (1) crowd wisdom, (2) crowd creation, (3) crowd voting, and (4) crowd funding. These will be discussed in the following subsections.

3.7.1. Crowd Wisdom

Crowd wisdom or Wisdom of Crowds principle attempts to harness a broad audience’s knowledge in order to solve problems, predict future outcomes, or help direct corporate strategy. It is believed that a crowd, given the right set of conditions, will almost always outperform any number of employees even concentrated groups of highly specialized people and this claim is supported by studies made by Caltech professor Scott E. Page.

Crowd wisdom is further differentiated into three types: (1) prediction markets, (2) crowdcasting, and (3) idea jam.

Prediction Markets

Prediction markets function like stock markets, however, instead of shares people invest in possible outcomes. Many companies make use of prediction markets internally, allowing their employees to trade in outcomes such as inventory, sales goals, and manufacturing capacity, thereby crowdsourcing the decision making process. The possible outcomes are presented as stocks, and participants can buy and sell stocks according to their knowledge about the outcome. The outcome with the highest value is considered the most probable outcome. Prediction markets make use of large, diverse networks of people who are not experts, but often possess unique knowledge. Ideally, prediction markets do not represent a community as interaction between the participants could introduce bias to the outcome. An example of a prediction market applied in real world scenarios are elections wherein people choose or invest in a particular candidate

who they think best fits a specific position, and in return have the winning candidate's leadership become the voters' incentive.

Crowdcasting

In crowdcasting, an individual with a problem broadcasts it to the widest possible audience in the hope that someone will solve it. Since an open call is made to a large and diverse crowd, the chances of finding a solution are much better than with more traditional approaches. An example of crowdcasting is InnoCentive. It is a platform that opens up scientific problems to a crowd of about 160,000 people who are interested in science and compete for a prize.

Idea Jam

Idea jams are basically massive, online brainstorming sessions. It is similar to crowdcasting, with the only major difference being that the call for submissions in an idea jam is much more open-ended. Instead of attempting to solve a particular problem, solutions are created for problems that do not yet exist, by allowing the crowd to discuss whatever topics they are interested in. Generally, a forum is provided where people can discuss current products and provide ideas for future products.

3.7.2. Crowd Creation

Crowd creation is defined as the crowdsourcing of activities that make use of the creative energy of participants. It is similar to user generated content, although crowdsourcing generally involves building a business around it. As opposed to crowdcasting and prediction markets, interaction between participants seems crucial, as crowdsourcing creative work generally involves a tight community with a deep commitment to the task and each other. Financial incentives are also generally absent. The community aspect also increases the quality of the work, as participants are striving to enhance their reputation.

3.7.3. Crowd Voting

Crowd voting leverages a community's judgment to organize, filter, and stack-rank content. In this approach, the crowd is given an opportunity to express their opinion through voting or rating. Information gathered through this method can be used by an organization for decision making. It is classified as the most popular form of crowdsourcing, which generates the highest levels of participation. The 1:10:89 rule is applied to this approach which states that out of 100 people, 1 will create something valuable, 10 will vote and rate submissions, and 89 will consume creation. In using crowd voting, identifying the most relevant content based on its popularity with the crowd becomes rather easy.

An example of crowd voting would be reality TV shows wherein people vote for their favorite contestants. Threadless is another great example of a company that makes use of crowd voting as they allow consumers to vote for T-shirts they want the company to manufacture and sell through the company's website.

3.7.4. Crowd Funding

Crowd Funding circumvents the traditional corporate establishment to offer financing to individuals or groups that might otherwise be denied credit or opportunity. Two groups of typically under-funded populations include individuals in developing nations and

amateur musicians. Kiva, a microlending portal, offers an example of crowdsourcing. Kiva provides a marketplace for aspiring entrepreneurs in developing nations to seek out financing for projects that is not readily available in their home markets.

3.8. Validating Crowdsourced Information

One of the main risks in crowdsourcing is the authenticity and accuracy information gathered by using this method of data collection. This is due to the fact the crowds participating in these undertaking belong to undefined groups of people. Sometimes, people are just not equipped with the needed amount of information regarding a particular event and as a result, unknowingly publish inaccurate information. It is also unavoidable to have individuals who are keen on deliberately submitting fabricated data just to cause trouble. Validating crowdsourced information is crucial especially in disaster management as misleading information in this field often cost lives.

This section discusses some of the possible techniques that can be used in order to filter out most, if not all, inaccurate and misleading crowdsourced information.

3.8.1. Filtering

Filtering crowdsourced data can become means of validation (Meier, 2009c).

One way of filtering crowdsourced data is by comparing contents of reports in order to determine if they are exact replicas of another. This can be marked as suspicious behavior as it can indicate that someone may simply be copying and pasting the same report in order to trick the system. With this, fooling the system will become a bit more time consuming as someone committing this trickery must sent multiple reports with different wording in order to be successful.

Another way of filtering crowdsourced data is through comparing the source of similar reports. The greater the diversity of media used to report an event, the more likely that the event actually happened. For example, information extracted from 15 reports were identical, wherein five reports were submitted by webforms, nine were submitted by SMS, and six were submitted via Facebook, is more reliable than 15 identical reports being submitted through Twitter only. With this filter, people trying to exploit the system will have to send several emails, text messages, and etcetera using different means of describing a particular event.

Another way of filtering crowdsourced data is to identify the email addresses, IP addresses, and mobile phone numbers in question in order to determine if it is only coming from one person. With the implementation of this filter, people wishing to sabotage the system will have to send emails from different accounts and IP addresses, different mobile numbers, and so on.

Anything looking suspicious will be tagged for human review. The point of implementing these filters is to make exploiting the system as time consuming and as frustrating as possible.

3.8.2. Triangulation and Scoring

Assuming that the tagging process of the information extraction module has been finished, a highly structured event dataset will be obtained.

	Place	Day	Time	Actor(s)	Event	Source	Source Location
1	Girgaon, Mumbai	28-Nov	11:34	Terrorists	Attack	BBC	London
2	Girgaon, Mumbai	28-Nov	11:47	Terrorists	Bomb	Twitter	Dehli
3	Mumbai	28-Nov	11:48	Terrorists	Kill	Twitter	San Francisco
4	Girgaon, Mumbai	28-Nov	11:51	Terrorists	Attack	CNN	Dehli
5							
6							
7							
8							
9							
10							

Figure 3.6: Highly Structured Event Dataset (Ushahidi)

A simple machine analysis to cluster same event together can be applied and thereby combine duplicate event-data into one.

	Place	Day	Time	Actor(s)	Event	Sources	Source Location
1	Girgaon, Mumbai	28-Nov	11:34	Terrorists	Attack, Bomb, Kill	BBC, Twitter, CNN	London, Dehli, San Francisco
2							
3							
4							
5							

Figure 3.7: Clustered Events (Ushahidi)

A simple weighting or scoring schema to assign a reality score to determine the probability that the event reported really happened can now be applied in order to further validate the information. For example, an event that is reported by more than one source is more likely to have happened. This increases the reality score of the event above and pushes it higher up the list. One could also score an event by the geographical proximity of the source to the reported event, and so on. These scores could be combined to give an overall score.

3.8.3. Crowdsourcing

Because of the fact that validation and scoring algorithms can never be 100 per cent accurate, developers may use crowdsourcing as a means of external validation.

Bounded and unbounded crowdsourcing can be used in validating information externally. Bounded crowdsourcing is when those doing the crowdsourcing are trusted entities, meaning their input can be considered as accurate. However, bounded crowdsourcing is limited to a relatively small number of individuals. Unbounded crowdsourcing, on the other hand, is open crowdsourcing, meaning information provided by these individuals cannot be automatically tagged as trusted; however, it has the advantage of aggregating a vast amount of information from diverse sources. By combining bounded and unbounded crowdsourcing, there will be an increase in (1) overall reporting, and (2) in the ability to validate reports from unknown sources.

External validation can be done through implementing a voting mechanism that makes use of both bounded and unbounded crowdsourcing. Confirmation coming from bounded crowdsourcing is given more weight as compared to unbounded crowdsourcing.

If a member of the crowd confirms that a series of reports were indeed fabricated, the system should be able to automatically flag content coming all associated email

addresses, IP addresses, and/or mobile phone numbers. In other words, the system will start rating the credibility of users as well.

3.8.4. Ranking

Content sources may be given scores based on user behavior via a learned “trust” algorithm (Meier, 2009c). For example, when multiple sources are aggregated, and the content from one source is consistently deemed to be more accurate than content coming from other sources, scores are assigned to these sources. Content indicates a great deal regarding the source itself and trusted sources can be given priority.

3.9. Sources of Information

This section presents the proposed sources where the system will extract information.

The system will gather information from a number of online sources, particularly, local news sites, community blogs, web forms, and social networking sites, specifically, Facebook and Twitter. It will also be able to retrieve information through reports submitted by people through the use of web forms in the web site that comes with the system. In addition, it will also be able to gather information from SMS.

Information extracted from these resources will be the location of the event, the time of the event, the persons involved in the event, the type of event, and further details describing the event.

3.9.1. Official News Sites

Philippine news sites are a great source of verified information as articles posted here are official reports and thus are authenticated. The system will automatically gather news article from Philippine news sites with the use of web crawlers. Aside from mapping information extracted from the articles in a virtual map, it will also be categorized as verified information and will be used to cross check the validity of reports coming from unofficial sources.

3.9.2. Webforms

Webforms resemble papers that can be filled out by users. In essence, webforms will represent survey or report forms that are filled out and submitted through the Internet. The system’s website will allow both registered and unregistered users to submit reports or requests with the use of webforms. They will be asked to provide details regarding their report or request. Reports sent using webforms will be tagged for verification.

If the reporter is not a registered user, he will be asked for his name. He will also be asked to indicate the nature of his report. There are three natures of report: (1) disaster event report, (2) help request report, and (3) missing person report. In each of these reports, the reporter will be asked to provide the location and time of the event, and also further details describing it.

3.9.3. Social Networking Sites

Online social networking sites could solve many problems plaguing information dissemination and communications when disaster strikes, according to a report from US researchers in a recent issue of the International Journal of Emergency Management.

In the wake of natural disasters and terrorist activity, online services have become increasingly prominent as useful tools to get the news out faster than traditional media, to provide timely information sources, and even to re-connect people affected directly or indirectly as events unfold.

With the dominance of Facebook and the growing popularity of Twitter in the Philippines, the communities built by these two social networking sites have become great platforms for facilitating information sharing during disaster situations in the country.

Facebook

The system can track posts from fan pages of official news companies. Information gathered here will be categorized as trusted and verified as these come from authenticated sources which will be used in the validating information from unofficial sources. In some events, Facebook users also post reports in these fan pages. These will also be extracted and will be tagged for validation.

A dedicated fan page for the system can also be set up where in people can post reports and/or requests on its wall. All information gathered here will be subject to validation.

Information from Facebook posts will be extracted with the use of the Facebook API. This API is a platform for building applications that are available to the members of Facebook. It allows applications to use the social connections and profile information of users, subject to their privacy settings. The API uses RESTful protocol and responses are localized and in XML format.

Twitter

A dedicated account for the system can be set up and will be set to follow Twitter accounts of trusted sources such as news sites and government organizations. Information gathered from these sources will be categorized as trusted and verified for they come from authenticated sources and will be used in validating information from unofficial sources.

A predefined hashtag can also be provided and people will just need to include this on their tweets whenever they want to report an incident. All information gathered from this kind of tweets will be subject for verification.

Information from tweets will be extracted with the use of the Twitter's REST and Search API. Twitter REST API methods allow access core Twitter data. This includes update timelines, status data, and user information. The Search API methods allow interaction with Twitter Search and trends data. The API presently supports the following data formats: XML, JSON, and the RSS and Atom syndication formats, with some methods only accepting a subset of these formats.

3.9.4. SMS

In the event that reporters have no means of accessing the Internet, the flow of information to the system must not be compromised. Reporters will be able to send reports

via sending SMS to a predefined number dedicated to the system. A template must be followed for this type of report.

[Name of Reporter]<space>[Location]<space>[Event/Request/Missing]<space>[Details]

The time of the report will be the timestamp on the SMS upon receipt. All reports sent via SMS will be subject to validation unless they belong to official government or non-government organizations.

4.0. The SOMIDIA System

This chapter presents proposed system. It is divided into six sections. The first section discusses the system's overview. The second section outlines its objectives discussing the tasks that the system must accomplish. The third section describes its scope and limitations based on the outlined objectives. The fourth section provides an in-depth discussion of its architectural design. The fifth section presents the its front and back-end features. And lastly, the sixth section lists the different resources needed to use it.

4.1. System Overview

Social Media Monitoring for Disasters (SOMIDIA) is a crisis mapping system that focuses on plotting authentic crisis events on an interactive map in near real time. The system will be leveraging on crowdsourcing as its means of input. It will gather the information to be plotted by extracting information from different online sources, particularly news sites, blogs, webforms, and social networking sites, specifically Facebook and Twitter, and also from reports sent through SMS.

The system will automatically gather textual resources from online sources. The gathered resources will then be processed and will be extracted of relevant information. A map will then be plotted with information based on the extracted information.

Users may submit reports of disaster incidents or requests for help either by using webforms found on the system's website, by sending SMS to the system's dedicated number, or by posting information on dedicated pages of the social networking platforms Facebook and Twitter. The interactive map and list of reports will be presented registered users in such a way that the most relevant to their area of responsibility are given more emphasis. They may also tag reports they will be handling in order to avoid concentrating help in one place. Anyone who has access to the website will be able to view the interactive map and to browse information regarding submitted reports to learn of its current status. However, only registered users will be able to help in validating reports through means of a voting system by tagging a report as authentic or not.

4.2. System Objectives

This section presents the general and the specific objectives of the proposed system.

4.2.1. General Objective

To develop a crisis mapping system that is able to provide visualization of validated crisis events based on crowdsourced information in near real time.

4.2.2. Specific Objectives

The following are the specific objectives of the system.

1. To automatically retrieve textual resources from online sources;
2. To extract relevant information from these textual resources;
3. To validate the gathered information;
4. To score and rank the point of origin of the gathered information;
5. To determine the weights which will be used to quantify the credibility of extracted information;

6. To plot the incident reports in an interactive map in near real time;
7. To present the interactive map and list of reports in a manner most relevant to the user;
8. To facilitate external validation of submitted reports.

4.3. System Scope and Limitations

SOMIDIA is a crisis mapping system that focuses on plotting authentic crisis events on an interactive map in near real time.

The system will be deploying a number of crawlers to crawl the Internet every specific time interval in order to facilitate near real time information retrieval and extraction. Textual resources posted within the span of two hours before every deployment phase of the crawler will be retrieved.

The system will retrieve textual resources from online sources, such as official news sites and blogs, and official government and non-government organization sites and blogs. The system will also retrieve Facebook posts and Twitter tweets using the Facebook API and Twitter API, respectively. However, it must be noted that due to the privacy settings of Facebook and Twitter, the system will only be able to crawl and gather information from accounts and pages that are set to public. Facebook posts that will be retrieved are those found on the dedicated fanpage of the system as well as fanpages of official news sites and government and non-government organizations. For Twitter, all tweets containing a predefined hashtag will be retrieved by the system. Tweets on the system's dedicated account as well as on official news companies, and government and non-government organizations will be retrieved as well. In the case of the emergence of new possible sources, a module for adding these sources to the list of websites that will be crawled will be provided to system administrators.

The system will also be gathering information from SMS reports. A dedicated number for sending disaster reports to feed the system will be setup. Reporters will send their reports to this number with a predefined template. The text messages that will be received will automatically be forwarded to the system for processing. Taglish and text speak will be handled by the SMS processing module however accuracy of extraction will not be as high as extraction from English text.

Gathered resources retrieved from the information gathering process will then be tagged for extraction. Relevant information, particularly location, date, time, and type of the event described by the resources will be extracted. It will be assumed that all event reports gathered from Twitter, Facebook, and SMS will contain at least the city and barangay level locations. The system will be able to handle textual resources written in English and Filipino. Extraction for Filipino text will be limited to keyword based pattern matching because there are no stable tools as of yet that can handle the Filipino language. As a result, information extracted from English text will be tagged as more accurate as compared to information extracted from Filipino texts.

Both internal and external validation modules will be implemented in order to ensure the authenticity of gathered information.

As for internal validation, the extracted information will be validated by clustering reports that refer to the same event and scoring their credibility rating. Reports are considered referring to the same event if they have the same event type and location. Scores are computed based on the number of templates produced for the event and other factors

such as reliability of source, source type, and extraction. Sources of information will be given credibility scores based on the history of their actions as logged by the system. Reports coming from registered users will be tagged with credibility levels based on their credibility scores.

The system will have three types of users: (1) system administrators, (2) reporters, and (3) volunteers. System administrators are people assigned to maintain the system. Reporters, on the other hand, can be registered or anonymous users who send in reports of disaster activities. And lastly, volunteers are registered users who are part of official government and/or non-government organizations who can also send in reports regarding disasters and, in addition, can plot organization activities on the map.

A hierarchy in terms of credibility will be imposed upon these users. System administrators and volunteers are given highest credibility, followed by registered reporters. The least trusted will be anonymous reporters as the system has no means of tracking the behavior of these users.

Extracted events will be plotted on an interactive map as clusters of similar reports. A report for each event plotted will also be maintained. These information will be presented in a manner most relevant to the user, provided that he is logged in to the system's website. Relevance will be based on the registered location of the user.

A voting mechanism will be implemented in order to facilitate external validation. External validation will be imposed on submitted reports. Only registered members will be allowed to vote up or vote down reports. Results of this voting mechanism will also affect the credibility scores of the sources of the reports. Users will also be given the capability of reporting unreliable sources which will be reviewed by website administrators.

4.4. Architectural Design

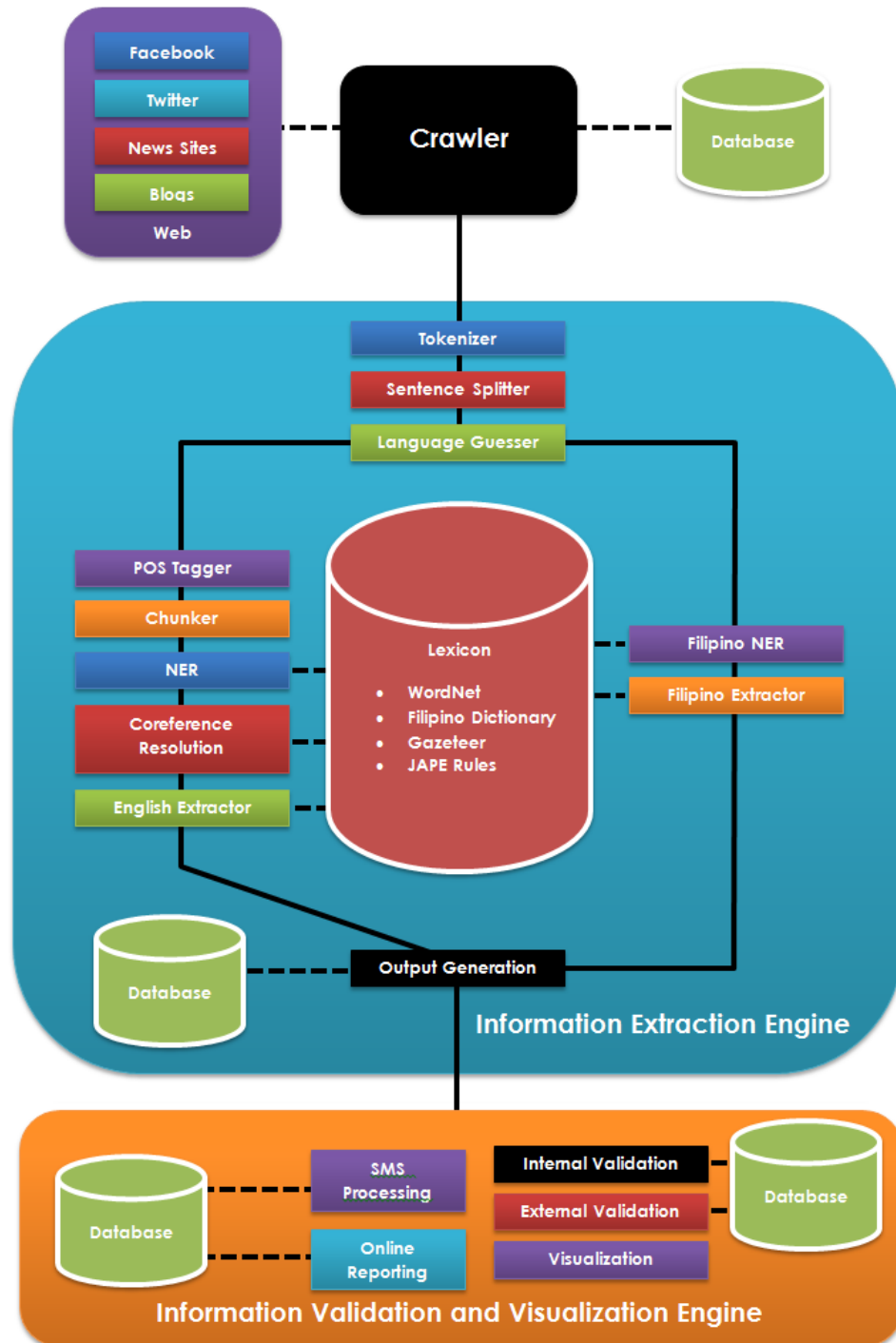


Figure 4.1: SOMIDIA Architectural Design

Web Crawler

This module would accept a list of hyperlinks of websites and would output a group of reports regarding disasters in the Philippines. It would use a keyword-based search on a specific list of trigger words that would identify an article as a relevant source of information and retrieve it for extraction. This module would also clean the data retrieved of noises such as images and mark-up languages for easier processing by the succeeding modules and output only the necessary information for extraction. It would include in the output the date and time of post, the source and source type of each text for computation of the credibility of the text. The retrieved reports would then be stored in the database for later use. This module would only crawl the Internet at specific times that are pre-determined. The plan is to make use of the opensource tool Crawler4J (Ganjisaffar, 2010), modifying it to suit the requirements of the research such as incorporating the Graph API and the Twitter API for retrieval of data from Facebook and Twitter. Figure 4.2 shows an example of retrieving an English post from Facebook while Figure 4.3 shows an example of retrieving a Filipino post from Twitter.


Input:	Output:
 Philippine Daily Inquirer (Official) Typhoon Pedring caused severe flooding in Makati City last night. Motorists are unable to pass through Ayala Avenue due to the flood it has caused. Like · Comment · Reshare · 11 hours ago via HootSuite · @	Source ID : 01 Source Type : Facebook Source : Inquirer Date posted : 08/07/11 Time posted : 9:21 A.M. Report: "Typhoon Pedring caused severe flooding in Makati City last night. Motorists are unable to pass through Ayala Avenue due to the flood it has caused."

Figure 4.2: Sample Input/Output of Crawler Module from Facebook

Input:	Output:
 Snapps03 Snapps Magpanay Maraming motorista ang nastranded sa Ayala Ave. Makati dahil sa bahang naidulot ng Bagyong Pedring. #baha #bagyo # 1 minuto ang nakalipas	Source ID: 002 Source Type: Twitter Source: Snapps03 Report: "Maraming motorista ang nastranded sa Ayala Ave. Makati dahil sa bahang naidulot ng Bagyong Pedring." Date posted: 08/07/11 Time posted: 9:30AM

Figure 4.3: Sample Input/Output of Crawler Module from Twitter

Tokenizer

The Unicode Tokenizer accepts as input the text from the crawler module that is stored in the database. It divides the document into simple tokens like numbers, punctuations and words. The tokenizer outputs either an array containing the words tokenized in this form "A", "sample", "input", or an array that contains each offsets of the tokens in an input string. The plan is to use OpenNLP's (The Apache Software Foundation, n.d.) tool for tokenizing. Figure 4.4 shows an example of tokenizing an English text while Figure 4.5 shows an example of tokenizing a Filipino text.

Input:	Output:
Source ID : 01 Source Type : Facebook Source : Inquirer Date posted : 08/07/11 Time posted : 9:21 A.M. Report: "Typhoon Pedring caused severe flooding in Makati City last night. Motorists are unable to pass through Ayala Avenue due to the flood it has caused."	<report = "001"> ["Typhoon" "Pedring" "caused" "severe" "flooding" "in" "Makati" "City" "last" "night" "." "Motorists" "are" "unable" "to" "pass" "through" "Ayala" "Avenue" "due" "to" "the" "flood" "it" "has" "caused" "."] </report>

Figure 4.4: Sample Input/Output of Tokenizing English Text

Input:	Output:
Source ID: 002 Source Type: Twitter Source: Snapps03 Report: "Maraming motorista ang nastranded sa Ayala Ave. Makati dahil sa bahang naidulot ng Bagyong Pedring." Date posted: 08/07/11 Time posted: 9:30AM	<report = "002"> ["Maraming" "motorista" "ang" "nastranded" "sa" "Ayala" "Ave." "Makati" "dahil" "sa" "bahang" "naidulot" "ng" "Bagyong" "Pedring" "."] </report>

Figure 4.5: Sample Input/Output of Tokenizing Filipino Text

Sentence Splitter

The Sentence Splitter module basically accepts as input the list of tokens and annotation lists as input. This module uses the list of abbreviations to distinguish a simple period from an abbreviation period and the list of tokens to form a sentence. The final output would be a list of sentences. For example, given the sentences "I have a dog. My dog's name is fluffy" the system will output "I have a dog" and "My dog's name is fluffy" as two different entities. The plan is to use OpenNLP's (The Apache Software Foundation, n.d.) tool for sentence detection. Figure 4.6 shows an example of sentence splitting a tokenized English text while Figure 4.5 shows an example of sentence splitting a tokenized Filipino text.

Input:	Output:
<report = "001"> ["Typhoon" "Pedring" "caused" "severe" "flooding" "in" "Makati" "City" "last" "night" "." "Motorists" "are" "unable" "to" "pass" "through" "Ayala" "Avenue" "due" "to" "the" "flood" "it" "has" "caused" "."] </report>	<report = "001"> [["Typhoon" "Pedring" "caused" "severe" "flooding" "in" "Makati" "City" "last" "night" "."] , ["Motorists" "are" "unable" "to" "pass" "through" "Ayala" "Avenue" "due" "to" "the" "flood" "it" "has" "caused" "."]] </report>

Figure 4.6: Sample Input/Output of Sentence Splitting English Text

Input:	Output:
Report ID: 002 Source Type: Twitter Source: Snapps03 Report: "Maraming motorista ang nastranded sa Ayala Ave. Makati dahil sa bahang naidulot ng Bagyong Pedring." Date posted: 08/07/11 Time posted: 9:30AM	<report = "002"> ["Maraming" "motorista" "ang" "nastranded" "sa" "Ayala" "Ave." "Makati" "dahil" "sa" "bahang" "naidulot" "ng" "Bagyong" "Pedring" "."] </report>

Figure 4.7: Sample Input/Output of Sentence Splitting Filipino Text

Language Guesser

This module would enable the system to differentiate the English and Filipino articles. This allows a much more specified processing of the document due to the difference between the two languages that may result to incorrect analysis of the text. Languages are detected using an approach based on frequency distribution of words from the language. It accepts an input of a tokenized document from the sentence splitter and would output the document with an added metadata of the language of the documents. It would pass English texts to the POS tagger while passing Filipino texts to the Filipino NER. Figure 4.7 shows an example of language guessing an English text while Figure 4.8 shows an example of language guessing a Filipino text.

Input:	Output:
<report = "001"> [["Typhoon" "Pedring" "caused" "severe" "flooding" "in" "Makati" "City" "last" "night" "."] , ["Motorists" "are" "unable" "to" "pass" "through" "Ayala" "Avenue" "due" "to" "the" "flood" "it" "has" "caused" "."]] </report>	<report = "001" " lang= "ENG"> [["Typhoon" "Pedring" "caused" "severe" "flooding" "in" "Makati" "City" "last" "night" "."] , ["Motorists" "are" "unable" "to" "pass" "through" "Ayala" "Avenue" "due" "to" "the" "flood" "it" "has" "caused" "."]] </report>

Figure 4.8: Sample Input/Output of Language Guessing an English Text

Input:	Output:
<report = "002"> ["Maraming" "motorista" "ang" "nastranded" "sa" "Ayala" "Ave." "Makati" "dahil" "sa" "bahang" "naidulot" "ng" "Bagyong" "Pedring" "."] </report>	<report = "002" " lang= "FIL"> ["Maraming" "motorista" "ang" "nastranded" "sa" "Ayala" "Ave." "Makati" "dahil" "sa" "bahang" "naidulot" "ng" "Bagyong" "Pedring" "."] </report>

Figure 4.9: Sample Input/Output of Language Guessing a Filipino Text

POS Tagger

The POS Tagger module accepts as input a list of English sentences from the language guesser. Processes in this module involve the tagging of tokens into its corresponding Part-of-speech, whether it's a noun, adjective, adverb, verb or others. The output will be a list of tokens each with their corresponding part-of-speech tag. The research is considering using OpenNLP's (The Apache Software Foundation, n.d.) tool for POS tagging. Figure 4.9 shows an example POS tagging an English text.

Input: <report = "001" lang="ENG"> [["Typhoon" "Pedring" "caused" "severe" "flooding" "in" "Makati" "City" "last" "night" "."] , ["Motorists" "are" "unable" "to" "pass" "through" "Ayala" "Avenue" "due" "to" "the" "flood" "it" "has" "caused" "."]] </report>	Output: <report = "001" lang="ENG"> [["Typhoon_NNP" "Pedring_NNP" "caused_VBN" "severe_JJ" "flooding_NN" "in_IN" "Makati_NNP" "City_NNP" "last_JJ" "night_NN" "."] , ["Motorists_NNS" "are_VBP" "unable_JJ" "to_TO" "pass_VB" "through_IN" "Ayala_NNP" "Avenue_NNP" "due_JJ" "to_TO" "the_DT" "flood_NN" "it_PRP" "has_VBZ" "caused_VBN" "."]] </report>
---	---

Figure 4.10: Sample Input/Output of POS Tagging an English Text

Chunker

The phrase chunker module accepts as input the set of tokens from the POS tagger and outputs grouped tokens according to their part of speech. This module groups words to form noun phrases and verb phrases. The plan is to use OpenNLP's (The Apache Software Foundation, n.d.) Noun and Verb chunker to determine noun phrases and verb phrases. Figure 4.10 shows an example chunking an English text.

Input: <report = "001" lang="ENG"> [["Typhoon_NNP" "Pedring_NNP" "caused_VBN" "severe_JJ" "flooding_NN" "in_IN" "Makati_NNP" "City_NNP" "last_JJ" "night_NN" "."] , ["Motorists_NNS" "are_VBP" "unable_JJ" "to_TO" "pass_VB" "through_IN" "Ayala_NNP" "Avenue_NNP" "due_JJ" "to_TO" "the_DT" "flood_NN" "it_PRP" "has_VBZ" "caused_VBN" "."]] </report>	Output: <report = "001" lang="ENG"> [[["Typhoon_NNP", "Pedring_NNP"]_NP, ["caused_VBN"]_VP, ["severe_JJ", "flooding_NN"]_NP, ["in_IN"]_PP, ["Makati_NNP", "City_NNP"]_NP, ["last_JJ", "night_NN"]_NP, "."] , ["Motorists_NNS"]_NP, ["are_VBP"]_VP, ["unable_JJ"]_ADJP, ["to_TO", "pass_VB"]_VP, ["through_IN"]_PP, ["Ayala_NNP", "Avenue_NNP"]_NP, ["due_JJ"]_PP, ["to_TO"]_NP, ["the_DT", "flood_NN"]_NP, ["it_PRP"]_NP, ["has_VBZ", "caused_VBN"]_VP, "."]] </report>
--	---

Figure 4.11: Sample Input/Output of Chunking an English Text

English Named Entity Recognizer

This module accepts as input the tokens outputted from the chunker module, focusing on the proper nouns. The plan is to use LingPipe's (Alias-I, 2003) named entity recognizer tool for the English NER due to its flexibility. Making use of a combination of the three approaches namely, dictionary-based, rule-based, and statistical-based is being considered. Basic named entities are names of persons, organizations, and locations, which are listed in a gazetteer. Figure 4.11 shows an example recognizing named entities in an English text.

Input: <report = "001" lang="ENG"> [[["Typhoon_NNP", "Pedring_NNP"]_NP, ["caused_VBN"]_VP, ["severe_JJ", "flooding_NN"]_NP, ["in_IN"]_PP, ["Makati_NNP", "City_NNP"]_NP, ["last_JJ", "night_NN"]_NP, "."] , ["Motorists_NNS"]_NP, ["are_VBP"]_VP, ["unable_JJ"]_ADJP, ["to_TO", "pass_VB"]_VP, ["through_IN"]_PP, ["Ayala_NNP", "Avenue_NNP"]_NP, ["due_JJ"]_PP, ["to_TO"]_NP, ["the_DT", "flood_NN"]_NP, ["it_PRP"]_NP, ["has_VBZ", "caused_VBN"]_VP, "."]] </report>	Output: <report = "001" lang="ENG"> [[["Typhoon_NNP", "Pedring_NNP"]_NP, <typhoon>, ["caused_VBN"]_VP, ["severe_JJ", "flooding_NN"]_NP, ["in_IN"]_PP, ["Makati_NNP", "City_NNP"]_NP, <location>, ["last_JJ", "night_NN"]_NP, "."] , ["Motorists_NNS"]_NP, ["are_VBP"]_VP, ["unable_JJ"]_ADJP, ["to_TO", "pass_VB"]_VP, ["through_IN"]_PP, ["Ayala_NNP", "Avenue_NNP"]_NP, <location>, ["due_JJ"]_PP, ["to_TO"]_NP, ["the_DT", "flood_NN"]_NP, ["it_PRP"]_NP, ["has_VBZ", "caused_VBN"]_VP, "."]] </report>
--	---

Figure 4.12: Sample Input/Output of NER in an English Text

Coreference Resolution

The input for the coreference resolution module will be the tokens annotated with their POS tags and entity labels. Noun counterparts of pronouns and classification of noun phrases will be resolved and substituted using the Russian Mitkov algorithm (Mitkov, 1998), and WordNet (Princeton, 2011) will be used as the lexicon. Normalization of tokens, such as standardizing date formats and collapsing similar sentences into one will also be done in this phase. The expected output will be the annotated document with the noun counterparts substituted to pronouns. Figure 4.12 shows an example of coreference resolution in an English text.

Input: <report = "001" lang="ENG"> [["Typhoon_NNP", "Pedring_NNP"]_NP_<typhoon>, ["caused_VBN"]_VP, ["severe_JJ", "flooding_NN"]_NP, ["in_IN"]_PP, ["Makati_NNP", "City_NNP"]_NP_<location>, ["last_JJ", "night_NN"]_NP, "._"] , ["Motorists_NNS"]_NP, ["are_VBP"]_VP, ["unable_JJ"]_ADJP, ["to_TO", "pass_VB"]_VP, ["through_IN"]_PP, ["Ayala_NNP", "Avenue_NNP"]_NP_<location>, ["due_JJ"]_PP, ["to_TO"]_NP, ["the_DT", "flood_NN"]_NP, ["it_PRP"]_NP, ["has_VBZ", "caused_VBN"]_VP, "._"]]</report>	Output: <report = "001" lang="ENG"> [["Typhoon_NNP", "Pedring_NNP"]_NP_<typhoon>, ["caused_VBN"]_VP, ["severe_JJ", "flooding_NN"]_NP, ["in_IN"]_PP, ["Makati_NNP", "City_NNP"]_NP_<location>, "7:00PM onwards, October 6, 2011"]_NP_<date/time>, "._"] , ["Motorists_NNS"]_NP, ["are_VBP"]_VP, ["unable_JJ"]_ADJP, ["to_TO", "pass_VB"]_VP, ["through_IN"]_PP, ["Ayala_NNP", "Avenue_NNP"]_NP_<location>, ["due_JJ"]_PP, ["to_TO"]_NP, ["the_DT", "flood_NN"]_NP, ["Typhoon_NNP", "Pedring_NNP"]_NP_<typhoon>, ["has_VBZ", "caused_VBN"]_VP, "._"]]</report>
---	--

Figure 4.13: Sample Input/Output of Coreference Resolution in an English Text

English Extractor

This module accepts the annotated document from the coreference resolution module and outputs single or multiple templates. The module is used for extracting the information from an annotated source. It is equipped with a set of JAPE rules (Lakin et al, 2009) used for extracting the information needed from the text. These rules are typically written with a left hand side, where the pattern that makes up the rule is located, and a right hand side where the manipulation of the annotation is placed from the left hand side. The patterns will be matched against the sentences and the ones that match will be added to the template (see Table 4.1). These rules are created using a two-tiered bootstrapping approach.

The bootstrapping algorithm begins with a small set of seed words or rules. These seeds are used to learn extraction patterns that reliably extract members of the same type. The learned extraction patterns are then used to generate new patterns, and the process repeats. The next level of bootstrapping retains only the most reliable patterns produced by the first bootstrapping process and then re-starts it from scratch. Using the output from Figure 4.11 as input, the output generated can be seen in Table 4.1.

Table 4.1: Partially-Filled Extraction Template from English Extraction

Report ID	
Source ID	
Event Type	Flood
Time	7:00PM - onwards
Location	Ayala Avenue, Makati City
Date Extracted	
Language	English

Filipino Named Entity Recognizer

This module accepts as input the tokenized Filipino sentences from the language guesser. The plan is to create their own NER because there is no existing Filipino NER tool. This module would be implemented using a combination of the dictionary-based and rule-based approach. Basic named entities are names of persons, organizations, and locations, which are listed in a gazetteer. Example of rules to be used: "sa" <location>, "si" <person>. Figure 4.12 shows an example of recognizing named entity in a Filipino text.

Input:	Output:
<pre><report = "002" lang = "FIL"> ["Maraming" "motorista" "ang" "nastranded" "sa" "Ayala" "Ave." "Makati" "dahil" "sa" "bahang" "naidulot" "ng" "Bagyong" "Pedring" "."] </report></pre>	<pre><report = "002" " lang= "FIL"> ["Maraming" "motorista" "ang" "nastranded" "sa" "Ayala" _<location> "Ave." "Makati" _</location> "dahil" "sa" "bahang" "naidulot" "ng" "Bagyong" _<typhoon> "Pedring" _</typhoon> "."] </report></pre>

Figure 4.14: Sample Input/Output of NER in a Filipino Text

Filipino Extractor

This module accepts Filipino texts from the Filipino NER and would output single or multiple templates for the output generation module. The input are assumed as tokenized and the location entities are tagged as <location> by the Filipino NER module. This module would first tag Filipino words that would pertain to an event such as "<event type = flood>baha </event>, <event type= earthquake> lindol </event>". The module would then make use of pre-defined rules that will extract the necessary information from the text. Example of such rules is <event> "sa" <location>. Using the output from Figure 4.12 as input, the output generated can be seen in Table 4.2.

Table 4.2: Partially-Filled Template from Filipino Extraction

Report ID	
Source ID	
Event Type	Flood
Time	7:00PM - onwards
Location	Ayala Ave, Makati City
Date Extracted	
Language	Filipino

Output Generation

This module accepts the annotated document and partially filled template from the extractor modules and outputs a report based on the template and its meta-data that will be stored in the database. The report consists of the report id, source id, event type, time, location, date extracted and language (see Table 4.3 and Table 4.4). All extracted information that is entered in the template is normalized to a specific format for input in the report. Missing information may exist after the extraction but all templates that do not have an event type and location would not be stored in the database. Using the output from the extractor modules (see Table 4.1 and Table 4.2), Table 4.3 shows the

output generated by English Extraction while Table 4.4 shows the output by Filipino Extraction.

Table 4.3: Filled Incident Report using English Extraction

Report ID	001
Source ID	001
Event Type	Flood
Time	7:00PM - onwards
Location	Ayala Avenue, Makati City
Date Extracted	08/08/11
Language	English

Table 4.4: Filled Incident Report using Filipino Extraction

Report ID	002
Source ID	002
Event Type	Flood
Time	7:00PM - onwards
Location	Ayala Avenue, Makati City
Date Extracted	08/08/11
Language	Filipino

SMS Processing Module

[Disaster Type]<Space>[Location]

Figure 4.15: Template of SMS Message

This module receives text messages from different users using a GSM/GPRS modem and outputs a report that contains necessary information for extraction. Messages that will be received will be processed to check if it follows the specified format (see Figure 4.13). Messages that follow the format will be extracted for information while messages that don't follow the format will be discarded. Both the original message and the generated report would be stored in the database. Figure 4.13 shows the format of the original message that will be stored in the database while Table 4.5 shows the generated report that will be stored in the database for validation.

Table 4.5: Filled Incident Report using SMS Extraction

Report ID	003
Source ID	003
Event Type	Flood
Time	7:32PM
Location	Ayala Avenue Makati City
Date Extracted	08/08/11
Language	SMS

Input:	Output:
	Source ID : 03
	Source Type : SMS
"DIA Flood LOC Ayala Ave. Makati DET severe flooding causes traffic"	Source : 09273200417
	Date posted : 08/06/11
	Time posted : 7:32 P.M.
	Report: "DIA Flood LOC Ayala Ave. Makati DET severe flooding causes traffic"

Figure 4.16: Sample Input/Output of SMS Processing

Internal Validation

This module clusters reports that refer to the same event and scores their credibility rating. It accepts as input reports stored in the database and outputs a group of events and their credibility rating and stores them in the database. Reports are considered referring to the same event if they have the same event type and location. Credibility scores are computed based on the number of templates for the event and other factors such as reliability of source, source type, extraction, etc. Once the credibility score reaches a certain threshold the credibility rating would be changed to "Credible".

External Validation

This module further verifies the credibility of reports with the use of a voting mechanism. Users can either vote up or vote down a particular report based on their knowledge of the incident. Only registered users will be able to make use of this module. Each vote will affect the credibility score of the report and its source.

Visualization

This module presents the reports in a way that will be relevant to the system's users. Information will be presented using both bulletins and maps. In the map view, the information will be plotted onto the map based on the location of the incident. The reports on the map can be viewed according to their event type. The bulletin consists of a list of reports showing the event type, location, date, time and credibility rating of the report. The sources and original texts of the reports can be seen when a report is selected in the bulletin. Both credible and not credible reports will be shown in the map and bulletins but credible reports will be shown priority.

4.5. System Functions

This section presents the functions of the proposed system. All functions found in this section are under the visualization module and accessed through the system's website.

Crisis Map Module

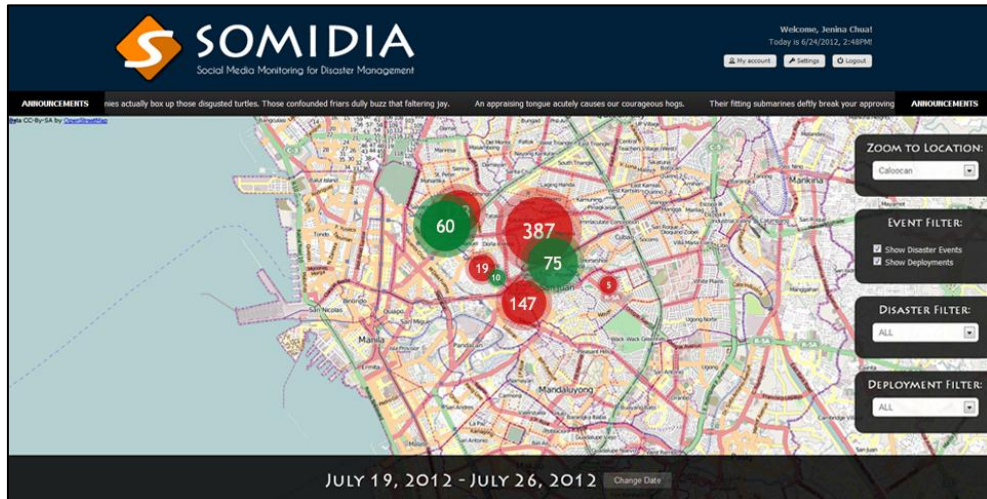


Figure 4.17: SOMIDIA Crisis Map

- View Crisis Map

This function allows anyone with access to the website to view the crisis map with plotted incident reports. Events that happen within a week from the date that the site has been accessed are initially plotted.

They may also see allow the visualization of actions of government and non-government organizations. These will also be shown in the form of clusters on the map but of different color as compared to disaster clusters to avoid confusion (disaster: red; actions: green). The viewer may also opt to show disasters only, actions only, or both clusters on the map.

- Filter Plotted Events

This function allows anyone with access to the website to view the map with only specific events plotted. He may opt to show disaster events only, action events only, or both clusters on the map. Furthermore, viewers may also filter the type of event they want to see.

- Change Date

This function allows anyone with access to the website to view the crisis map with only incidents that were plotted given a specific range of time. This is done by indicating the desired start and end dates.

Report Module

- Submit Report

This function allows anyone with access to the website to submit reports. There are 3 kinds of reports. A report can either be an incident report, a request for help, or a missing person report.

In submitting incident reports, the reporter will be asked to indicate the location of the event and the type of the event (e.g. flood, collapsed building). He will also be asked to provide a description of the event.

In submitting requests for help, the reporter will be asked to indicate the location where help is needed. He will also be asked to provide the type of help he needs (e.g. resource, assistance) and a description of the situation.

In submitting missing persons report, the reporter will be asked to indicate the name of the missing person. He will also be asked to provide the location, date, and time the missing person was last seen. The reporter will also have the option of providing a picture of the missing person.

If the reporter is an unregistered user, he has the option of indicating his name and contact details. This will be used in order contact the reporter if needed.

The screenshot shows the 'SUBMIT A REPORT' page of the SOMIDIA website. At the top, a navigation bar includes links for Home, My Account, Reports, Submit a Report, Directories, About, and Logout. Below this is a dark blue banner with the text 'DO YOU HAVE A DISASTER EVENT TO REPORT?' and a subtext: 'Help SOMIDIA create a bridge between disaster victims and responders by reporting an incident. By doing this simple act, you might be saving someone else's life.' The main heading is 'SUBMIT A REPORT'. The form is divided into two columns. The left column contains fields for 'Report Title' (a text input), 'Event Category' (a dropdown menu with 'Typhoon' selected), 'Date of Event' (a date picker showing January 1, 2012), 'Time of Event' (a time picker showing 00:00 AM), and 'Report Description' (a large text area). The right column is titled 'Find the location of the event:' and features a location dropdown menu with 'Caloocan' selected, a map of Caloocan, Philippines, and a 'Refine Location' section with an 'Input address...' field. At the bottom of the form are 'Submit Report' and 'Reset' buttons.

Figure 4.18: Submitting a report

- Browse Reports

This function allows anyone with access to the website to browse the list of submitted reports. Details found in the list view will be constrained to username, account type, source type, kind of report, content of the report, and credibility rating of the reporters.

BROWSE ALL REPORTED EVENTS

Here is a list of all documented reports so far. The reports are sorted by date and time in descending order. To filter the reports, feel free to use the filtering options on the left-hand side.

BROWSE REPORTS

Title:

Location:

Source Type:

☒ SOMEDIA
 ☒ Twitter
 ☒ Facebook
 ☒ News Media
 ☒ SMS

Source Name:

Disaster Type:

☐ Typhoon
 ☒ Flood
 ☐ Fire
 ☐ Earthquake
 ☐ Landslide
 ☐ Missing Person

EVENT TYPE	LOCATION	SOURCE	CREDIBILITY
Flood	Brgy. Sapalan, Makati	Twitter	High
Flood	Brgy. Sapalan, Makati	Twitter	High
Flood	Brgy. Sapalan, Makati	Twitter	High
Flood	Brgy. Sapalan, Makati	Twitter	High

Figure 4.19: Browsing and Filtering Reports

- View Report

This function allows visitors to view all the details of a selected report. They are also able to

- Filter Reports

This function allows anyone with access to the website to filter submitted reports. They may filter the list in terms of the kind of report, the location, the source type, and the credibility of the source.

- Verify Report

This function allows registered users to verify and influence the credibility of submitted reports through the means of a voting mechanism. Users may vote up or vote down a particular submitted report thus affecting the credibility score of the report's source. Votes coming from users living near the location of the event described in the report are given more value.

Account Module

- Create User Account

This function allows the creation of user accounts which will be used to access registered users only functions of the system's website. The person wishing to create an account will be asked to provide a username and a password which will be used in logging in to the website. He will also be asked to provide a few

personal details, particularly his name, address, email address and mobile phone number. A user's personal information will not be shown to the public. It will only be used to map submitted reports (e.g, a report submitted using a particular mobile phone number is mapped to a registered user with the same mobile phone number) and to assign credibility scores to a specific user. In addition to personal information, the user may also integrate his Twitter and/or Facebook accounts to his system account. This will also be used to map Twitter and Facebook posts to existing user accounts in the system. Lastly, he must also indicate the type of his account. An account can be a reporter account or a volunteer account. Reporter accounts are used by ordinary users, while volunteer accounts are used by members of official government and/or non-government organizations. Accounts tagged as organization will be tagged for approval by the website administrator.

- Login/Logout

This function allows registered users login to the website by providing the username and password of his account. He also specifies whether the account he is accessing is an organization or a user account. By logging in, the user is granted access to website functions available to registered users only. After finishing using the website, the user may logout.

- Edit Account

This function allows registered users to edit or update his user account details. The user may change all of his account details with the exception of his username.

Organization Module

- Add Organization

This function allows website administrators to add organizations to the organization registry. He will be asked to provide details regarding the organization. These information include the name of the organization, the location of its main office, its area of responsibility, its contact numbers, its official email address, URL of official website, Facebook pages and Twitter accounts, if applicable, and the name of the person in-charge.

- Membership

This function allows website administrators to add and delete members of particular organizations in the registry.

- Plan Deployment

This function allows organizations to map deployments on the crisis map (e.g. area their volunteers will be deployed). They may do this by indicating their area of action on a map, as well as supporting details such as the time and date of the deployment and specific information regarding the deployment.

Site visitors are able to view mapped deployments in the crisis map. Furthermore, they may browse the list of deployments, view details with regards to specific deployments, as well as filter deployments.

- Send Announcement

This function allows send out announcements to site visitors. Organizations announcements are posted at the default page of the site.

Site visitors are able to view announcements of organizations. Furthermore, they may browse the list of announcements, view details with regards to specific announcements, as well as filter announcements.

Directory Module

- View Directory

A directory of the different organizations and establishments that can prove to be helpful during emergency situations are compiled and made accessible to site visitors. The locations of these entities are plotted in a map. Viewers may filter the map to show specific types of establishments only. Viewers are also able to browse the directory in a list form wherein they can query for establishments based on available filters.

4.6. Physical Environment and Resources

This section outlines the minimum software and hardware requirements of the system.

4.6.1. Minimum Software Requirements

- Windows XP
- MySQL
- ADODB database abstraction library
- PHP 4.0
- Apache Server

4.6.2. Minimum Hardware Requirements

- 2 GB RAM
- High-speed Internet connection
- Server

5.0 Design and Implementation Issues

This chapter discusses the design and implementation that have been followed, as well as, the issues encountered during the implementation of the system.

5.1 Resource Gathering

This subchapter would be explaining how the research gathered the necessary data from different sources of information.

5.1.1 Trigger Words

Trigger words is a list of words that are used by the crawler in order to determine whether a certain text article is written in English or Filipino and also whether it is disaster related or not. The list of trigger words is composed of 89 English and 221 Filipino words (See Appendix C).

Initially, the group planned to create a separate list of words for determining the language used in a given text article and for determining whether the text article is disaster related or not. It was also decided that all Filipino words found in a dictionary will be used as the language guesser. However, upon further researching about creating the list of trigger words, the group found out that combining the list of words for language guessing and for disaster association proves to be more practical than separating them.

In order to do this, a sufficient amount of documents that will be processed must first be gathered. The documents in connection to the study's domain are text articles that are disaster related. One of the main issues encountered in performing this task was collecting a huge amount of disaster related news articles in the English and in the Filipino language. The process was hard to automate due to the accuracy required in order to generate a reliable list of words. Initially, the group tried to automatically crawl news articles and just filter out non-disaster related articles. However, this process proved to be more time consuming thus leading the group to manually collect the news articles. 500 disaster related articles were manually collected for each language, totaling to 1000 articles.

After collecting and preprocessing the articles, the frequency of each of the words in the articles was counted. The Term Frequency – Inverse Document Frequency (TFIDF) algorithm was utilized to accomplish this task. TFIDF is an algorithm coded in C that determines the relevance of a word in a set of documents or document queries. It weighs the value of a word given a specific domain using the following formula:

$$w_d = f_{w,d} * \log (|D|/f_{w,D})$$

Where $f_{w,d}$ is the frequency of a word in a single document

$f_{w,D}$ is the number of documents that contains the word

D is the number of documents

This algorithm weighs the relevance of a word not only based on the number of times it appears in a document. It also measures the relevance of a given word's existence in the whole set of documents related to a particular domain.

After initial success in processing small amounts of articles, it became apparent that articles in huge amounts could not be handled due to memory issues. Because of this, the tool running the TFIDF was recoded into the JAVA platform in order to resolve issues on memory capacity. After recoding, a list of trigger words with their corresponding TFIDF scores was successfully generated.

This algorithm is chosen over algorithms like Term Frequency Scoring because of its effectiveness. In a simple frequency scoring, the words are scored based on the times it appears in a whole set of documents. With this approach, there is no way of ensuring whether the top words are really relevant to the context of the domain or not. For example, articles such as "the", "a", "has" are very common words used in English text articles. Solely basing on the frequency count of these words will put them as the highest scored words in a given document. With the formula $|D|/f_{w,D}$ that is part of the whole TFIDF algorithm's formula, this problem is resolved. When D , the number of documents is divided by $f_{w,D}$, the number of documents that contains the word, articles that are present in all documents then computed to be zero. So when multiplied with its frequency, its score will be less and less valuable. This ensures that the top words generated are relevant to the context.

The list of trigger words gathered using this method is used in the crawler module. It is used by the crawler in identifying whether a certain text article is in English or Filipino and whether it is disaster related or not. For example, the word "sunog," which means fire in English, implies that the text with the word in it is a Filipino text. Also, it implies that the text is about a disaster since the word is commonly used to alarm people that there is a fire going on. On the other hand, the word "flood", implies that there is a flood somewhere in the world and that the text is written in English.

5.1.2 Database of POI's, Street Names, and Coordinates

This database contains all points of interest including hospitals, schools, and public landmarks in Metro Manila. In addition, it also contains all the metropolis' street names including high ways, alleys, and main roads. Each of these has corresponding longitude and latitude coordinates that point to their precise position in a map.

Initially, the group planned to create a list of all places in Metro Manila without their corresponding coordinates. After matching the name of a place to the location indicated in a given report, an API with a searching tool will be used to locate its location in a virtual map. The data for this were manually collected from various sources. The issue that was encountered while doing this was the availability of the resources and their corresponding formatting. For example, in popular landmarks, the resources came from different sources because information from only one source proved to be lacking. A source with a complete compilation of all landmarks in Metro Manila is still nonexistent. Using the gathered information, the group had to deal with different formatting and numerous information duplications. Also, for this implementation, it is possible to have numerous points in the map show at the same time. For example, Jollibee was given as the point of interest in a given report. Since there are numerous Jollibee fast food restaurants in Metro Manila, the system will then have to search for the correct one, a task proving to be nearly impossible to accomplish by merely using a search function of

an API. As a result, the system decided to change the implementation of the database to contain the coordinates of each location entry.

Table 5.1: Data Representation of Location Coordinates

Location	City	Amenity	Postal	Longitude	Latitude
Brother's Burger	Taguig	restaurant		121.051537	14.5501827
Brother's Burger	Makati	restaurant	1200	121.016564	14.5595541
Brother's Burger	Muntinlupa	restaurant		121.033981	14.421676
Brother's Burger	Pasig	Fast food	1605	121.063562	14.585257

Table 5.1 shows a sample data representation of the location, its extended information, and its coordinates. The location is the specific location itself may it be a point of interest or a street name. The extended information, in the sample above “Taguig, Makati, Muntinlupa, Pasig,” is stored in order to differentiate similar location names. For example, Brother’s Burger, a burger joint, has many branches all around the Philippines. The extended information points out specifically which Brother’s Burger is referred to. The last two columns are longitude and latitude respectively that points out the specific coordinates of the location on the map.

In changing to this implementation, the group decided to use OpenStreetMap (OSM) as it is the most available for developers. It also contains the most complete information about locations in Metro Manila. OSM allows users to download and extract a file from their website that contains all information about a given part of their virtual map. This contains street names, points of interests, and their corresponding coordinates. With this, the issue of multiple possible points in the map is solved. For example, when the system is looking for Jollibee, the data gathered from OSM contains extended information on Jollibee. It contains the street it resides on, the city, zip code, and other information that makes it unique from other branches. With this, the issue on duplicates is eliminated making each of the location unique on its own.

There are numerous tools available on the Internet for extracting data from OSM and transferring them into a database. The tool that is usually used in transferring OSM data into either MySQL or Portgres SQL database is Osmosis. The issue that the group encountered with Osmosis is its outdated technical manual. Some functions are altered and renamed which hindered the group to successfully transfer the data. Because of lack of documentation to refer to, the group decided to search for other tools with the same functions as Osmosis. The group stumbled on some tools but requires Linux to operate. Because of unavailability of a usable machine, the group decided to manually transfer and extract the data by coding a JAVA program that will do the data migration.

After successfully extracting all the data from OSM, it is now then cleaned to remove unusable entries that only eat up memory space. The database is used by the front end in order to know what part of the map the location should be plotted on. The front end will first be given information on the disaster and the location. After that, it will search through the database where it can acquire the longitude and latitude of the location that will enable it to determine the location on the map. The names of the locations are also used in the pattern extractor module as seed words. They are used to find patterns that will be able to extract names of the locations that were retrieved from the OSM.

5.1.3 SQL Table

The information extraction and validation engine makes use of a MySQL database to store and retrieve necessary information. These backend engines mainly use the following tables in the database: (1) report, (2) category, (3) area, and (4) user.

The report table houses all reports that the system has gathered. The system stores the following:

- report ID of a particular report, a unique set of integers that identifies a report
- SOMIDIA ID of the poster, if available
- title of the report, if available
- content of the report
- Category ID of the report which references the Category table
- Date and Time of the Event
- Date and Time when the report was retrieved
- Location of the event
- Longitude and Latitude coordinates
- Cluster Score
- Extraction Score
- Source Type (e.g. SOMIDIA, Twitter, Facebook)
- Source ID (e.g. SOMIDIA ID, Twitter ID, Facebook ID)

The system assigns cluster and extraction scores depending on the number of similar reports and the success of the extraction process of a particular report, respectively.

Table 5.2: Representation of Report Table in the mySQL Database

Report ID	Poster	Title	Content	Category
1	jeninaluchua	n/a	There is a fire in Caloocan	3
2	snapps_03	n/a	Grand Central is burning!	3
3	cojustin28	n/a	Kasalukuyang nasusunog ang Grand Central sa may Caloocan	3
4	herman_jr866	Grand Central is currently burning	Grand Central Gotesco Mall is currently burning	3

Time of Event	Date of Event	Time Posted	Location	Longitude
n/a	n/a	2012-03-17 00:12:11	Caloocan	120.983
n/a	n/a	2012-03-17 00:55:28	Grand Central	120.984444
n/a	n/a	2012-03-17 01:08:44	Grand Central	120.984444

2012-03-17	00:00:00	2012-03-17 01:32:23	Grand Central	120.984444
------------	----------	---------------------	---------------	------------

Latitude	Cluster Score	Extraction Score	Source Type	Source ID
14.653	10	6.5	Twitter	JLCee
14.005	10	7.5	Twitter	Snapps03
14.005	5	6	Facebook	justin.co
14.005	5	4.5	SOMIDIA	herman_jr866

The category table is referenced by the reports table in order to identify which category a particular report belongs to. In the reports table, only category IDs are given. In order to get the actual name of a particular category, it must match the id to this table. This has been done in order to facilitate flexibility in such a way that if the system gets an upgrade with regards to the kinds of disasters that it can extract. By doing this, it would be easy for developers to add disaster categories into the system.

Table 5.3: Representation of Category Table in the mySQL Database

Category ID	Name	Description
1	Earthquake	Description of Earthquake
2	Rain/Flood	Description of Rain/Flood
3	Fire	Description of Fire

The area table is also referenced by the report table. Based on the location tagged on a particular report, the system searches for its specific coordinates from the area table. It then maps the identified coordinates onto the report table in order to allow it to be marked on a virtual map.

Table 5.4: Representation of Area Table in the mySQL Database

Area ID	Name	Parent ID	Longitude	Latitude
0	None	0	0	0
1	Caloocan	0	120.983	14.653
2	Grand Central	1	120.984444	14.005

Due to the fact that the system makes use of a reputation model in order to determine the credibility of a particular user, the system accesses the user table whenever it encounters reports. Based on the Source Type and Source ID extracted on a particular report, it searches the user table for that particular user. If the user is nonexistent, it adds a new record. On the other hand, if the user exists, the system manipulates the credibility score which acts as a multiplier based on the validation results of the report.

Table 5.5: Representation of User Table in the mySQL Database

User ID	Username	Password	Type	First Name
1	jeninaluchua	Password1	1	Jenina
2	snapps_03	Password2	1	Angelo
3	cojustin28	Password3	2	Justin
4	herman_jr866	Password4	3	Herman

Last Name	Sex	Birthdate	Location	Email
Chua	F	1992-04-15	Caloocan City	jenina.chua@yahoo.com
Magpantay	M	1992-03-26	Makati City	angelo.magpantay@gmail.com
Co	M	1992-03-08	Nagtahan	herman_jr866@yahoo.com
Cheng Jr.	M	1992-09-24	Quezon City	cojustin28@yahoo.com

Mobile Number	Twitter	Facebook	Credibility Score	Creation Date
09264888381	JLCee	jenina.chua	10	2012-01-21 11:00:23
09273343433	Snapps03	angelo.magpantay	10	2012-01-21 14:21:43
09212122465	HermanJr866	herman.cheng	2.1	2012-03-04 08:32:14
09232321232	n/a	justin.co	5.4	1012-03-07 10:51:49

5.1.4 Extractor Seed Words

The extractor seed words are a list of words that the system will need to extract. The list is composed of words that pertain to specific disasters that the system handles such as flood, typhoon, fire, and earthquakes (See Appendix D), as well as the names of locations that were retrieved from the OSM. The list of disaster words were created by getting all the synonyms and permutations of the disasters that the system handles, as well as some of the other forms of the words that the group noticed in the tweets that they collected. This list is different from the list of trigger words used in the crawler as the trigger words were meant to find common words in disaster-related articles and not necessarily disaster-related words. The list of locations names were simply taken from the database of locations taken from OSM.

5.2 Crawler Module

The crawler module consists of different sub-modules that retrieve documents that pertain to disaster events from a variety of sources such as web sites, Twitter, Facebook, and SMS. This module also contains a module for classifying the retrieved documents to determine if they are related to disaster-management and to determine the language of the text.

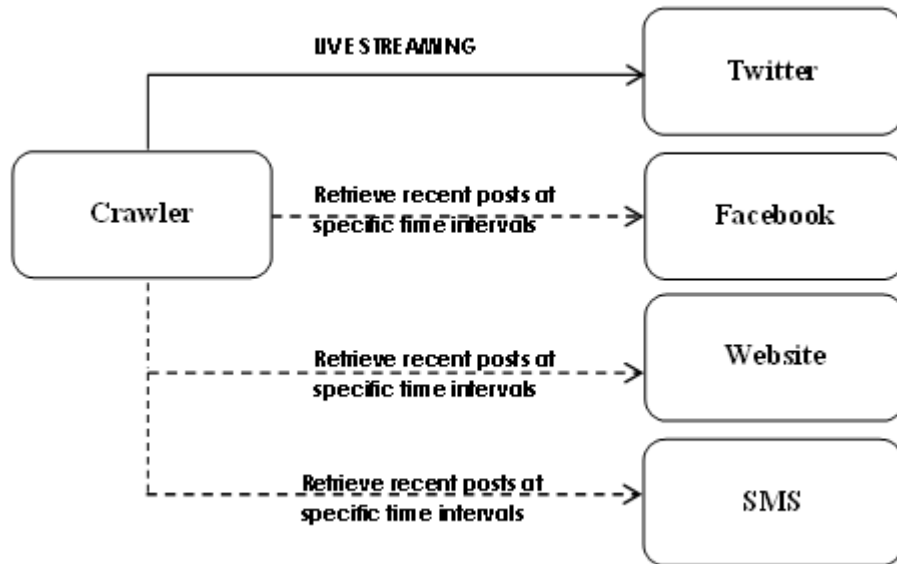


Figure 5.1: Crawler Module Illustration

5.2.1 Website Crawler

The web crawler sub-module handles the retrieval of documents from different Philippines news websites. The system made use of Crawler4j (Yasser, 2007), an open source web crawler for java, developed by Yasser Ganjisaffar. This module makes use of a text file that contains urls of websites that the system will crawl and will output a series of text files that contains news articles relating to disasters that are separated by language. See Appendix E for the complete list of websites being crawled by the module. The website crawler only retrieves recent articles at the end of the specified interval.

Crawler4J is an interface that allows for multiple crawlers to retrieve articles which enables a faster gathering of documents for each websites. This was an important feature for the system as it allows for the system to gather documents from multiple websites and helps the system to achieve its objective of near a real-time information extraction. The multi-threaded nature of crawler4j that allows for multiple crawlers requires two classes to be created: a controller class and a crawler class.

The controller class, `WebsiteCrawlerController`, must first initializes configuration of each crawler before creating the crawlers with the specific configurations as shown in Code Listing 5.1. Configurations of the crawlers include number of crawlers, politeness delay, maximum depth crawled, maximum pages crawled, able to resume crawling, domains that the crawler will crawl, and directory to store temporary files. This class also handles the proper shutdown of the crawlers that it created.

Code Listing 5.1

```

public void runCrawler() {
    //initialization of configuration
    config.setCrawlStorageFolder
  
```

```

        (crawlStorageFolder);
        config.setPolitenessDelay(500);
        config.setMaxDepthOfCrawling(maxDepth);
        config.setMaxPagesToFetch(maxPages);
        config.setResumableCrawling
            (resumeCrawling);
        crawlerDomains = FileManipulation.
            readFile(directory).split("http://");

        pageFetcher = new PageFetcher(config);
        robotstxtServer = new RobotstxtServer
            (robotstxtConfig, pageFetcher);
        controller = new CrawlController(config,
            pageFetcher, robotstxtServer);

        customData.add(crawlerDomains);
        customData.add(console);
        controller.setCustomData(customData);

        for (String s : crawlerDomains) {
            controller.addSeed("http://" + s);
        }
        //starting of crawlers
        controller.start(WebsiteCrawler.class,
            numberOfCrawlers);
    }

```

The crawler class, `WebsiteCrawler`, is the class that controls the process retrieval of documents from each link. It specifies whether or not to retrieve the article based on the URL of the webpage using the method `shouldVisit()` and what is done during the retrieval process using the `visit()` method.

The `shouldVisit()` method simply determines whether the next webpage needs to be retrieved or not by checking the url of the webpage and comparing it to the domain name of the website that it is crawling. This prevents the crawler from moving to a different website that is unrelated to the news website that it is crawling. Another filter of this method is checking whether the link leads to another webpage or a different file type by checking the extension of the link.

The `visit()` method was first implemented by simply retrieving all the text of the website and then writing the output into a text file. This however presented the issue of the “clutter” presented by the different components of the website that is not part of the news article such as logos, navigational elements, templates, advertisements, etc. This “clutter” affected the classification of the article since it may include words that are completely unrelated to the article itself. Ideally, the only part of the website that the system would crawl is the news article itself so as to minimize the error in classifying the document.

This issue was resolved by making use of another tool, Boilerpipe (Kohlschutter et al., 2010) , which makes use of shallow text features analysis for classifying the individual text elements in a web page. This allowed the system to detect and remove the surplus “clutter” around the main textual content of a web page. The system made use of the ArticleExtractor of Boilerpipe which is an extractor that is tuned to detect the main

content of news articles as shown in Code Listing 5.2. This extractor needed an input of the actual html code of the webpage which was retrieved by the crawler.

Code Listing 5.2

```
urlStream = new StringReader(html);
inputSource = new InputSource(urlStream);
input = new BoilerpipeSAXInput(is);
doc = in.getTextDocument();
article = ArticleExtractor.INSTANCE.getText(doc);
```

Another issue encountered was during the attempt to filter the webpages being visited by the crawler to articles a specific area of the webpages like the Metro section of new web sites so as to limit the articles retrieved to be focused on the target area of Metro Manila. The problem however is that the URL of different news websites follows different formats, as shown in Table 5.6, that it is difficult to create a rule that will address all formats. It would be possible to tailor a filter for each website however it will be needed for every website that will be added into the list and so will be quite difficult to update the web crawler for those who are not proficient in programming.

Table 5.6: Sample links of news websites

News Website	List of news article	News Article
Inquirer.net	newsinfo.inquirer.net/category/inquirer-headlines/metro/page/2	newsinfo.inquirer.net/220097/ltfrb-defers-cut-on-jeepney-taxi-fares
Philippine Star	http://www.philstar.com/nation/metromanila.aspx?start=1&maxrows=10	http://www.philstar.com/nation/article.aspx?publicationSubCategoryId=65&articleId=822743
Manila Bulletin	http://www.mb.com.ph/content/main-news	http://www.mb.com.ph/articles/364090/p35567-pay-for-teachers...
ABS-CBN	http://www.abs-cbnnews.com/list/Nation	http://www.abs-cbnnews.com/nation/07/01/12/china-sends-patrol-ships-disputed...
GMA	http://www.gmanetwork.com/news/archives/news/metromanila	http://www.gmanetwork.com/news/story/263748/news/metromanila/dole-only-8-of-158-bus-firms...

5.2.2 Facebook Crawler

The Facebook crawler sub-module handles the retrieval of documents from Facebook sources. The system made use of RestFB (restfb.com, n.d.), a java client tool that leverages on Facebook Graph API to retrieve articles from Facebook. This module makes use of a text file that contains username of users that the system will get articles from, and the id of the last post that the system retrieved. The system currently crawls from 54 different Facebook pages that are primarily from official news media,

government agencies, and non-government organizations (See Appendix F). The system will output a series of text files that contains articles relating to disasters that are separated by language. The facebook crawler retrieves recent posts at the end of a specific time interval.

The use of Facebook Graph API for any interaction with Facebook requires an access token which cannot be generated without creating an external application in Facebook. This provides a client_id and client_secret that is needed to acquire an access token. The system retrieves an access token by creating an HTTP GET request that is sent to Facebook using the requestAccessToken() method as shown in Code Listing 5.4.

Code Listing 5.3

```
private String requestAccessToken() {
    data = URLEncoder.encode("client_id", "UTF-8") + "="
        + URLEncoder.encode(app_id, "UTF-8");

    data += "&" + URLEncoder.encode("client_secret",
        "UTF-8") + "=" + URLEncoder.encode(app_secret,
        "UTF-8");

    data += "&" + URLEncoder.encode("grant_type", "UTF-
        8") + "=" +
        URLEncoder.encode("client_credentials", "UTF-
        8");

    url = new URL("https://graph.facebook.com/oauth/
        access_token?" + data);
    conn = url.openConnection();
    conn.setDoOutput(true);
    wr = new OutputStreamWriter(conn.getOutputStream());
    wr.write(data);
    ...
}
```

After obtaining an access token, the system then retrieves the entire feed of each user that is in the input file. Each post is then determined for its language and relevance to disaster management and saved into a text file similar to the website crawler. The information stored in the text file includes the post_id, user_name, date_posted, and the post itself. An issue encountered in the implementation of the Facebook crawler was that it kept crawling the entire feed of the user. The id of the last post retrieved for each user was then saved in the original text file containing the list of users to be crawled.

Another issue encountered in the implementation is the limited number of post that can be crawled by the system. Majority of the users' posts cannot be accessed due to the privacy settings of the users. The only posts that the system can access are only those users whose profiles are set in public mode. These users however are not many in number and there is no way to automatically retrieve a list of users with these settings especially since the users need to be located within Metro Manila. One solution to this problem is to create an application within Facebook that will request permission from

users to access their posts. This solution however would still require the app be spread to plenty of users to increase the number of sources of articles.

5.2.3 Twitter Crawler

The Twitter crawler sub-module handles the retrieval of documents from Twitter sources. The system made use of Twitter4J, a java client tool that leverages on Twitter API to retrieve articles from Twitter. This module makes use of a Twitter account that follows relevant users that the system will get articles from and it will output a series of text files that contains tweets relating to disasters that are separated by language. Unlike the two previous two modules, the twitter crawler continuously listens for new tweets but would save all the tweets it retrieved at the end of the time interval.

The system made use of Twitter API's User Stream API to retrieve all the tweets of those users that the system follows. User Streams provide a stream of data and events specific to the authenticated user. User Streams not only retrieves all tweets of followed twitter users, but also all tweets that mention these users. The group created the @SOMIDIA_IEDM twitter account to use as the authenticated user that will follow relevant persons that the system will gather articles from. Code Listing 5.5 shows how to access the user stream using Twitter4J.

Code Listing 5.4

```
twitterStream = new TwitterStreamFactory()
    .getInstance();
twitterStream.addListener(getListener());
twitterStream.user();
```

Twitter4J retrieves tweets the moment the account's friends update their status. This is implemented in the Twitter Stream listener as shown in Code Listing 5.6. The `onStatus()` method of the listener process each tweet of the users that are being followed, retrieving information such as `tweet_id`, `user_name`, `time_posted`, and the tweet itself. Each post is then determined for its language and relevance to disaster management and saved into a text file similar to the website crawler. The group also included additional information regarding the user who tweet the article in order to improve the analysis of the credibility of each tweet. The following information was included in the saved text file: `num_of_followers`, `num_of_following`, `status_count`, `account_age`, and `user_location`.

Code Listing 5.5

```
private static UserStreamListener listener = new
    UserStreamListener() {

    public void onStatus(Status status) {
        dateFormat = new SimpleDateFormat
            ("MM/dd/yyyy HH:mm:ss");
        date = new Date();
        dateReceived = status.getCreatedAt()
            .toString();
```

```

        user = status.getUser().getScreenName();
        tweet = status.getText();
        id = String.valueOf(status.getId());

        lg = new LanguageGuesser(tweet);

        if (lg.guessLanguage().equals("English")) {
            tweetListEng.add(new Tweets(id, user,
                dateReceived, tweet));
        } else if (lg.guessLanguage().
            equals("Filipino")) {

            tweetListFil.add(new Tweets(id, user,
                dateReceived, tweet));
        } else {
            System.out.println("Not DM");
        }
    }
}

```

An issue with regards to the number of tweets that the system is able to retrieve was encountered. In the first implementation of the crawler, the twitter account @SOMIDIA_IEDM is only following 108 users. These users were selected by the group because they are accounts that are primarily used as an avenue for reporting news or accounts of noted journalists. Example of these users include @gmanews, @govph, @inquirerdotnet, @MMDAtweets, @PAGASAFFWS, etc. The accumulated tweets from these users a day however amount only to around 2000 tweets, without the removal of non-disaster related tweets. The group addressed this issue by following users who mention the original users being followed. The users however must have posted at least 1000 tweets to ensure that they actively use twitter and would imply that they would be a possible source for disaster related events. The system now have two types of twitter accounts that it follows, the original 108 user accounts (see Appendix G) composed of media reporters, LGUs, and NGOs, and the second type is 1580 twitter accounts that it would treat as a community. The group would weigh the tweets of users from the first group more important than the tweets from the second group.

5.2.4 SMS Crawler

The SMS crawler sub-module handles the retrieval of sms documents directly from the cellphones of users. The system made use of Globe's SMS/MMS API, which provides a short-code number and a 4 digit suffix code for users to text in order for them report directly to the system through sms messaging. The sub-module follows a simple template of <disaster-type>[space]<location> (e.g. FLOOD De La Salle University) in order for users to easily remember how to make use of the feature. The sub-module does not need to pass through the language guesser and the extractor modules due to the template and would only need to undergo validation before being shown in the website. The sub-module stores all sms message into the database the moment it receives them from the users.

An issue with the SMS Crawler is that the SMS/MMS API is limited to globe subscribers only. This limitation prevents a huge number of people from accessing the system. Another issue is that the number of the user must be registered first into the API for them to be able to interact with the API. This would mean that only those people who

are registered to the system's website and provided a mobile number would be able to report through sms. Since the API did not provide a way to register a number automatically, it means that an administrator would need to manually encode the mobile numbers of users into the Globe API which is very tedious.

5.2.5 Language Guesser

The Language Guesser sub-module is the module that classifies the article to determine if the text is related to disaster management and determines the language used by the text. It takes as input the text to be classified and returns the language of the text. If it returned the text "None", it means that the article is not related to disaster management.

This classifying process is done by a simple keyword search using the trigger words generated as the keywords to be searched. The keyword search is executed using LingPipe's Approximate Dictionary-Based NER (see discussion on NER) to determine if a specific number of keywords from the list of trigger words were found within the text, it will label the text as disaster-related. It would then count the number of words found that it associated with the Filipino language. If the count reaches a certain threshold, it would classify the text as Filipino and it would classify the text as English if it doesn't reach the threshold.

Code Listing 5.6

```
public String guessLanguage(String input) {
    String result = "None";
    int filCount = 0;

    if (!entitiesFound.isEmpty()) {
        if (entitiesFound.size() < wordThreshold)
            return result;

        for (String[] arrTemp : entitiesFound) {
            if (arrTemp[1].equalsIgnoreCase
                ("Filipino")) {
                filCount++;
            }
        }
        if (entitiesFound.size() > 0) {
            if (filCount >= filipinoThreshold) {
                result = "Filipino";
            } else {
                result = "English";
            }
        }
    }
    return result;
}
```

5.3 Preprocessing

The preprocessing module is composed of different sub modules, each representing a preprocessing step as a preliminary for the information extraction process. Multiple tools performing the same task were used in implementing each sub module. This was done in order to make the system flexible when it comes to supporting multiple tools and enabling easy integration of new tools. In addition, it also allows user selection of tools. An abstract class was used for each sub module in order to enforce uniformity among the tools used.

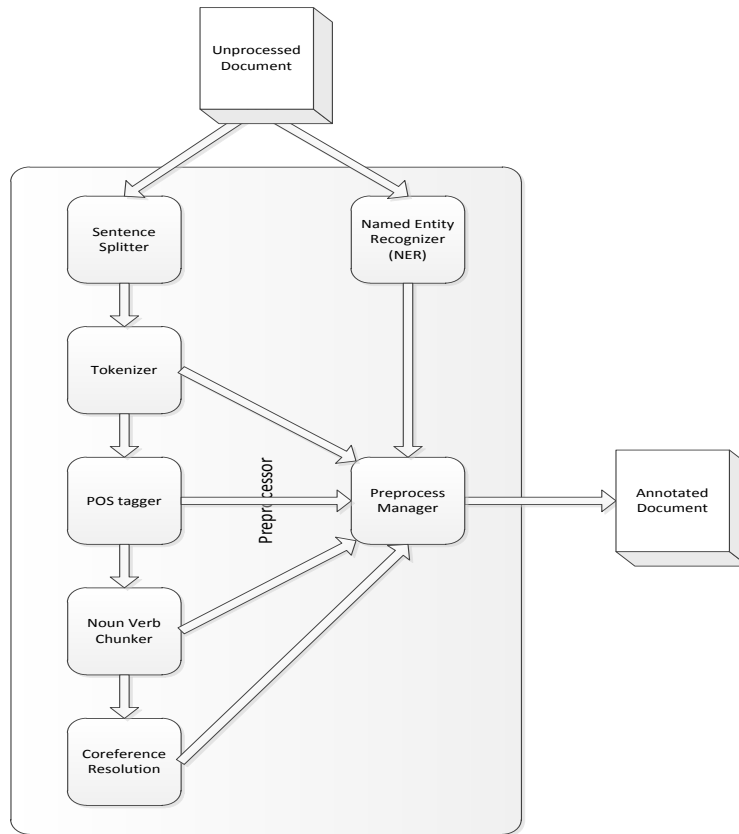


Figure 5.2: English Preprocessing Module Sequence Diagram

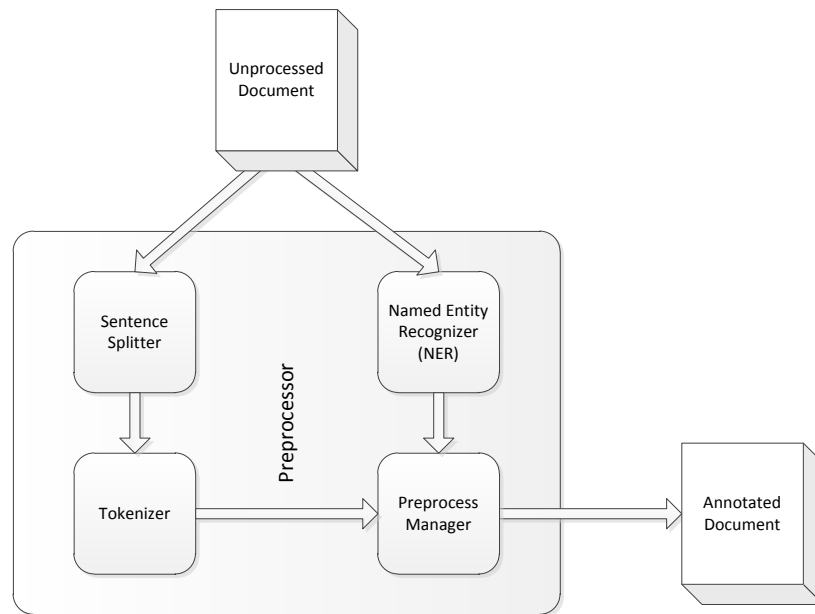


Figure 5.3: Filipino Preprocessing Module Sequence Diagram

5.3.1 Sentence Splitter

The sentence splitter sub module is the first of the many modules in the preprocessing step. It is in charge of parsing the input document and separating its content into individual sentences in order to prepare it for succeeding modules. It takes in a string as an input and returns an array of string values where each sentence is placed on a string array index.

Any of the two tools, namely the LingPipe and the OpenNLP sentence splitters, can be used to perform the processes involved in this sub module.

The LingPipe sentence splitter starts off by initializing an instance of the tokenizer `IndoEuropeanTokenizerFactory`. `INSTANCE` as well as a new sentence splitter module by calling the function `MedlineSenteceModel()`. The tool splits the text of the input document by making use of the `splitSentence()` method. In this method, the sentence is tokenized into tokens and whitespace tokens through the following code:

Code Listing 5.7

```

TOKENIZER_FACTORY.tokenizer(inputString.toCharArray()
    ,0,inputString.length()).tokenize(tokenList,
    whiteList)

```

The whitespace tokens are string values found in-between the found tokens. After tokenization, the sentence splitter model detects the bounds of the sentences and then proceeds to rebuild the individual sentences using the detected bounds and returns them. This is done using the following code:

Code Listing 5.8

```
SENTENCE_MODEL.boundaryIndices(tokens, whites)
```

As for the OpenNLP sentence splitter, it starts off by initializing a sentence splitter model as follows:

Code Listing 5.9

```
modelIn = new FileInputStream("model/en-sent.bin");  
sentenceModel model = new SentenceModel(modelIn);  
sentenceDetector = new SentenceDetectorME(model);
```

For each input document received through the `splitSentence()` method, the input document is sent into the generated module which, in turn, returns the split sentences with the following code:

Code Listing 5.10

```
sentenceDetector.sentDetect(inputString);
```

In terms of comparing the two tools that were used for the sentence splitter sub module, the LingPipe sentence splitter uses a model developed from a corpus of medical document. It also requires that the input document be tokenized first before breaking down the document into individual sentences. On the other hand, the OpenNLP sentence splitter makes use of a model developed from open NLP's training data and does not require tokenization of the input document before functions. The OpenNLP sentence splitter is preferred over the LingPipe sentence splitter due to the fact that it does not need the tokenization of the input document before being able to function, thus reducing processing time.

5.3.2 Tokenizer

The tokenizer sub module immediately follows the sentence splitter sub module. It is in charge of parsing a given sentences into individual tokens which will be used by succeeding preprocessing sub modules. It takes in a sentence input produced by the sentence splitter sub module and returns an array of string values where each token is placed on a string array index.

Any of the two tools, namely the LingPipe and the OpenNLP tokenizers, can be used in performing the processes involved in this sub module.

The LingPipe tokenizer starts off by declaring a `tokenizerFactory` which handles the generation of the tokenizer. Input sentences are sent to this module by calling the `tokenize()` method. By calling this method, the input sentences are one by one sent

to the generated tokenizer which then returns the tokenized sentences. The following code snippet shows how this process is done:

Code Listing 5.11

```
TOKENIZER_FACTORY.tokenizer(strTemp, 0, strTemp.length)
.tokenize();
```

On the other hand, the OpenNLP tokenizer starts off by initializing the tokenizer model as follows:

Code Listing 5.12

```
modelIn = new FileInputStream(inputModel);
tokenizerModel model = new TokenizerModel(modelIn);
tokenizer = new TokenizerME(model);
```

For every sentence passed to this module using the `tokenize()` method for processing, it returns the tokenized data of the sentence input. This following code snippet shows how this process is done:

Code Listing 5.13

```
tokenizer.tokenize(inputString);
```

One of the issues encountered in implementing the two tools in order to allow user selection was the existence of “extra” features in one of the tool (OpenNLP’s tokenizer) that was absent in the other (LingPipe’s tokenizer). The group’s initial solution for this problem was to ignore the “extra” features of one of the tools in order to enforce uniformity. However, this solution introduced a new problem encountered on the latter part of the preprocessing step. During the compilation of all the completely preprocessed data, the lack of the “extra” data in the output of one of the tools made it difficult to process them in the succeeding modules as it required the adding of extra steps in order to make use of the data properly. This problem was introduced by adding the “extra” features to the tool that was lacking them. The methods that were implemented in order to allow the other tool to perform the extra features are the following:

- `getWhitespaceTokens()`
- `getTokenizeSpansfunction()`
- `stringIndexToTokenIndex()`
- `stringIndexToTokenIndex()`

In terms of comparing the two, LingPipe’s tokenizer performs faster as compared to OpenNLP’s. This is mainly due to the simplicity of the LingPipe tokenizer. However, this simplicity introduces limitations to the tool. The LingPipe tokenizer is unable to detect

acronyms separated by periods (".") such as "U.S.A.". Using this example, the letters U, S, and A, will be treated as individual tokens along with the periods that separated the letters: {"U", ".", "S", ".", "A", "."}

The OpenNLP tokenizer is more complex as it made use of a corpus to develop a model that enables it to perform better tokenization. Because of these reasons, the OpenNLP tokenizer is preferred over the LingPipe tokenizer.

5.3.3 POS Tagger

The Part-of-Speech (POS) tagger sub module immediately follows the tokenizer sub module. It is in charge of associating each input token with a part of speech given a particular sentence. It takes an array of tokens produced by the tokenizer module as input and returns an array of string values representing the part of speech tag of each of the input tokens.

Any of the two tools, namely the LingPipe and the OpenNLP POS taggers, can be used in performing the processes involved in this sub module.

The LingPipe POS tagger starts off by initializing a Hidden Markov Model (HMM) which will be used to identify the pos tag of a particular token. In order to start the tagging process, tokenized sentences are passed one by one to the HMM through the tag() method. Each tokenized sentence received by the HMM identifies the corresponding POS tag and returns N results through the following method:

Code Listing 5.14

```
tagger.tagNBest(tokenList, MAX_N_BEST);
```

The method then identifies which of the N results scored the highest and transfers the tags to an array of string values corresponding to the tokenized sentences. The array is then returned in order to be used by the succeeding sub module. A complete code snippet as to how LingPipe's POS tagger is shown as follows:

Code Listing 5.15

```
fileIn = fileIn = new FileInputStream(inputModel);  
objIn = new ObjectInputStream(fileIn);  
HiddenMarkovModel hmm = (HiddenMarkovModel)  
objIn.readObject();  
tagger = new HmmDecoder(hmm);
```

On the other hand, the OpenNLP POS tagger starts off similarly as that of LingPipe's. It initializes the model that comes as the tool's default. This is done by calling the following methods:

Code Listing 5.16

```
fileIn = fileIn = new FileInputStream(inputModel);  
objIn = new ObjectInputStream(fileIn);  
HiddenMarkovModel hmm = (HiddenMarkovModel)
```

```
objIn.readObject();
tagger = new HmmDecoder(hmm);
```

For every tokenized sentence fed into this tool, it is sent to the model which returns an array of string values containing the POS tags of the input. The array output is then returned in order to be used by the succeeding sub module,

The performance of the tools was measured by creating a java application that will calculate the time it takes for the tools to tag a given text with POS tags. The group used three input texts of different sizes: small, medium, and large.

Table 5.7: LingPipe vs. OpenNLP Performance Results

Input Size	Input Text	Time (in milliseconds)	
		LingPipe	OpenNLP
Small	Hello World!	678	1866
Medium	Most large cities in the US had morning and afternoon newspapers. Large cities in the US had morning and afternoon newspapers.	700	1922
Large	Bangkok (The Nation/ANN) - Despite the weaker external environment, growth in Southeast Asia... (See Appendix H for full text)	844	2048

In terms of comparing the two tools used, one major difference between the LingPipe and the OpenNLP POS taggers is their performance. The former performs POS tagging in a shorter amount of time given a text of a short length as compared to the latter. Although the difference between the two is only within a matter of seconds, longer texts would lead to a longer tagging process thus greatly affecting the overall performance of the system. Another difference between the two is that the LingPipe POS tagger can perform N-best tagging ensuring that the output is the best out of N trials. The OpenNLP POS tagger lacks this feature. Due to the superior performance of the LingPipe POS tagger of the OpenNLP POS tagger, the former is preferred

5.3.4 Named Entity Recognizer

The Named Entity Recognizer (NER) sub module immediately follows the POS tagger sub module. It is in charge of associating tokens with their corresponding entity type. It takes in a string as input and returns the data regarding found entities. Since the input of this module is a simple text, the preprocessing step that it requires is sentence splitting.

For this sub module, four tools were implemented and/or created: (1) LingPipe's Statistical method, (2) LingPipe's Approximate Dictionary method, (3) an Exact Dictionary method, and (4) OpenNLP's method.

LingPipe's statistical method is used by first declaring a chunker model used to determine entities in a document. This process is shown in the following code snippet:

Code Listing 5.17

```
File modelFile = new File(inputModel);
chunker = (NBestChunker)AbstractExternalizable
    .readObject(modelFile);
```

This method receives sentences through the `findEntities()` method. By calling this method, each sentence is passed to the chunker which searches for a set of entities `N` times using the `chunker.nBest(cs, 0, cs.length, MAX_N_BEST)` function. The set that gets the highest score is then selected and has the entity, the type, and the starting and ending indices of the entity on the string extracted. In some cases, entities on the string overlap with each other. In order to minimize the amount of data, this method scans the sentence for entities from left to right. For every encountered entity, it is checked whether it overlaps with any entity that has already been identified. If so, it is added into the set of identified entities. If not, it is discarded. After scanning the sentence, the set of identified entities is returned for the use of the succeeding sub module.

As for the LingPipe Approximate Dictionary method, it starts off by initializing an empty dictionary through the following code snippet:

Code Listing 5.18

```
TrieDictionary<String> dictionary = new TrieDictionary
    <String>();
```

The dictionary is then built one entry at a time. In order to simplify this step, the entries of the dictionary are read and entered one entry at a time from a text file. The file consists of lines of text where each line is either an entry or a category. The syntax of an entry is a string and the syntax of the category is "<Category>:" + string. The entity type of each entry is determined by the nearest category above the entry. The tool reads the file one line at a time and determines if the line is either a category or an entry. If it is a category, it extracts the "string" and saves it into a variable. On the other hand, if it is an entry, it enters it into the dictionary as the entity alone with the category currently stored as the entity type. The process of doing this is shown in the following code snippet:

Code Listing 5.19

```
currentEntry = new DictionaryEntry<String>
    (currentFileLine.trim(), currentCategory);

dictionary.addEntry(currentEntry);
```

From the generated dictionary, an approximate dictionary is created to identify entities in a text. `maxDistance` is an integer variable that holds the maximum allowable edit distance of a query text to an entry. Edit distance is a measure of how close or how far a

string is to another string. It is the minimum number of edit, delete, and/or add to transform one string to another.

Code Listing 5.20

```
TokenizerFactory tokenizerFactory =  
IndoEuropeanTokenizerFactory.INSTANCE;  
  
WeightedEditDistance editDistance = new  
FixedWeightEditDistance(0, -1, -1, -1, Double.NaN);  
  
chunker = new ApproxDictionaryChunker(dictionary,  
tokenizerFactory, editDistance, maxDistance);
```

Each sentence is passed to this tool through the `findEntities()` method. By calling this method, the sentence input is sent to the approximate dictionary. Found entities are then returned as output. In some cases, overlapping entities in a sentence are found. In order to minimize this phenomenon, the tool scans for the entities from left to right. Whenever it identifies two overlapping entities, the entity having the higher score (lower edit distance) is selected and the other one is discarded. The final set of entities is then returned.

The Exact Dictionary method, on the other hand, makes use of hash functions and three tables in order to speed up string comparisons used in identifying an entity. The tables used in this tool are: (1) the token length table, (2) the single token table, and (3) the multi token table.

The token length table stores the first token of each entry along with its corresponding token length, the number of tokens on a particular entry. The single token table stores entities with token length equal to one (`== 1`) along with its entity type. Lastly, the multi token table stores entities with token length greater than one (`> 1`). The first token of the entry is extracted and added to the token length table along with its token length and adds the entry along with its corresponding entity type to the multi token table.

During the searching process, this method determines whether a given token is an entity token with a token length of one or more by referencing the token length table. If the entity's token length is one, its entity type is extracted from the single token table. If its token length is greater than one, the tool extracts the entire length of the possible entity and looks it up on the multi token table. If a match is found, the tool retrieves its corresponding entity type. If the current token is not entered on the token length table, the current token is either not part of the dictionary or not the first token on the dictionary entry. On either case, the token in question is skipped.

For the last method, the OpenNLP method starts off with an initialization of its model:

Code Listing 5.21

```
modelIn = new FileInputStream(inputModel);  
TokenNameFinderModel model = new  
TokenNameFinderModel(modelIn);  
nameFinder = new NameFinderME(model);
```

Each model used by this method is specific to an entity type. This means that it needs to use multiple models in order to detect different types of entities. For every sentence it receives for processing, it tokenizes the sentence and passes it to the `nameFinder` model using the function `nameFinder.find(sentence)` which returns a set of found entities. These entities may overlap with each other. In this case, the tool scans the sentence from left to right and discards the flaking overlapping entities. The remaining entities are then returned.

An issue encountered for this particular method stems from the lack of data extracted for the initial design. It originally returned the string of the entity along with its entity type. This turned out to be troublesome during latter modules as it required additional processing to properly process the data. The solution that the group opted for was to add other data on the output such as the start and end spans of the entity found on the string.

In terms of comparing the tools and methods used, LingPipe's approximate dictionary method and exact dictionary method are both dictionary-based NERs. This means that for an entity to be recognized, it must be entered into the input dictionary of these methods use. The difference of these methods is that the approximate dictionary NER can identify entities within an edit distance given a dictionary entry. This means that if the dictionary contained an entry "computer" and the threshold accepted by this tool is set as one, the string "computers" would still be identified as an entity since the edit distance is one, which is within the threshold. Whereas the exact dictionary NER can only match strings with entries that are exactly the same.

Unlike the dictionary approach to NER, LingPipe's statistical method and OpenNLP's method are statistical-based NER systems. Both of these make use of a corpus to develop a model used by the NER. Both of these NER tools can identify a person, place, or organization given a text. The difference between the two is how they make use of the models. OpenNLP's method uses one model for each of the entity types that it searches for, whereas LingPipe's statistical method uses one model for all its entities. This means that for OpenNLP's NER to identify persons, place, and organization, it needs to use three models on three separate occasions (the use of threads to identify entities is not advised), whereas the latter would only need one.

LingPipe's approximate dictionary method is preferred over the exact dictionary method because it has the capacity of approximate matching of dictionary entries to strings. In addition to this, it can simulate the exact dictionary method when its threshold is set to 0. Between the two statistical methods, LingPipe's method is preferred over OpenNLP's because it can identify different entities concurrently. This enables it to perform faster when identifying multiple entities from a text.

Preference between the approximate dictionary method and the LingPipe's statistical method should be based on the approach of the users to solve a problem. The dictionary method enables more control to the users compared to the statistical method, but it requires a large set of data to run properly. Users have to enter each entity into the dictionary manually.

In cases where an entity is not recognized by the NER used, the simple adding of the entity into the dictionary of dictionary-based NERs is the only fix needed for the NER to be able to automatically adopt to the change. On the other hand, the statistical method requires the retention of the model in order to identify the new entity.\

5.3.5 Noun Verb Chunker

The Noun Verb Chunker sub module is in charge of identifying noun phrases and verb phrases given a POS tagged sentence. It takes in POS tags from the POS tagger module of a sentence as input and outputs an array list of integer arrays where each integer array contains the start and end token index of a phrase.

This tool adopts the noun and verb chunker as discussed in LingPipe's Algorithm. It searches for a noun phrase by identifying whether a POS tag is part of a set of initial noun phrase tags or not. The span of the noun phrase is extended until last instance of a token where its POS tag is part of a set of continuing noun phrase tags. POS tags that are part of the set of initial noun phrase tags are determiners, adjectives, nouns, and pronouns. POS tags that are part of the set of continuing noun phrase tags are determiners, adjectives, nouns, pronouns, adverbs, and punctuations.

This tool searches for verb phrases similar to the way it searches for noun phrases: making use a set of initial and continuing verb phrase tags. POS tags that are part of the set of initial verb phrase tags are verbs, auxiliary verbs, and adverbs. POS tags that are part of the set of continuing verb phrase tags are verbs, auxiliary verbs, adverbs, and punctuations.

5.3.6 Coreference Resolution

The Coreference Resolution sub module is in charge of searching for pronoun(s) in a document and identifying the noun phrase that it is pointing to. This module takes in tokens from the tokenizer sub module, POS tags from the POS tagger sub module, noun phrase spans and verb phrase spans from the noun verb chunker sub module and returns a set of coreferenced data.

This tool utilizes the algorithm of RuslanMitkov to identify the coreferenced data. Given a pronoun from a document, a specific number of sentences in front of the sentence that contains the pronoun as well as the sentence housing the pronoun itself are considered. From the sentences considered, the noun phrases that are found in those sentences are tagged as candidates to be the coreference or the data being referred to by the pronoun. On the sentence where the pronoun in question is found, only the noun phrases that comes before the pronoun is considered as candidates.

The RuslanMitkov utilizes a scoring approach in determining the likely coreference. Each candidate starts off with a score of 0. The candidate that scores that highest is the one that is most likely to be the coreference and is selected as the coreference data. If multiple candidates scores tie occurs among the highest scorers, the coreference is then determined based on the distance of the candidates to the pronoun.

The Ruslan Mitkov Algorithm consists of multiple rules where each candidate is scored. In order to speed up and to simplify the task of identifying the coreference, the text inputs of this sub module is first compiled into an array of CoRef Objects, where each object represents a token along with its other features namely its POS and its phrase type data, before the Ruslan Mitkov rules are applied to them. For every pronoun encountered by this sub module, the compiled data, the candidates for that pronoun along with the pronoun itself, are passed through each rule successively.

The Ruslan Mitkov Rules are as follows:

Rule of Definiteness: The rule states that definite noun phrases are given a score of 0 and indefinite ones are penalized with a score of -1. Definite noun phrases are noun phrases that are modified with a definite article and possessive or demonstrative pronouns. Indefinite noun phrases are noun phrases that are modified with indefinite articles. Definite article is the article “the”. Possessive pronouns are the articles “my”, “mine”, “our”, “ours”, “your”, “yours”, “his”, “her”, “hers”, “its”, “one”, “their”, and “theirs”. Demonstrative pronouns are the articles “this”, “these”, “that”, and “those”. Indefinite articles are “a” and “an”. Under this rule, the candidate of the current pronoun are scanned for one of the articles discussed. If it contains a march, it merits a score of 0 if it is definite and -1 if it is indefinite.

Table 5.8: Rule of Definiteness

Condition	Action
If candidate noun phrase is definite	No action
If candidate noun phrase is not definite	candidateScore -= 1

Rule of Givenness: Under this rule, the first noun phrase of a non-imperative sentence is credited with a score of +1. Otherwise, it does not merit any score. Imperative sentences are sentences that are giving commands or advice. Under this rule, the tool scans each of the sentences considered. If the first word/token is neither a verb nor an adverb, the score of the first candidate on the said sentence is incremented.

Table 5.9: Rule of Givenness

Condition	Action
If candidate noun phrase is the first in a sentence and sentence is non-imperative	candidateScore += 1
Else	No action

Rule of Indicating Verbs: Under this rule, if a verb in the considered sentence is part of a predefined verb set, the noun phrase following it is rewarded with a score of +1. Otherwise, it does not merit any score. The verb set is as follows: {“discuss”, “present”, “illustrate”, “identify”, “summarise”, “examine”, “describe”, “define”, “show”, “check”, “develop”, “review”, “report”, “outline”, “consider”, “investigate”, “explore”, “assess”, “analyse”, “synthesise”, “study”, “survey”, “deal”, “cover”}. Under this rule, the sentences considered for the current pronoun are scanned. If a verb is found to be part of the verb set listed, the immediate candidate following the verb merits a score of +1.

Table 5.10: Rule of Indicating Verbs

Condition	Action
If the verb preceding a candidate noun phrase is part of the set special verb	candidateScore += 1
Else	No action

Rule of Lexical Reiteration: This rule states that noun phrases repeated within the same paragraph two times or more are credited with +2. On the other hand, noun phrases that are repeated once are credited with +1. Noun phrases that are not repeated will not merit any score. Under this rule, the candidates considered for the

current pronoun are counted if there are repetitions within themselves. If it is repeated more than twice, it merits a score of +2. If it is repeated only once, it merits a score of +1.

Table 5.11: Rule of Lexical Reiteration

Condition	Action
If candidate noun phrase is repeated 2 times or more	candidateScore += 2
If candidate noun phrase is repeated once	candidateScore += 1
If candidate noun phrase is not repeated	No action

Rule of “Non Preposition” Noun Phrases: Noun phrases that are part of a prepositional phrase are penalized with a score of -1. Otherwise, it does not merit any score. Prepositional phrases are phrases where a preposition is followed by a noun phrase. Under this rule, the sentences considered for the current pronoun are scanned for prepositions. If a preposition is found, the tool looks at the token following it. If it is a candidate, it merits a score of -1.

Table 5.12: Rule of “Non Preposition”

Condition	Action
If candidate noun phrase is not prepositional	No action
Else	candidateScore -= 1

Rule of Collocation pattern Preference: This rule states that if a noun phrase conforms to the pattern “<noun Phrase/pronoun><verb>” or “<verb><noun phrase/pronoun>” and the pattern is similar to that of the pronoun, it is credited with a score of +2. Otherwise, it does not merit any score. Given the phrase: “Press the key down and turn the volume up Press it again.” used as an example in Ruslan Mitkov’s Research paper, the noun phrase “the key” is credited with a score of +2 because it conforms to the second pattern. In addition the pronoun also conforms to the pattern. Under this rule, the tool first identifies whether the pronoun conforms to any of the two patterns or not. This is done by looking at the verb surrounding the pronoun. If the pronoun conforms to the rule, the identified verb is stored, as well as the type of pattern the pronoun conformed to. The candidates are then scanned if it conforms to the same pattern with same verb. If it does, it merits a score of +2. In the example given, the tool would identify that the pattern the pronoun conformed to is “verbAtFront” and the verb is “press”. The system would then scan the candidates if the token preceding it (since the pattern found is “verbAtFront”) is “press”. If it is, it conforms to the collocation pattern and would merits the score.

Table 5.13: Rule of Collocation Pattern Preference

Condition	Action
If candidate noun phrase has the same collocation pattern preference as the preposition	candidateScore += 2
Else	No action

Rule of Immediate Reference: This rule states that immediate reference to a noun phrase can hint that the said noun phrase is the coreference. This is credited with a score of +2. Immediate reference can be identified if it conforms to the pattern “(you) <verb><noun phrase> ... con (you) <verb> it ... con (you) <verb> it” where “(you)” is optional and con = conjunction = {“and”, “or”, “before”, “after”, ...}. Under this rule, the tool looks at the current pronoun. It then backtracks from the current pronoun making sure it still conforms to the pattern until it reaches a candidate. In that case, the candidate merits a score of +2. If during backtracking the pattern didn't match, backtracking is halted.

Table 5.14: Rule of Immediate Reference

Condition	Action
If candidate noun phrase is immediately referenced by a pronoun (if candidate noun phrase conforms to the patter)	candidateScore += 2
Else	No action

Rule of Referential Distance: Under this rule, if the pronoun is part of a complex sentence, the noun phrase in the previous clause is credited with a score of +2, noun phrases found on the previous sentence is scored +1, noun phrases 2 sentence back is scored +0, and so on. In simple sentences, noun phrases found on the previous sentence is scored +1, noun phrases 2 sentences back is scored +0, and so on. Complex sentences are sentences with two or more clause(s). This tool identifies if a sentence is complex by counting the occurrence of a clause by counting the presence of noun phrases and verb phrases. Under this rule, candidates on the same sentence as the pronoun are merited with a score of +2. Candidates on the previous sentence are merited with a score of +1. Candidates 2 sentence back merits a score of 0, and so on.

Table 5.15: Rule of Referential Distance

Condition	Action
If candidate noun phrase is part of a complex sentence and part of a clause preceding another clause which contains a preposition	candidateScore += 2
If candidate noun phrase is n sentence away from a sentence containing a preposition	candidateScore += 2 - n

5.3.7 Preprocess Manager

The preprocess manager is the class that manages the different tools utilized in each preprocessing operation. This also compiles all the data. The input of this module is a document and the output of this is an ArrayList of ArrayLists of HashMaps where each token along with its corresponding data is stored.

This class starts by initializing all the tools involved in the process. Once all the tools have been initialized, it then runs the input data through different tools. In some cases, the output of one tool is immediately fed to the next tool. Once the different data has been collected, the system then starts to compile them.

Compiling starts by first instantiating a container to hold compiled data. It instantiates an array list to hold the entire document being preprocessed. Then compiling is done one sentence at a time. For every sentence, the system generates a new array list to hold the current sentence being processed. The system then starts to compile the data of the current sentence. At this level, compiling is done one token at a time. For every token encountered by this class, it generates a new HashMap and enters the string and part-of-speech of the current token. It then checks whether the current token is a noun phrase or a verb phrase. If it is either one of those, it enters its phraseType as noun or verb. If not, it is entered as none. The system checks the phrase type of the current token by going through the found phrases on the current sentence from the noun verb chunker. If the current token index falls between a found phrase span, it is identified as a phrase. After it checks the token's phrase type, it checks if the current token is an entity identified by the NER. If it is, the POS tag of the said entity is changed to "ner" and it enters the entity type of the token into the current HashMap. In some cases where the entity found consists of multiple tokens, these tokens are merged into one. The system identifies whether a token is an entity or not by converting the output spans of the found entities from the NER module from string index to token index. The system then checks if the current token index falls between entity token index spans. If it does, it is identified as an entity. The system merges multi-tokened entities by replacing the string of each of the entities' first token into the string of the entire entity. It then skips the tokens corresponding to the entity's remaining token. Following the NER is the coreferencer. For every pronoun the system encounters, the system searches the corresponding coreference that that pronoun refers to. This is done by searching through the output of the coreference resolution module. Once the corresponding coreference is found, it replaces the pronoun with the found coreference. In cases where the coreference consists of multiple tokens, it replaces the said pronoun with multiple tokens. In other cases where the coreference is a multi-tokened entity, the system first checks if the coreference tokens have been merged. In that case, it compensates for this and only copies the first token from the coreference that is part of the entity. The system checks if the tokens have been merged by going through the list of entities found (the same list used by the NER). After the compilation of data from the coreference resolution sub module, the current HashMap that contains all the data of the current token is added to the sentence ArrayList. Once all tokens from the sentence are added, the sentence ArrayList is added to the Document ArrayList.

Within this sub module, there are two main managers, the English and the Filipino managers. The one discussed above is the English manager. The Filipino preprocess manager is very similar to that of the English. The only difference is that it skips on some of the preprocessing sub modules namely the POS tagger, the Noun Verb Chunker, and the Coreference Resolution sub modules as there are no stable tools for the Filipino language associated to these preprocessing steps.

5.4 Extractor

The extractor module identifies patterns found in a document using the extracted information from individual tokens as outputted by the preprocessor module. The system takes in a set of rules during instantiation and an input document. It then returns the found extractions (if there are any) along with its extraction type. Each rule accepted by this module is composed of one or more lookup tokens, where a lookup token has the syntax: "<lookup type:lookup value>". So if a rule were to have to look up, given that the "ner" of a particular token is "person", the corresponding lookup token is as follows "<new:person>".

Before extraction, the system first processes the input rules for extraction. Identifying patterns in a document is similar to that of a state transition diagram where the lookup tokens are the transitions. For every rule, the system generates $n + 1$ states, where n is the number of lookup tokens in a rule. The transition from the first state to the second state is the first lookup token. The transition of the second state to the third state is the second lookup token of the current rule, and so on. There are also some extra transitions that are used to support special cases such as the use of the Kleene star (*) in the rule (This is seldom used when this module is paired with the pattern extraction module).

Extraction is done one token of the input document at a time. Before extraction is done, the system first feeds the input document into a preprocessor manager. Then it starts the extraction. For every token in the document, it goes through all rules listed. It sends the current token being read along with its data to the rules. If the lookup data matches the value, then the current rule transitions to the next state and saves the current state. If it does not match, the current state is set back to the initial state. If a rule manages to transition from the initial state to the final state, it can be said that that rule's pattern is found in the document.

On the rules, there are also keys to identify which tokens need to be extracted and what extraction type it is. This is symbolized with the keyword "[as]" followed by the extraction type. For example, if we were to create a lookup token that extract a token with an NER of "person" as PERSON, the syntax for this lookup token is as follows: "<ner:person>[as]PERSON". Whenever a rule reaches the final state, it backtracks and identifies the tokens that correspond to the look up token with an extraction key ([as]). It then retrieves that data and saves it along with its extraction type.

Extraction continues until the end of the document is reached. Extraction is only done on every sentence, meaning rules cannot surpass one sentence. This means that at the end of a sentence, all the rules are reset and these current states are their initial states.

An example of a rule is as follows:

`<ner:DISASTER><string:in><ner:LOCATION>[as]LOCATION`

The example given will search for the pattern that is composed of a disaster followed by the string "in" then followed by a location. The location is then extracted. An example string that this pattern will accept is as follows:

"There is a fire in Caloocan"

Given the text, "fire" will be mapped to <ner:DISASTER>, "in" will be mapped to <string:in>, and "Caloocan" will be mapped to <ner:LOCATION> and will be extracted. The visualization of the rule given above as a state diagram is shown in Figure X.

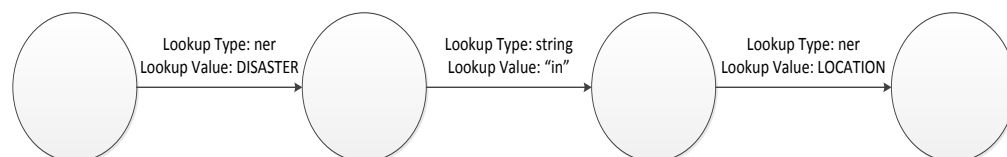


Figure 5.4: Visualization of Rule as State Diagram

5.5 Pattern Extractor

The pattern extractor module is in charge of identifying possible patterns that can be used for extraction. This tool takes in a set of documents and a set of seed words, and outputs a set of rules for possible extraction. Given a document, this system will generate patterns to extract words from the document that is present in the extractor seed words.

A document is any text that is related to the domain of the extractor. In the case of SOMIDIA, it is on the domain of disaster. An example document is as follows:

There is a fire in Caloocan

This system works by preprocessing the documents through the preprocessor manager. It then identifies the words in the document that are also present in the seed words which are the words to be extracted. For all words found, the system generates all possible rules from it. This is done by the system by going through all possible window setup and all possible combination of lookup tokens. Windowing is the term used to describe which section of the document is considered or is used for computation. Minimum window size, maximum left window size, and maximum right window size is also used. Minimum window size dictates the minimum number of tokens in a window. Maximum left window size dictates the maximum number of tokens to the left of the word to be extracted. Maximum right window size dictates the maximum number of tokens to the right of the word. After all possible rules is generated, it saves these rules into a HashMap and stores its occurrence count. The system then continues to the next word that is present in the list of seed words. This is done until the entire set of documents has been processed and a large number of rules have been generated.

After preprocessing, each token in the input is annotated with its corresponding preprocessed data (features). These data are the string, the POS tag, the entity type, and the phrase type. For a more comprehensive discussion on how the preprocessor processes the input document, refer to section 5.3: Preprocessor. The preprocessed document is stored in an array and at each index is a hashMap that stores the features of the token at the current array index. The preprocessed document can be illustrated as follows:

```
[
  [{string:There},{ner:none},{pos:ex},{phrasetype:none}],
  [{string:is},{ner:none},{pos:bez},{phrasetype:verb}],
  [{string:a},{ner:none},{pos:at},{phrasetype:noun}],
  [{string:fire},{ner:none},{pos:nn},{phrasetype:noun}],
  [{string:in},{ner:none},{pos:in},{phrasetype:none}],
  [{string:Caloocan},{ner:LOCATION},{pos:ner},{phrasetype:noun}]
]
```

Figure 5.5: Sample Raw Text

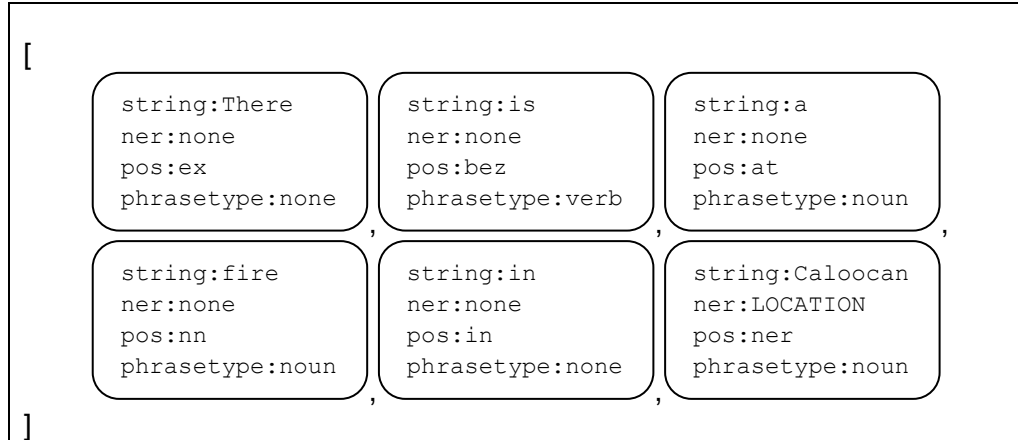


Figure 5.6: Graphical Illustration of Raw Text

Given the example above, the rules generated where the word to be extracted is “Caloocan”, a minimum window size is 3, a maximum window size on either side is 3, and “Caloocan” is part of the LOCATION class in from the word to be extracted list are as follows:

Code Listing 5.22

```
<string:a><string:fire><string:in><string:ANY>[as] LOCATION
<string:fire><string:in><string:ANY>[as] LOCATION
<string:fire><string:in><ner:LOCATION>[as] LOCATION
<ner:DISASTER><string:in><ner:LOCATION>[as] LOCATION
<string:a><string:fire><string:in><ner:LOCATION>[as] LOCATION
<string:a><ner:DISASTER><string:in><string:ANY>[as] LOCATION
<string:a><ner:DISASTER><string:in><ner:LOCATION>[as] LOCATION
<ner:DISASTER><string:in><string:ANY>[as] LOCATION
```

In the example, only the features string and NER are used. Since no word follows the word to be extracted, only the tokens to the left of the word to be extracted are used. This means that all the flanking tokens used in the rule is located to the left of the extraction token, or the extraction token is to the right most side of the rule. Because the max left window size is set to 3, only the 3 tokens to the left of the word to be extracted are used. This can be seen in the example above as the number of flanking tokens to the left of the extraction token will never exceed 3. The setting of max left window also means that the first 2 tokens of the document is not used, and as such, the generated rules never made use of the tokens “There” and “is”. Because the module permuted the size of the window, it can be seen in the example above that the number of tokens is not equal. The second, third, forth, and last rule is only composed of 3 token, where the rest is composed of 4. The lower count of the token is 3 because of the setting minimum window size which controls the minimum allowable tokens in a rule are 3. Unlike the minimum, the maximum number of allowable tokens is not explicitly set. It is computed by adding the maximum left window, the maximum right window, and one (1). In the case of the example, the maximum window is computed to a sum of seven (7). But since no word follows the word to be extracted, the maximum length of the rule

generated is only 4 (maximum left window size + 1 = 4). The set of features used for each rule is also permuted, This can be seen in the example in rule 1 and 5. The two rules are very similar except that the last token, the extraction token, is different in the two rules. In one rule, it uses the feature “string”. In the other, it uses the feature “ner”. This is because the module tries to run through all possible set of features of the input. But then again, the features POS and `phraseType` is not used in the example. Inspecting the other rules, it can be observed that the ner features is not used for all tokens. In the case of the first and third token to the left of their respective extraction tokens, the feature ner is never used. This is because looking at the preprocessed document, both of these tokens have an ner feature of none, meaning it is not associated with any entity type or the result is null since it is effectively an invalid results. Rules that contain these cases are removed automatically to save the integrity of the rules and to minimize the number of permutations to the processed. Other cases where rules are removed are where the feature `phrasetype` is “none”. Finally, in the example above, all extraction token is the token associated with the token Caloocan and all of it are extracted as the entity “LOCATION”. This is because for the example above, the word to be extracted is “Caloocan” and “Caloocan” is part of the LOCATION class from the list of words to be extracted.

After the rules have been generated, the system then minimizes and cleans the rules to increase the efficiency of the extractor. First the system minimizes the rules by removing all rules with an occurrence count of 1. Occurrence count is the number of times the rule is generated throughout the processing of the entire set of documents. The logic behind this besides the fact that it increases speed is rules with only one occurrence count tend to be overly specific rule or rules that will only work on a single case or on a very small percentage of cases. The system then feeds the remaining rules into an extractor and gives the same input document used by this module. Rules that are able to generate more extraction than the occurrence count of the said rule is removed. The logic behind this is that rules that generate more extraction are able to extract data that is unwanted, in other words these rules are too general and are capable of extracting unwanted data.

Table 5.16: Rules for minimizing pattern extractor general rule

Rule	Condition	Action
RemoveSingleOccurencePattern	If the number of times a pattern is generated is one (1)	Remove/delete the rule
RemoveGeneralRules	If the rule extracts an unwanted data (If the rule extracts more data than the number of times the rule is generate)	Remove/delete the rule

With the remaining rules, the system goes through the words to be extracted again and selects one rule that extracts that word. This is done for all the words to be extracted. The rule selected by the system is the one with the highest occurrence count. In some cases there are no rules that remain that extract the words to be extracted. In this case, that word is skipped. This is acceptable because the setup of that word would most likely be unique and has a low chance of being encountered again by the system.

An issue encountered in this module is in cases where the input documents or corpus has a small data set. The issue here is that most rules generated would only have an

occurrence count of 1. In this case, these rules will be removed during cleaning. To address this, the step to remove single occurrence rules can be skipped. This would not greatly decrease the pattern set's performance since the system would still prefer rules with higher occurrence patterns in the end. Another issue encountered is in some cases, there are words to be extracted with rules that tie. To address this, a preference score is added to the system. The preference score of a rule is determined by the lookup type used by the tokens of that rule. For every lookup token of that rule, it will receive either a positive or negative score. So in cases where rules tie, the rule with the higher preference score is preferred. The corresponding score of each type of lookup is determined by the user of the tool or by the one that instantiates this tool.

5.6 Validation

This module scores each report that was successfully extracted by the system. The input of this rule is the data extracted from the extractor module along with the original document. The system then adds additional data and sends it to the database.

Once an extraction is received by this module, it determines the longitude and latitude of the location of the extraction. This is done by querying the extracted location to the 'area' table of the database and retrieving its corresponding longitude and latitude.

After this, the system determines from who the report came from. This system accomplishes this task by querying the report poster ID of current report from the 'users' table of the database. The report poster ID can either be the facebook ID, the twitter ID, the mobile number, or the SOMIDIA ID of the poster of the original document. If a poster has registered for a SOMIDIA account, their respective IDs are stored and tracked by the system. The system uses these IDs to determine their corresponding service ID (somidia ID; ID for use of this project). When the poster is not registered to the service, the extraction score is given the default value.

After determining this, the system then retrieves the scores of the extraction. Scores are used to determine the reliability of the extraction. Two score are used for each extraction, the cluster score and extraction score. The cluster score is determined by the number of extraction with similar events or when extractions are relating to a single event. This is done by querying the number of reports similar to the current document on the 'reports' table. The group identifies if two reports are similar whenever their disaster type and disaster location are same and if the time reported between the two reports is within a predefined window span. The logic here is that the more a disaster is being broadcasted, the higher the possibility that it is true. To handle extractions that are copied throughout a social media or simply repost or retweets, these extraction's cluster score are reduced by 50%. This is done by the system by looking at the current extraction in the database (on the 'reports' table) and searching for an original input document similar to the current extraction of this module. The logic behind this is that reposts and retweets can easily flood the social media without it being helpful for validation since most people repost a message without validating it. After the cluster score is determined, the system increments the cluster score of the entries in the database that relates to the current extraction being processed by this module.

After determining who posted the report, the system determines the extraction score of the extracted report. The extraction score is determined by the credibility score of the poster at the time the system receives the extraction. This is done upon querying the author of the input document (step 2). At that time, system also retrieves and stores the author. The credibility score of any user is computed by the number of post that was

validated over the total number of post of that poster. The credibility score of the users does not update automatically, but must be explicitly called or executed.

Once all these data are collected, the extraction along with all the additional data is added into the database.

Table 5.17: Steps/Processes in validation

Step #	Step	Action
1	Get longitude and Latitude of the report.	Query the 'disaster location' on the 'area' table of the database and retrieve the corresponding longitude and latitude.
2	Get the poster (author) of report.	Query the 'report author ID' and the 'report type' to the 'users' table of the database and retrieve the corresponding serviceID (somidialID) and the credibility score of the user.
3	Get the cluster score of the report.	Query the number of reports on the 'reports' table in the database with similar disaster type, disaster location, and within a predefined time window from the report in question.
4	Get the individual score of the report.	Access the saved credibility score retrieved from the database during step 2.
5	Determine if report is a retweet and adjust individual score accordingly.	Query the original document of the report on the 'reports' table of the database. If a report is found with the same original document, the current report's individual score is reduced by 50%
6	Increment the cluster score of similar reports.	Update the reports on the 'reports' table in the database with similar disaster type, disaster location, and within a predefined time window from the report in question and increment each of their cluster score by 1.
7	Add extraction to the database.	Inserts the report along with its extraction and other data to the 'reports' table of the database.

Report score(X) = Cluster score(X) + Individual Score(X)

Cluster Score(X) = total number of related extractions within a time window.

Individual Score(X) = reputation of the reporter (Credibility score(Y)) at the time the report is extracted.

Credibility Score (Y) = (number of valid reports of Y - number of non-valid reports of Y) * 0.1 + 1

Note: valid reports are reports whose Report score reached a predefined threshold.

5.7 Visualization Engine

The visualization engine is made up of a series of JSP web pages that presents a map plotted with disaster events as well as supporting reports and documents of the events in question. The system leverages on the opensource maps made available by the OpenStreetMaps initiative.



Figure 5.7: Screenshot of SOMIDIA's Crisis Map

The engine is implemented using the Struts 2 MVC framework in order to facilitate communication between the frontend and backend systems. This framework was specifically chosen in due to its simple architecture resulting to great extensibility to other frameworks.

5.7.1 Loading and Pushing Data and Information

A Java service method for retrieving and loading information associated to a particular page being loaded is called via a script written in JavaScript. Depending on service method involved, parameters may or may not be passed. The method will retrieve the required data and information from a database and will return a JSON object.

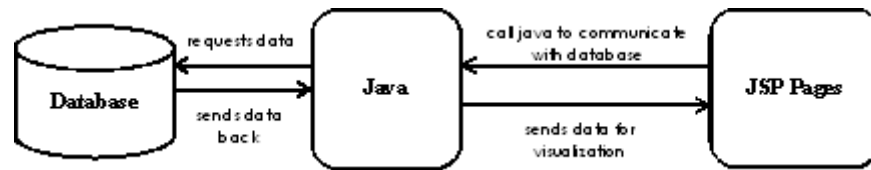


Figure 5.8: Illustration of Data Loading Process

The information contained within the returned JSON objects are then integrated into the website's user interface via Javascript functions.

Similar to loading data and information, pushing data via webforms from webpages (e.g. submitting reports online) are done by calling a Java service method with a script written in Javascript. The information inputted by the user will serve as the parameters that will be passed to the Java service method. If all input passes all input validation mechanisms, a java function will be called to execute a MySQL script specifically for adding/editing the data in the system's database.

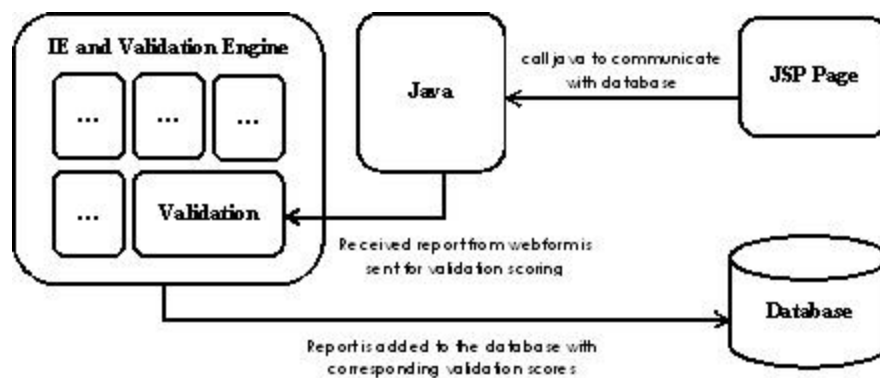


Figure 5.9: Illustration of Data Pushing Process

5.7.2 OSM Integration

OpenStreetMaps was chosen over Google Maps because the data used in the map were gathered with collaborative crowdsourcing thereby providing free geographic data such as street maps to anyone who needs them. The legal and/or technical restrictions and copyright issues on geographic data on Google Maps placed restrictions as to how the information they provide can be used. Another reason why OpenStreetsMap was used is because the streets of Metro Manila are more comprehensive in its map as compared to that of Google Maps.

The JSP pages communicate with OpenStreetMaps using the OpenLayers API. The OpenLayers API contains scripts written in JavaScript that will allow the displaying of maps in web browsers. The API functions by making use of layers. The map is always placed on the first layer. While succeeding layers are for map markers.

The map is often initialized using the code below wherein the instantiation of the OpenLayers Map requires the name of the "div" where the map will be displayed as a parameter. The actual map layer is then using the .addLayer() function of the API. Since there has already been collaboration with OpenLayers and OpenStreetMaps, a function

that will display the OSM map onto the layer is readily available. The following is done through the code snippet below:

Code Listing 5.23

```
map = new OpenLayers.Map("mapdiv");  
map.addLayer(new OpenLayers.Layer.OSM());
```

Markers are placed on a new layer on top of the map layer. Initializing a layer for markers is very similar to that of creating a layer for the map. Once a variable has been initialized as a marker layer, it is simply added to the original map layer as shown in the code snippet below:

Code Listing 5.24

```
var markers = new OpenLayers.Layer.Markers( "Markers" );  
map.addLayer(markers);
```

JavaScript functions have been created in order to place markers on the marker layer. In order for there to be markers in the marker layer, the markers need to be initialized with the information such as its supposed longitude and latitude position on the map, as well as the icon which will be used as marker. The marker is then added into the marker layer as shown in the code snippet below.

Code Listing 5.25

```
var mark = new OpenLayers.Marker(lonLat, icon);  
markers.addMarker(mark);
```


6.0 Results and Observations

This chapter presents observation and results of the evaluation made on the back-end of the system. The group tested the system on a total of 23,969 tweets gathered from different sources, which were manually annotated by the group.

6.1 Data Gathering

The group primarily focused on gathering data from Twitter to test its efficiency and accuracy on text from Social Media. Twitter provided a less restrictive domain of privacy compared to Facebook and so allowed the group to collect sufficient data for testing.

A total of 21501 tweets using an earlier version of the system's Twitter Crawler wEere collected. The version of the crawler used in gathering tweets listened and collected tweets from the Twitter Account created by the user much like the current version but it does not classify whether or not it is related to disaster nor does it detect the language used in the tweet. The crawler was deployed in the Center for Language Technology (CeLT) laboratory and collected an estimate of 20 000+ tweets a day. The tweets used for testing were from the tweets collected during the events of July 3, 2012, where a Low Pressure Area (LPA) passed through the Philippines causing floods and other related events throughout Metro Manila.

A number of tweets from an external source of data other than the system's crawler were also collected to compensate for errors that may come up for the limitations of its crawler. Data from Topsy, which is an online repository of tweets since 2008, were gathered. Topsy does not allow for users to listen to incoming tweets unlike the Twitter API but it provides an API, Otter API which is a REST-ful API, for searching tweets in their repository. The group collected 2468 tweets from Topsy using the keywords "baha" and "flood" along with the different cities in Metro Manila for example, "baha manila" and "flood makati". These were the keywords used so that it will match the type of disasters that the group expected to retrieve from the crawler.

A total of 23 969 tweets in total were collected for testing the system's module. These tweets were manually annotated by the group for them to compare the expected results with the actual results of the system. The collected tweets were annotated with 5 features namely: (1) if the tweet is disaster related or not, (2) if the language used is English or Filipino, (3) the type of disaster the tweet is referring to, and (4) the location that the disaster refers to. A tool in annotating the tweets was created in order to speed up the process of annotation.

6.2 Evaluation of Crawler Module

The Crawler module's primary objective is to provide the system with tweets that are disaster-related and to determine the language of these tweets in order to use the most suitable extraction process for the tweet. Thus, the accuracy and effectiveness of the crawler's ability to classify if the tweet is a disaster or not and determine the language of the tweet were evaluated through the use of different measures, particularly Precision, Recall and F-Measure.

6.2.1 Classification of Disaster

One of the objectives of the crawler module, which is to determine if the article is disaster-related, can be seen as a classification problem since the answer to the

question is a simple “Yes” or “No”. Following that premise, the group evaluated the module using the terms **true positives (TP)**, **true negatives (TN)**, **false positives (FP)**, and **false negatives (FN)** to determine the precision, recall and f-measure of the module which is common in classification tasks.

Methodology

The module was evaluated by comparing the manually annotated list of tweets to the classified tweets of the system in the two dataset collected. The results were then labeled as true positive if both the manual and automatic annotation classified the tweet as a disaster, true negative if both classified the tweet as not a disaster, false positive if it was classified by the system as a disaster but manual annotation classifies it a not a disaster, and false negative if it was classified by the system as not a disaster but manual annotation classifies it as a disaster. Table 6.1 shows some examples depicting each label of result.

Table 6.1: Sample of classification results

Tweet	Expected Result	Actual Result	Label
"Floods hit Espana, Quiapo, and Malate areas in Manila because of the heavy rains,	Yes	Yes	True Positive
@DepEd_PH may pasok ba?	No	No	True Negative
@PTVph has an early morning weather newscast.	No	Yes	False Positive
@rsgaad Edsa P tuazon Tunnel - Gutter deep	Yes	No	False Negative

The total number of true positives, false positives, and false negatives were then used to compute the precision, recall and f-measure of the system. In classification tasks, precision is the percentage of true positives to all classified positives.

$$\text{Precision} = \frac{tp}{tp + fp}$$

Recall on the other hand is the percentage of true positives to all elements that should have been classified as positives.

$$\text{Recall} = \frac{tp}{tp + fn}$$

The traditional F-Measure is the weighted harmonic mean of precision and recall because it provides a measure that balances between precision and recall.

$$F_{\beta} = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

Results and Observations

The initial results of the system evaluation can be seen in Table 6.2.

Table 6.2: Initial Results of Evaluation

Source	Articles	TP	FP	TN	FN	Precision	Recall	F1-Measure
Crawler	21501	577	5070	15755	98	10.22%	85.48%	18.25%
Topsy	2468	1144	1298	13	13	46.85%	98.88%	63.57%
Total	23969	1721	6368	15768	111	21.28%	93.94%	34.69%

Table 6.2 shows that the system has a particularly low precision score which is the effect of having an absurd amount of false positives in the classification. The low precision score greatly affected the F-measure of the system despite the system having a high recall score. The large number of false positive classification can be attributed to the quality of trigger words used by the system to determine if the tweet is disaster or not. The list of trigger words used by the system was generated by using words that have a high TFIDF score in terms of disaster news articles, which means that the words are commonly used in disaster news articles. However, it must be taken into consideration that the words generated are under the domain of news articles, which are usually long stories unlike tweets that are restricted to 140 characters. The limited words in tweets mean that the words used by people in twitter to convey disasters are different from how news articles are written. Examples of such tweets are shown in Table 6.3.

Table 6.3: Sample false positive tweets due to TFIDF words

Tweet	Trigger word located
Some of you are lucky I don't know where you live.	Lives
@papaJunie as of now wala eh, unbeatable talaga si Sir sa buhay ko. :))	buhay
@ABSCBNNews @ABSCBNNewsLife IT'S ABOUT TIME!!!	time
@Robin_Benitez you may call Manila Traffic Hotline 527-3087. #mmda	mmda
@dost_pagasa update!!!!	pagasa

Another set of trigger words were used to see if the accuracy of the crawler can be improved. They made use of the words used in the extractor seed words, which are a list of words that are synonymous to disasters that the system handles along with the different permutations of the words. From a list of 23 486, the group reduced the trigger words to a mere 892 words. Table 6.4 shows the result of the experiment.

Table 6.4: Result of evaluation using extractorSeedWords as Trigger words

	Articles	TP	FP	TN	FN	Precision	Recall	F1-Measure
Crawler	21501	468	968	19857	207	32.59%	69.33%	44.34%
Topsy	2468	1139	1289	22	18	46.91%	98.44%	63.54%
Total	23969	1607	2257	19879	225	41.59%	87.72%	56.43%

The results presented in Table 6.4 show that by changing the trigger words, the precision of the system improved in the crawler tweets dataset. From a 10% precision, it improved to a 32.58% precision rate; improving the overall f-measure from 18.25% to an f-measure of 44.34%. The results from the Topsy dataset did not particularly improve because of the nature of how these tweets were collected. They were collected using a search on terms like “baha” and “flood”, which means these terms are found in almost every tweet thereby reducing the variety of tweets that were collected. The improvement on the precision from the crawler dataset however cost the system a drop in recall. This is because the quality of classification would depend on the words present in the list, requiring all possible permutations of the words if one wanted a better classification. Table 6.5 shows some of the false negative classification that resulted in the new trigger word list. This problem however is easily remedied by adding new words to the trigger words list. Table 6.6 shows the improvement on the results by adding words that Twitter users commonly used in conveying disasters while Table 6.7 shows the words added to the trigger words. This shows that just by adding a few more permutation of the given words, it would greatly improve the recall of the system.

Table 6.5: Sample false negative due to trigger words restriction

Tweet	Disaster word in tweet	Disaster word in trigger word
TOP HIRIT Ilang sasakyan. tumirik matapos lumusong sa mga binahang kalsada sa QC.	binahang	Baha, bumabaha, mabaha, binabaha
@kuyakim_atienza Ganyan din po dito sa Sta. Mesa, Manila :(Mabilis pong bumaha!	Bumaha	Baha, bumabaha, mabaha, binabaha

Table 6.6: Results of evaluation after adding a few words

	Articles	TP	FP	TN	FN	Precision	Recall	F1-Measure
Crawler	21501	609	1261	19564	66	32.57%	90.22%	47.86%
Topsy	2468	1139	1289	22	18	46.91%	98.44%	63.54%
Total	23969	1748	2550	19586	84	40.67%	95.41%	57.03%

Table 6.7: Words added to the list of trigger words

Filipino		English		
bahaing	malakas ang ulan	bad weather	rainy	Rainshowers
bahang	masamang panahon	floodings	strong rain	
binabagyo	maulang	floods	thunderstoms	
binaha	pagbaha	heavy rain	thunderstorm	
binahang	pag-ulan	heavy rain fall	unyielding rain	
bumabaha	umulan	heavy rains	rain showers	
inulan	bumaha	non stop rain	raining	

This shows that just by adding a few more permutation of the given words, it would greatly improve the recall of the system which rose from 87.72% to 95.41%. The F-measure of the system also improved albeit only little, 56.43% to 57.03%. Despite that the increase in precision and recall through the change and addition of trigger words, the precision is still particularly low bringing the entire f-measure down. This is because of the limitation of the system to only use a keyword-based search in classification. This allows room for errors in classification such as tweets like questions, sentiments, and other tweets unrelated to reporting a disaster but mentioning the keywords used. This was the primary reason for false positive data classified by the system. Table 6.8 shows sample false positive tweets that show such examples.

Table 6.8: Words added to the list of trigger words

Tweets	Reason
@MMDA flood situation to kamuning and e.Rodriguez please?	Questions
@MMDA good morning. baha ba sa any part of Makati? Kalayaan? Makati Ave? Paseo? de la rosa?	Questions
Driving myself to Pasay.. No traffic, no flood, Coldplay on the radio. :D	Negative Events
@JanieOctia wala po ng reported na flooded area sa C5 :)	Negative Events
RT @inquirerdotnet: Marikina starts dredging to avoid another 'Ondoy' flood http://bit.ly/9EV6X1	Unrelated Information
PNoy visits Marikina flood victims http://bit.ly/iDy25b #news	Unrelated Information

RT @harbido: One of the worst things in the Philippines - Manila flood. Maitim, malapot, masangsang. :\ 	Sentiments
OMW to Makati for @MetroMagPH shoot! Sana wala ng baha.	Sentiments
@LonieMe @MrGenerous143 Don't you ever try to be one. Haha. Haters will flood.	Different Meaning of the word
Oyes, buttercookies. Parang nasunog yung ilalim eh. XD pero di naman lasang sunog.	Different Meaning of the word

As the table shows, there are 5 types of information that the system usually classifies as false positive. The first is tweets that are portrayed as questions about disasters. These tweets are one of the most common and at the same time harmful to the system as they contain all the relevant information that the system extracts such as a disaster type and a location, but they are often not validated or true information. The second type is tweets that say that there is no disaster at a specific location. Much like the first type, this type often makes a mistake of classifying these tweets as disasters when they are actually saying otherwise. Though these two types of tweets may be used in the future as another way to validate tweets, they are currently generating noise in the system due to the system not understanding their purpose. The other 3 types of information, such as unrelated information, sentiments and different meaning of the trigger words, composes majority of the false positive tweets that the system classified. These types of tweets are difficult to filter out using keyword based search, especially in the twitter dataset which needs a small threshold of trigger words because of its limited number of characters.

6.2.2 Classification of Language

One of the objectives of the crawler module is to determine the language used by the article. This can also be seen as a classification problem since the answer to the question can only be either “English” or “Filipino”. Following that premise, the classification of the language used was also evaluated using the terms **true positives (TP)**, **true negatives (TN)**, **false positives (FP)**, and **false negatives (FN)** to determine the precision, recall and f-measure. The difference in the evaluation of the classification of disaster and language is that the measurements of the classification of both categories were recorded in the classification of the language, while only the classification of the positive group (“Yes”) were recorded in the classification of disaster.

Methodology

The classification of the language was evaluated by comparing the manually annotated tweets and the automatically annotated tweets, whose classification of the disaster resulted to a “True Positive”. This methodology allows for an unbiased result of the language classification as it removes the errors that may propagate through the misclassification of the disaster since the classification of the language would also depend on the classification of the disaster. Both the accuracy of the classification of both “Filipino” and “English” were evaluated similar to how the classification of disaster was evaluated.

Results and Observations

Table 6.9 shows the results of evaluation of “English” classification while Table 6.10 shows the results of evaluation of “Filipino” classification.

Table 6.9: Evaluation of “English” Classification

	Articles	True Positive	False Positive	True Negative	False Negative	Precision	Recall	F1-Measure
Crawler	430	424	26	153	6	94.22%	98.60%	96.36%
Topsy	827	679	208	104	148	76.02%	82.10%	79.23%
Total	1257	1103	234	257	154	82.50%	87.75%	85.04%

Table 6.10: Evaluation of “Filipino” Classification

	Articles	True Positive	False Positive	True Negative	False Negative	Precision	Recall	F1-Measure
Crawler	179	153	6	424	26	96.23%	85.47%	90.53%
Topsy	312	104	148	679	208	41.27%	33.33%	36.88%
	491	257	154	1103	234	62.53%	52.34%	56.98%

The “English” classification has a higher accuracy rather than the “Filipino” as shown by the two tables above given the 85.04% f-measure to 56.98% f-measure. However when only the dataset of the SOMIDIA crawler is observed, there is a closer results in terms of f-measure with 96.36% in “English” and a 90.53% in “Filipino”. The Topsy dataset have a low f-measure in both dataset compared to the SOMIDIA dataset given the 79.23% in “English” and a 36.88% in “Filipino”. These scores can be attributed to the quality of data that the Topsy dataset contains. It contains plenty tweets where the language used is actually English but contains hashtags of “baha” to be included in searches. Another type of tweet that the module misclassifies is when an “English” tweet is retweeted and a Filipino phrase is added. The important part is English but since there is a low threshold for it to be classified as Filipino, it is classified as “Filipino” instead.

6.3 Evaluation of the Information Extraction Module

The main goal of the Information Extraction module is to extract relevant information, namely the date, the time, the location, and the disaster type, of a particular disaster event report. The information extraction component of the system is adaptive and makes use of machine learning to generate patterns that will be used for information extraction. Since the date and time of reports can already be retrieved based on the timestamp of a particular report, the extraction module only extracts the location and disaster type.

The model for information extraction generated by the system’s machine learning module was tested and evaluated using 10-fold cross validation with the data gathered discussed in Section 6.1 as the data set. Furthermore, the accuracy and effectiveness of the information extraction engine’s ability to extract relevant information from disaster-related tweets were also evaluated with the use of different measures, particularly Precision, Recall, and F-Measure.

6.3.1 Methodology

From the list of manually annotated tweets, a total of 1500 tweets, 670 from the crawler and 830 from Topsy, was retrieved and used as the data set. After retrieval, the gathered tweets were shuffled using Java's Collections.shuffle() in order to ensure that the training set which will be used for model learning is a true reflection of the data mix found in the testing set. The shuffled data are then divided into ten (10) equal sized groups.

After preparing the data set, ideal machine learning settings that will be used for the model learning was identified. Using the 90-10 testing, where 90% of the data set is used for training and 10% is used for testing, a number of window setting were experimented upon. They then chose the window settings which showed the most promise based on the results of the experiments. The experiment is done on the Topsy and Crawler datasets individually.

Table 6.11: Results of experiments in determining best window settings for machine learning

Data Set	# of Tweets	Window Setting (mL-mR-m)	Correctly Extracted Disaster	Correctly Extracted Location	Correctly Extracted Disaster and Location
Topsy	83	3-3-3	47 (56.63%)	60 (72.29%)	43 (51.81%)
		2-2-2	52 (62.65%)	59 (71.08%)	42 (50.60%)
		3-3-2	51 (61.45%)	59 (71.08%)	42 (50.60%)
		4-4-2	50 (60.24%)	59 (71.08%)	43 (51.81%)
Crawler	67	3-3-2	20 (29.85%)	23 (34.33%)	-
		2-2-2	22 (32.84%)	24 (35.82%)	-

Window Setting:

mL – maximum left window size; **mR** – maximum right window size; **m** – minimum window size

From the results summarized in Table 6.11, it shows that the setting 2-2-2 works best for both the Topsy and the Crawler datasets. For the Topsy dataset, the 3-2-2 setting was chosen over the 2-2-2 despite the latter having higher performance scores. This decision is due to the former setting's ability to extract information more completely as compared to the latter as shown in Table 6.10 and 6.11. As for the Crawler dataset, the group opted to use the 2-2-2 due to its higher performance rating as compared to 3-2-2. The completeness of the extraction using both settings did not vary much so performance was used as the sole basis.

Table 6.12: Sample Data from Topsy Dataset

Tweet:	Baha n nm n "@ANCALERTS: FLOOD UPDATE: EDSA Quezon ave towards Trinoma - tire deep, http://t.co/ipEJp3L via @MMDA"
Annotated Disaster:	Flood

Annotated Location:	EDSA Quezon ave towards Trinoma
----------------------------	---------------------------------

Table 6.13: Extraction results of data in Table 6.10 using 2-2-2 and 3-2-2 Setting

2-2-2 Setting	
Extracted Disaster	Flood
Extracted Location	[EDSA]
3-2-2 Setting	
Extracted Disaster	Flood
Extracted Location	[EDSA, Quezon]

After determining the ideal settings, 10-fold cross validation was performed. Machine learning was used to generate a model trained nine out of ten of the subgroups of the original data set with the last remaining group being the test set. This process will generate a set of rules for extraction as output and will be used to run the extractor on the test set. The extracted results are then saved into a file. The process is repeated ten times until all ten subgroups have been used as a test set.

The quality of the generated rules is evaluated by determining the accuracy of the extraction process as the performance of the extraction module heavily relies on them. The number of correct extractions that the system was able to accomplish was counted. The results of the extraction were manually compared against the annotated data. Extractions that contain partial-data of the annotated ones are identified as positive instances and negative instances otherwise.

The accuracy of the information extraction engine is further evaluated by computing the precision, recall, and f-measure of the system. In order to compute for these, the number of relevant documents as well as the retrieved documents had to be determined. In the context of system, relevant words are the number of words that ideally needs to be extracted. On the other hand, retrieved words refer to the actual number of words that were extracted regardless of relevance.

In information retrieval tasks, precision is defined as the percentage of relevant words that have been retrieved in a given extraction.

$$\text{precision} = \frac{|\{\text{relevant words}\} \cap \{\text{retrieved words}\}|}{|\{\text{retrieved docun words}\}|}$$

On the other hand, recall is the percentage of relevant words that have been successfully retrieved.

$$\text{recall} = \frac{|\{\text{relevant words}\} \cap \{\text{retrieved words}\}|}{|\{\text{total relevant words}\}|}$$

And lastly, the traditional F-Measure is the weighted harmonic mean of precision and recall because it provides a measure that balances between precision and recall.

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

6.3.2 Results and Observations

Seven separate tests using different settings and dataset combinations were done in order to fully evaluate the system's machine learning and information extraction module.

For all of the tests ran, all tweets that have both correct and incorrect extractions for the disaster and location were counted. Results for the disaster and location were done separately, but tweets with both correct disaster and location extractions were counted as well.

It is inevitable for the dataset, coming from Twitter, a social networking site that does not impose a specific structure upon users when posting, to contain records that do not have complete information. And despite being incomplete, some of these tweets, like those shown in Table 6.14, are still considered as disaster related. The usual case is that disaster related tweets, mostly those written in Filipino, do not have the location indicated. Thus, the reason for having columns for disasters and locations blank annotations separate from those with annotations as seen in Table 6.15.

Table 6.14: Sample Disaster-Related Tweets with Incomplete Data

Tweet	Disaster	Location
Baha na naman.	Baha	n/a
Hi @kuyakim_atienza ! What's the cause of the heavy rains? May bagyo po ba?	Heavy rains	n/a
LAKAS NG ULAN!!! BAHHA NA!	Baha	n/a

The first test was done on the entire Topsy dataset using the 3-2-2 window setting. 54.73% of all the tweets contained within the dataset have had both their disaster type and location correctly extracted using this setup. It was also able to correctly extract the location of 76.87% and the disaster type of 62% of the tweets.

The second test was done on the English tweets contained within the Topsy dataset using the 3-2-2 window setting. The filtering out of non-English tweets was done during the preprocessing step of the data set. 71.21% of all the tweets contained within the dataset have had both their disaster type and location correctly extracted using this setup. It was also able to correctly extract the location of 87.24% and the disaster type of 77.24% of the tweets.

The third test was done on the Filipino tweets contained within the Topsy dataset, again using the 3-2-2 window setting. Much like the second test, non-Filipino tweets were filtered out during the preprocessing step of the dataset. Only 28.4% of all the tweets contained within the dataset have had both their disaster type and location correctly extracted using this setup. It was also able to correctly extract the location of 53.2% and the disaster type of 40.4% of the tweets.

The fourth test was done on the entire Crawler dataset using the 3-2-2 window setting. 19.40% of all the tweets contained within the dataset have had both their disaster type

and location correctly extracted using this setup. It was also able to correctly extract the location of 57.46% and the disaster type of 32.99% of the tweets.

During the identification of the ideal window setting for the machine learning module, the window setting 2-2-2 proved to be more accurate. However, the window setting 3-2-2 was able to extract more complete information. In addition, the difference between the two settings was not that high. As such, the Crawler dataset was also tested using the 2-2-2 window setting. 14.02% of all the tweets contained within the dataset have had both their disaster type and location correctly extracted using this setup. It was also able to correctly extract the location of 58.21% and the disaster type of 35.37% of the tweets.

The sixth test was done on the English tweets contained within the Crawler dataset using the 3-2-2 window setting. The filtering out of non-English tweets was done during the preprocessing step of the data set. 24.68% of all the tweets contained within the dataset have had both their disaster type and location correctly extracted using this setup. It was also able to correctly extract the location of 60.64% and the disaster type of 50% of the tweets.

The seventh test was done on the Filipino tweets contained within the Crawler dataset, again using the 3-2-2 window setting. Much like the second test, non-Filipino tweets were filtered out during the preprocessing step of the dataset. Only 12.38% of all the tweets contained within the dataset have had both their disaster type and location correctly extracted using this setup. It was also able to correctly extract the location of 24.21% and the disaster type of 52.11% of the tweets.

The summary of the results of the model evaluation can be found on Tables 6.15 and 6.16.

Dataset	Disaster					Location				
	Annotated Blank		Annotated Non-Blank			Annotated Blank		Annotated Non-Blank		
	Correct Extraction	Incorrect Extraction	Correct Extraction	Incorrect Extraction	Extracted Blank	Correct Extraction	Incorrect Extraction	Correct Extraction	Incorrect Extraction	Extracted Blank
Topsy (English & Filipino)	2	1	513	16	298	9	3	629	11	178
Topsy (English)	1	0	447	11	121	2	0	504	15	59
Topsy (Filipino)	1	0	100	13	136	5	4	128	15	98
Crawler (English & Filipino) [3-2-2]	22	4	199	24	421	154	13	231	23	249
Crawler (English & Filipino) [2-2-2]	22	4	215	25	404	154	14	236	24	242
Crawler (English)	13	4	222	48	183	62	28	223	58	99
Crawler (Filipino)	5	4	41	8	132	64	8	35	13	70

Table 6.15: Results of Model Evaluation (Part 1)

Table 6.16: Results of Model Evaluation (Part 2)

Dataset	Correct Extraction of Disaster AND Location	Total No. of Tweets	Score (Excluding Blank Data)		Score (All)	
			Disaster	Location	Disaster	Location
Topsy (English & Filipino)	446	830	62.03%	76.89%	62.05%	76.87%
Topsy (English)	413	580	77.20%	87.19%	77.24%	87.24%
Topsy (Filipino)	71	250	40.16%	53.11%	40.40%	53.20%
Crawler (English & Filipino) [3-2-2]	130	670	30.90%	45.92%	32.99%	57.46%
Crawler (English & Filipino) [2-2-2]	94	670	33.39%	47.01%	35.37%	58.21%
Crawler (English)	116	470	49.01%	58.68%	50.00%	60.64%
Crawler (Filipino)	15	190	22.65%	29.66%	24.21%	52.11%

It was observed that even though the system is able to correctly extract disaster event types and locations separately, the number of correct extractions of both the disaster type and location is significantly lower. In Table 6.16, it can be seen that the amount of location extracted from the tweets are greater than that of the disaster type it can extract. Obviously, reports with correctly tagged locations can still be plotted on the map, while those with only correct disaster type extractions cannot. And to know that a tweet has a location means that it has great potential to contain valuable information as compared to those tweets where you can only get the disaster type. It came to a decision of using the naïve key word searching in order to try and see if the tweet with the location is a possible source of disaster information. What the keyword search does is that among those tweets with extracted locations, it will run through the list of disaster words again used by the crawler in order to try and see if there is disaster type information that is missed during the extraction process. And as a result, all of the datasets had an average of 17.53% increase in their total score as compared to only relying on the extraction module of the system. This has increased the overall performance of the extractor, thus preventing possible information loss. The results summary of using naïve keyword search can be found on Table 6.17 and 6.18.

The information extraction module was also tested using another domain. A dataset containing reports of fire that were gathered using Twitter search was used to check how the extraction module would fare given a new domain. The results of the test are found on Table 6.17 and 6.19.

Table 6.17: Results of Evaluation with Naïve Keyword Searching

Dataset	Without Naïve		With Naïve	
	Tweets	Percent	Tweets	Percent
Crawler	96/463	20.73%	192/463	42.12%
Crawler 2	88/463	19.00%	197/463	42.55%
Crawler English	111/352	23.62%	178/352	50.57%
Crawler Filipino	13/105	12.38%	29/105	27.62%
Topsy	435/801	54.31%	597/801	74.53%
Topsy English	401/562	71.35%	485/562	86.30%
Topsy Filipino	71/239	29.71%	114/239	47.70%
Twitter Fire English	614/1048	58.59%	861/1048	82.16%

Twitter Fire Filipino	356/628	56.69%	435/628	69.27%
-----------------------	---------	--------	---------	--------

Table 6.18: Results of Evaluation using Performance Measures for IE

Dataset	Disaster and Location		
	Precision	Recall	F-Measure
Topsy (English & Filipino)	95.43%	44.34%	60.55%
Topsy (English)	94.33%	56.44%	70.62%
Topsy (Filipino)	81.82%	35.87%	49.6%
Crawler (English & Filipino) [3-2-2]	86.39%	25.30%	39.14%
Crawler (English & Filipino) [2-2-2]	84.18%	28.33%	42.39%
Crawler (English)	68.48%	41.06%	51.34%
Crawler (Filipino)	60.99%	20.28%	30.44%

Table 6.19: Results of Evaluation (Fire Dataset) using Performance Measures for IE

Dataset	Disaster			Location		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure
Twitter (English)	84.58%	67.65%	75.17%	93.70%	47.26%	62.83%
Twitter (Filipino)	91.99%	74.18%	82.13%	88.86%	41.28%	56.32%

Comparing the performance results of the tests of those done on the Topsy dataset versus the Crawler dataset, it clearly shows that the tests on the former had higher performance rates. This could be attributed to the nature of data collection process that was used to gather data for the Topsy dataset. The data set collected is more aligned as compared to that of the data sets that were crawled using the crawler module. This enabled the machine learning module to generate better rules for the dataset; thereby generating higher scores as compared to rules generated using the crawler dataset. One of the evidences for this is the number of rules that were generated during the model learning of the two datasets. With the crawler extracted dataset, an average of 677 rules were generated from a dataset with a size of 603. While with the Topsy dataset, an average of 566 rules were generated from a dataset with a size of 747. These results show that in Topsy, the machine learning module managed to generate rules that are capable of extracting a wider set of data compared to that of the crawler which led to the higher performance ratings of the Topsy dataset over that of the crawler.

Filipino and English data within the dataset were tested separately, as well as testing them as a whole. When run through the machine learning module, the experiment using the English dataset scored higher than the combined dataset (English and Filipino). However, the opposite happened with the Filipino dataset. The experiment using the Filipino dataset scored significantly lower as compared to the combined dataset. The

results of the experiments using the combined dataset placed in the middle of the two datasets. It can be said that the English data contained within the entire dataset has somehow pulled the collective score up. The big pull of the English dataset can be attributed to the fact that there are two times more English texts than Filipino texts. Given these, the use of the English and the Filipino datasets trained separately is preferred as opposed to the machine learning module trained on the English and Filipino datasets combined.

Comparing the results of the experiments using the English and the Filipino dataset, the one using the English dataset performed significantly better than the one the used the Filipino dataset. This can be attributed to the difference of the preprocessing between the two databases. This greatly affected the performance of the machine learning module. The difference between the two datasets is that the English one underwent preprocessing that made use of additional tools such as POS tagger and noun verb chunker. This enabled the machine learning module to utilize more features in terms of generating rules which improved its capabilities. Although these data are based on the empirical fata derived from the results of our test, this analysis is still not absolute. This is because the dataset collected for the Filipino language was relatively small as compared to the English dataset.

One limitation of the system is the lack of dedicated preprocessing tools for information extraction for the domain being worked on particularly those in the context of Filipino and micro texts. This limitation is most evident in the named entity recognizer module of the system. The task of the NER is to identify named entities in order for tokens to be tagged as such, and to combine or consolidate multiple tokens that refer to a single named entity. Due to the limitation of the NER currently available, the system was unable to identify named entities properly which affected the effectiveness of the rules the machine learning module was able to generate. And again, this affected the performance of the extractor module.

An example showing the limitation of the NER is as follows:

Given the tweet: “baha sa Sta. Cruz Manila”, The system should ideally generate the following rule: <string:baha> <string:sa> <ner:LOCATION>. But due to the limitation of the NER, the system was not able to identify that Sta. Cruz was actually part of Manila and as such should be consolidated as one token. Due to the limitation of the NER, the rule generated was: <string:baha> <string:sa> <ner:LOCATION> <ner:LOCATION>. Comparing the two rules, the second rule is more specific compared to the first. This contributed to lower score of the extraction modile. This is because the data that will go extraction should first conform to a stricter format in order to lead to successful extraction.

The limitation of the NER, specifically the lack of its capability to identify complex entities, such as Sta. Cruz Manila, poses another problem. The NER fails to consider “Sta. Cruz Manila” as a single entity and as a result, it is not consolidated into one token. This is the main reason for the sometimes incomplete extractions performed by the information extraction module. Instead of extracting “Sta. Cruz Manila” as a whole entity, the system will only be capable of extracting a part of it (e.g. Sta. Cruz or Manila). In some cases, the machine learning module is able to adapt to this limitation and will be capable of extracting both of these location. However, it will still be classified as two separate locations which they are not. Although the machine learning module was somehow able to adapt to these limitations, it is still not a solution to the problem with NER. For the NER to be able to extract both of the data, it has to generate a relatively

specific rule, which would only be useful on a small subset of the actual data. This is not ideal for information extraction. Although the ML was somewhat able to adapt to these limitations, it is still not a solution to the problem of the limitation of the tools. This is because, for the NER to extract both of the data, it has to generate relatively specific rule, which would only be useful on a small subset of an actual data, which is not ideal for information extraction.

Aside from disaster reports gathered from social networking site Twitter, the information extraction module was also tested using a dataset gathered from news sites. The results of the said test can be found on Table 6.20.

Table 6.20: Results of Evaluation (News Articles) using Performance Measures for IE

Dataset	Disaster and Location		
	Precision	Recall	F-Measure
News (English)	38.84%	54.63%	45.40%
News (Filipino)	68.25%	27.13%	38.82%

Comparing the results of the News extraction to the Twitter extraction, it can be observed that the Twitter dataset clearly has a better performance over its news counterpart. This is because of the nature of the Pattern Extractor Module, which tries to identify sequential patterns in the document. The Pattern Extractor is better for short text such as tweets as compared to long text such as news articles, since the significant features in short text that the tool leverages would most likely be located within range of the word to extract. This makes it easier for the tool to derive that pattern. The length of the shorter text also allows for a fewer permutations of patterns possible for the tool to generate, making it simpler for the tool to identify pattern and enables the tools to do a more extensive search compared to long articles. The significant features in long text might be located farther away from the word to extract, or it might be out of reach from the word to extract. This would mean that the tools would fail to create the proper pattern to extract the word. The length of the text also would not allow the system to do a more extensive search because of the large number of permutations that would take a great amount of resources and time to find the ideal patterns with an extensive search.

6.4 Evaluation of Information Validation

The main goal of the Information Validation module is to be able to validate, score, and rank all disaster-related text articles that the information extraction module successfully processes. A report scoring scheme and a user reputation model were implemented in order to assess the validity of reports. The information validation module was evaluated by making use of a simulation.

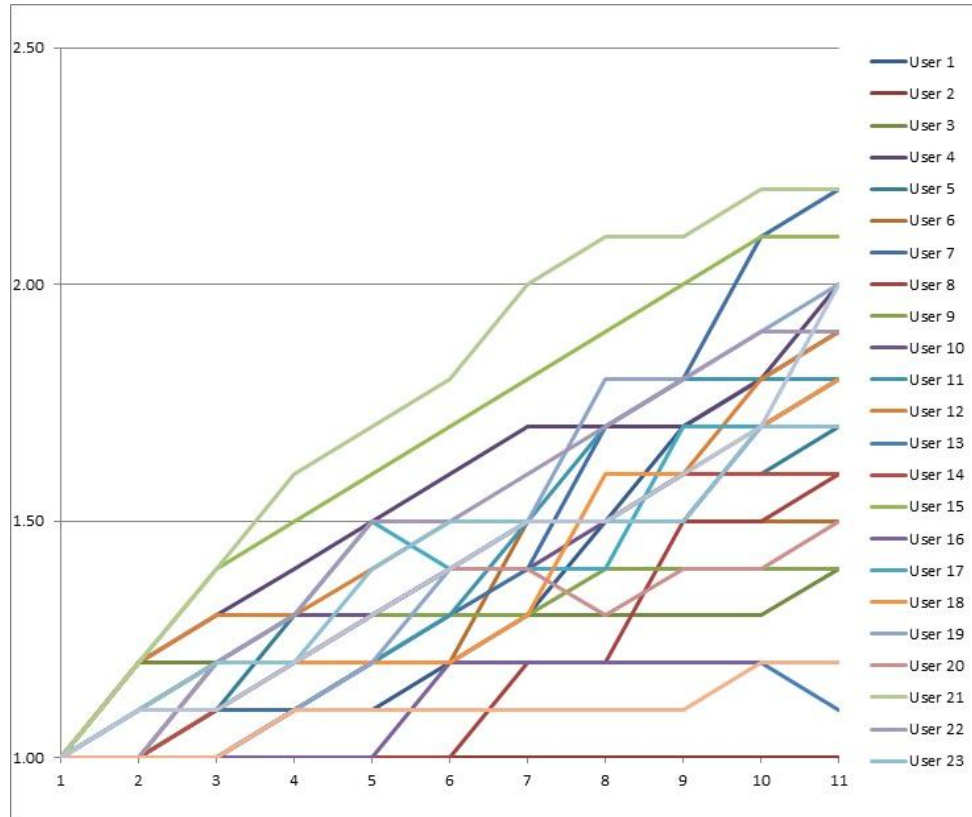


Figure 6.1: Illustration of the Entire User Reputation Model Simulation

6.4.1 Methodology

The validation module was tested through the use of a simulation. The whole simulation of the validation process includes data preparation, data simulation, and data collection.

In the data preparation phase, 200 inputs spread equally into 10 batches with 25 users in total were used. The inputs consist of the user id, disaster location, disaster, batch number for each entry. Each user id for the inputs are randomly selected in order to simulate a real world scenario where random users tweet about random things happening around them. The disaster location and the disaster itself are manually entered in order to simulate a variation of writing styles, on how each user reports a disaster. The batch number simply denotes the batch number the entry is in. 0-2 noise entries in every batch were also added in order to represent bad inputs caught by the system.

After preparing the data to be validated, it is now then fed to the validation module of the system. The validation system does the computation of the credibility of the user and then is stored in the database. To ensure that the system will acknowledge the separation between batches, the system calendar is set one month ahead in between processes.

6.4.2 Results and Observations

For the information validation module's evaluation, four users that best exhibited different behaviors that users of the system may have were selected:

- **User 2:** represents users who send purely false information
- **User 7 and 21:** represent users who send purely relevant information
- **User 17:** represents users who send a mixture of false and relevant information

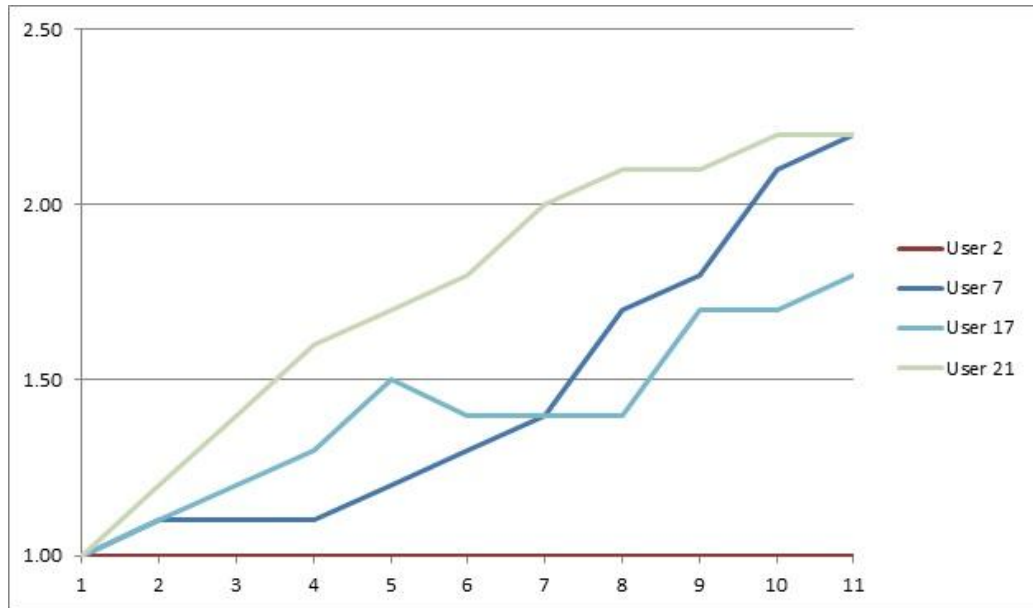


Figure 6.2: Illustration of User Reputation Model Simulation (with Selected Users)

User 2 represents users who only feed false reports into the system in order to game the system. As seen in Figure 2, the user's reputation did not go up and just stayed at 1 which is the lowest reputation score a user can get. The list of user 2's reports can be found on Table 6.18.

Table 6.21: User 2's Reports

Poster	Category	Date of Event	Location	ClusterScore	ExtractionScore
2	2	1/1/2012	Quezon	1	1
2	1	1/1/2012	Manila	10	1
2	2	2/1/2012	Makati	1	1
2	2	3/1/2012	Manila	12	1
2	2	3/1/2012	Quezon	1	1
2	2	3/1/2012	Manila	12	1
2	2	4/1/2012	Valenzuela	1	1
2	1	5/1/2012	Muntinlupa	1	1
2	1	5/1/2012	Makati	1	1

2	2	6/1/2012	Makati	1	1
2	1	7/1/2012	Muntinlupa	1	1
2	2	8/1/2012	Valenzuela	7	1
2	1	8/1/2012	Paranque	1	1
2	1	9/1/2012	Malabon	1	1
2	2	9/1/2012	Caloocan	7	1
2	2	10/1/2012	Makati	1	1

Users 7 and 21 represent users who only feed relevant and truthful reports into the system. As seen in Figure 2, these users' reputations only continue to increase until the last batch of reports has been sent. There has been no instance that their reputation scores went down. The list of user 7's and 21's reports can be found on Table 6.19 and 6.20, respectively.

Table 6.22: User 7's Reports

Poster	Category	Date of Event	Location	ClusterScore	ExtractionScore
7	2	1/1/2012	Cavite	8	1
7	2	4/1/2012	Makati	6	1.1
7	2	5/1/2012	Pateros	12	1.2
7	1	6/1/2012	Paranaque	9	1.3
7	1	7/1/2012	Pateros	8	1.4
7	1	7/1/2012	Pateros	8	1.4
7	1	7/1/2012	Quezon	10	1.4
7	2	8/1/2012	Pasay	11	1.7
7	1	9/1/2012	Quezon	12	1.8
7	1	9/1/2012	Quezon	12	1.8
7	1	9/1/2012	Quezon	12	1.8
7	1	10/1/2012	Pateros	5	2.1

Table 6.23: User 21's Reports

Poster	Category	Date of Event	Location	ClusterScore	ExtractionScore
21	2	1/1/2012	Cavite	8	1
21	2	1/1/2012	Cavite	8	1
21	1	2/1/2012	Marikina	9	1.2
21	1	2/1/2012	Paranaque	10	1.2
21	2	3/1/2012	Manila	12	1.4
21	2	3/1/2012	Mandaluyong	7	1.4
21	1	4/1/2012	Quezon	13	1.6
21	2	5/1/2012	Las Pinas	5	1.7
21	1	6/1/2012	Malabon	10	1.8
21	1	6/1/2012	Malabon	10	1.8
21	1	7/1/2012	Pateros	8	2

21	2	9/1/2012	Caloocan	7	2.1
----	---	----------	----------	---	-----

User 17 represents users who feed a mixture of false and truthful reports into the system. As seen in Figure 2, there are times when this user's reputation goes up and goes down. The list of user 17's reports can be found on Table 6.21.

Table 6.24: User 17's Reports

Poster	Category	Date of Event	Location	ClusterScore	ExtractionScore
17	2	1/1/2012	Cavite	8	1
17	1	2/1/2012	Marikina	9	1.1
17	2	3/1/2012	Mandaluyong	7	1.2
17	1	4/1/2012	Quezon	13	1.3
17	1	4/1/2012	Quezon	13	1.3
17	1	5/1/2012	Pateros	1	1.5
17	2	8/1/2012	Valenzuela	7	1.4
17	2	8/1/2012	Pasay	11	1.4
17	2	8/1/2012	Pasay	11	1.4
17	1	10/1/2012	Pateros	5	1.7

Based on the results of the simulation, it shows that the implemented reputation model is able to correctly assign reputation scores to the point-of-origin of a particular report.

Chapter 7: Conclusion and Recommendation

This chapter presents the final thoughts and learning points of the group upon finishing the study. It is divided into two sections. The first section discusses the overall conclusion of the project and outlines the goals and objectives that the research undertaking was able to meet. The second section, on the other hand, discusses some of SOMIDIA's possible areas for improvement that future researchers may take on.

7.1 Conclusion

Overall, all goals set for the research were met. SOMIDIA is able to automatically gather disaster reports from sources such as official news sites, social networking sites, particularly Facebook and Twitter, and SMS in near real-time. The system extracts relevant information from the gathered data by making use of bootstrapping techniques. The processed data are then validated and scores are given to the report's point of origin depending on the validity and relevance of the report. SOMIDIA's front-end system is able to present a visualization of disaster events by plotting them in an interactive map. Aside from these, the SOMIDIA website also provides features that will facilitate collaboration among users, may they be responders, volunteers, or normal citizens.

The success of this research provides an avenue for both the disaster managers and the people to communicate and collaborate for the safety and welfare for their fellow countrymen. The system's use of various social media platforms allows for the people's contribution to attain the goals of the system even if they are not aware of the system's existence, making for a greater coverage on the response of the disaster response teams.

More than just the contribution of the research on the field of disaster management, the research also contributes greatly to the academic community. The research shows a basis for an architecture of information extraction for the Filipino language. The research also shows a successful application of the techniques in Natural Language Processing on the unstructured text that is found in popular social media platforms, hopefully encouraging further works on the extensive information that can be retrieved on such platforms.

Though the system, at present, only focuses on disaster management, it opens possibilities in taking advantage of the massive information made available through social media. The adaptive nature of the system makes it easy for other developers to adopt its functions to create their own applications that make use of information gathered from social media. With this, services ranging from a more extensive and interactive traffic monitoring system to an accurate prediction of the next president of the Philippines have become possible. With future improvements to this research, there may come a time where developers become only limited by their imagination to provide innovative systems that make use of information provided by the ever growing popularity of social media.

However, numerous issues were encountered while developing the system. Collecting the data set which will be used in training the information extraction and machine learning modules of the system proved to be a problem since the system focuses on disaster events. Researchers had to wait for actual disasters in order to be able to get actual data which will be used for model learning and testing. Aside from this, there was also a lack of experts to annotate the data that the system has collected, thus forcing

manual annotation of data. The lack of stable preprocessing tools for the Filipino language also proved to be a hurdle in raising the accuracy of the information extraction and machine learning modules when dealing with text articles written using the said language. The absence of a complete Philippine NER also proved to be a problem.

The research undertaking was able to successfully meet the following objectives:

- Collect more data to create a larger dataset which will allow better model learning, testing and evaluation
- Review and analyze a number of existing information extraction and disaster management systems to come up with a disaster management system fit for the Philippine setting;
- Study and consider numerous information extraction techniques in gathering data for disaster management;
- Develop an adaptive information extraction system with bootstrapping capabilities as the system's main information extraction module;
- Review and evaluate existing tools and resources that could be integrated into the information extraction component of the system such as preprocessing tools provided by LingPipe and OpenNLP;
- Identify date, time, location and type of a given disaster event as relevant information for disaster management;
- Identify Twitter, Facebook, SMS, and official news sites as possible sources of data relevant to disaster management;
- Analyze and study different information validation techniques such as implementing reputation models, user behavior tracking, and crowdsourcing;
- Automatically gather disaster-related reports from the sources mentioned in number 5 in near real-time;
- Successfully extract relevant information collected by the system's crawler module may the report be written in English, Filipino, or Taglish;
- Validate reports by assigning a cluster score and an extraction score to each report; the former being based on the number of similar reports extracted within a given timeframe and the latter being based on the actual extraction process of the report;
- Implement a user reputation model to track, rank, and score user activity and points-of-origin of reports relating to the system;
- Successfully mark or plot disaster-related events on the interactive map in SOMIDIA's front-end system as soon as the back-end modules have successfully added the processed reports into the system's database;
- Present information most important to viewers of the interactive map in SOMIDIA's website by providing numerous filter and zoom to functions; and

- Facilitate external validation by allowing registered to vote up or vote down a particular disaster report.
- Implement SOMIDIA in such a way that it can be used to aid the mitigation, response, and recovery phases of disaster management

7.2 Recommendation

The SOMDIA system was able to meet all of its main objectives. However, throughout the system's entire development phase, a number of issues that were identified still need to be addressed. The following suggestions and areas of improvement for future research undertakings were noted:

- Inclusion of a lemmatizer sub module for the preprocessing step in order to lessen the need of adding numerous words in the list of trigger words and seed words used by the crawler and extractor modules, respectively
 - By making use of a lemmatizer, only the base form of a particular word that need to be recognized by the crawler and extractor modules need be added into the list
 - For example, the words “burn”, “burning”, “burned” will all be classified and tagged as “burn” only
 - This not only lessens the effort needed in completing the list of words, but also facilitate a cleaner and more structured tagging process because only a small number of tag words will be used
- Creation of stable preprocessing tools for Filipino and Taglish because results of testing show a significant increase in accuracy of extracting information from English texts that have previously gone through preprocessing
- Implementation of sentiment analysis on text articles in order to improve the accuracy of the crawler module
 - The crawler module yielded numerous false positive results due to non-disaster related reports containing disaster related trigger words (e.g. I love it when it rains heavily!, Is Taft already flooded?)
 - By determining the sentiments behind reports, the crawler will be able to more effectively determine which text articles are truly pertaining to disaster related events and are not just asking questions or stating sentiments that are non-disaster related
- Allow mapping of disaster names with disasters in real time
 - Reports coming social networking sites are often informally written, thus it can never be assumed that they will always include disaster words such as the word “typhoon”; Some people opt to include only the name of typhoons in their reports.
 - For example, during the height of typhoon Ondoy, text articles that come up, particularly tweets, that do not contain the word “typhoon” or “bagyo” and only the word “Ondoy” must still be tagged a disaster-related event categorized as typhoon.

- Leverage on geolocation features offered by social networking sites in order to determine the location of disaster related reports tagged with ambiguous location words such as “here”, “dito”, and “sa amin”.
- Use of TFIDF to generate a negative words list that would automatically eliminate a given text article as disaster-related
- Consider not only positive but also negative reports
 - Reports claiming that a particular disaster event did not or has not yet happened can also be valuable
 - Negative reports may be used in improving the current validation scheme
- Resolve problems in determining locations by populating the dictionary-based NER of the different permutations of locations names as the current dictionary-based NER only contains location names provided by the OSM database which only include formal location names (e.g. De La Salle University, DLSU, De La Salle, La Salle must be inputted into the NER and point to just one location)
- Creation of a knowledge-base for determining the exact location of a disaster event by mapping streets and municipalities to their corresponding cities, barangay districts to their corresponding municipalities, and so on
- Creation of a Philippine dedicated NER in order to allow more accurate extraction of disaster locations
- Improved coreference resolution for articles coming from social networks that makes use of special annotations (e.g. use of the at notation (@) and the hashtag(#) in tweets)
- Implementation of machine learning preprocessing – clustering in order to eliminate non-aligned data sets, thus speeding up rule generation and minimizing overly specific rules
- Implementation of machine learning classifiers such as SVM and make use of windowing and features of flanking tokens to identify whether the central token is a disaster or a location
- Expert annotation of disaster-related text articles that were used in training and testing system’s information extraction and machine learning modules
- Further testing on the effectiveness of the reputation model that was developed by determining whether the existence of a reputation model as one of the system’s validation scheme components proves to be better as compared to not having one at all
- Expert evaluation of the SOMIDIA’s front-end system
- Create an API of the information extraction engine that will allow it to be easily used for any other domain

- Partnership with an organization concerned with disaster management, preferably the Philippines NDRMCC, in order to improve and evaluate the current functionalities offered by the front-end system
- Creation of a social networking aspect for the system in that will foster a community dedicated to disaster management efforts
- Evaluation of the front-end system using usability testing
- Marketing of the system to spread awareness about its existence resulting to increased user participation
- Creation of a function that will make use of previously gathered reports in predicting details with regards to future disasters (e.g. extent of disaster based on number of reports gathered)

Bibliography

- Alias-i. (2003). Named Entity Tutorial. Retrieved October 12, 2011 from <http://alias-i.com/lingpipe/demps/tutorials/ne/read-me.html>
- Appelt, D., Hobbs, J., Bear, J., Israel, D., Kamayama, M., & Tyson, M. (2004, February 24). The sri muc-5 jv-fastus information extraction system. Available from <http://www.isi.edu/hobbs/muc5-fastus.pdf>
- Atkinson, M., Goot, E.V.D, Piskorski, J., & Tanev, H. (2008) Cluster-centric approach to news event extraction. In *Proceeding of the 2008 conference on New Trends in Multimedia and Network Information Systems*.
- Biagioli, C., Francesconi, E., Passerini, A., Montemagni, S., & Soria, C. (2005). Automatic semantics extraction in law documents. In *Proceedings of the 10th international conference on artificial intelligence and law* (pp. 133–140). New York, NY, USA: ACM. Available from <http://doi.acm.org/10.1145/1165485.1165506>
- Blow, C. & Jhalla, K. (2009) .Explaining SwiftRiver [Video file]. Available from <http://blog.ushahidi.com/index.php/2009/04/09/explaining-swift-river/>
- Brants, T. (2000). TnT - A statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference ANLP-2000*, Seattle, WA.
- Careem, M., De Silva, C., De Silva, R., Raschid, L., & Weerawarana, S. (2006, dec.). Sahana: Overview of a disaster management system. In *Information and automation, 2006. icia 2006. international conference on* (p. 361 -366).
- Citizens' Disaster Response Center. (2007). *2007 changing times, changing climate: Disasters in the Philippines 2007*. Available from <http://www.cdrc-phil.com/wp-content/uploads/2009/08/Disaster-Statistical-Report-2007.pdf>
- Citizens' Disaster Response Center. (2008). *2008 Philippine disaster report: Disaster statistical report '08*. Available from <http://www.cdrc-phil.com/wp-content/uploads/2009/08/Disaster-Statistical-Report-2008.pdf>
- Citizens' Disaster Response Center. (2009). *Philippine disaster report: Disaster statistics 2009*. Available from <http://www.cdrc-phil.com/wp-content/uploads/2009/08/2009-Philippine-Disaster-Report.pdf>
- Citizens' Disaster Response Center. (2010). *Philippine disaster report: Disaster statistics 2010*. Available from <http://www.cdrc-phil.com/wp-content/uploads/2009/08/PDR-2010.pdf>
- Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Aswani, N., Roberts, I., et al. (2010). Developing language processing components with GATE version 5 (a user guide). Retrieved July 22, 2010 from <http://gate.ac.uk/sale/tao/tao.pdf>
- Cunningham, H., Maynard, D., Bontcheva, K., & Tablan, V. (2002). Gate: A framework and graphical development environment for robust nlp tools and applications. In *Proceedings of the 40th anniversary meeting of the Association for Computational Linguistics*.

- Ekeklint, S. (2001). Semantic tagging. Available from <http://www.msi.vxu.se/sek/articles/semtag.pdf>.
- Etzioni, O., Banko, M., Soderland, S., & Weld, D. S. (2008, December). Open information extraction from the web. *Commun. ACM*, 51, 68–74. Available from <http://doi.acm.org/10.1145/1409360.1409378>
- Fahland, D., Gläßer, T. M., Leser, U., Quilitz, B., & Weibleder, S. (2007). HUODINI: Flexible information for disaster management. In *the proceedings of the 4th international iscram conference*.
- Finn, A., Computer Science & Informatics, U. C. D. S. of, Engineering, M. University College Dublin. College of, & Sciences, P. (2005). *A multi-level boundary classification approach to information extraction*. University College Dublin. Available from <http://books.google.com/books?id=qsHoSAAACAAJ>
- Ganjisaffar, Y. (2010). crawler4J – open source web crawler for java. *Crawler4J*. Retrieved October 16, 2011 from <http://code.google.com/p/crawler4j/>.
- Gosier, J. (2010, January 27). Separating the wheat from the califf. *iRevolution: From innovation to revolution*. Available from <http://irevolution.net/2009/08/08/proposing-crisis-mapping/>
- Grishman, R. & Yangerbar, R. (1998) Description of the PROTEUS/PET system as used for MUC-7. In *Proceedings of the 7th Message Understanding Conference (MUC-5)*.
- Grishman, R., Yangerbar, R., Tarpanainen, P., & Hutten, S. (2000). Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*.
- Hobbs, J. R. (1993). The generic information extraction system. In *MUC5 '93: Proceedings of the 5th Conference on Message Understanding* (pp. 87–91). Morristown, NJ, USA: Association for Computational Linguistics.
- IFRC. (n.d.). *About disaster management*. Available from <http://www.ifrc.org/en/what-we-do/disaster-management/about-disaster-management/>
- Jayram, T. S., Krishnamurthy, R., Raghavan, S., Vaithyanathan, S., & Zhu, H. (2006). Avatar information extraction system. *IEEE Data Engineering Bulletin*, 29, 2006.
- Jones, R., McCallum, A., Nigam, K., & Riloff, E. (1999). Bootstrapping for Text Learning Tasks. In *Proceedings of IJCAI-99 Workshop on Text Mining*, Stockholm, Sweden.
- Karamath, E., & Akyokus, S. (n.d.). Resume information extraction with named entity clustering based on relationships.
- Katz, B., Kaplansky, I., Schulman, R., & Ibrahim, A. (n.d.). Information Retrieval Using Patterns and Relations. Cambridge, Massachusetts, United States.
- Kit, C., & Webster, J. (1992). Tokenization as the initial phase in NLP. In *Proceedings of the 14th conference on Computational Linguistics*. Nantes, France.

- Kohlschütter C., Fankhauser P., & Nejd, W. (2010). Boilerplate Detection using Shallow Text Features. In *Proceedings of WSDM 2010 -- The Third ACM International Conference on Web Search and Data Mining*. New York City, NY USA.
- Kosseim, L. & Thiery, P. (2001) Proper Name Extraction from Non-Journalistic Texts. *Computational Linguistics in the Netherlands 2000*.
- Lakin, P., Thakker, D., & Osman, T. (2009, February 27). GATE JAPE grammar tutorial. Retrieved from <http://gate.ac.uk/sale/thakker-jape-tutorial/GATE%20JAPE%20manual.pdf>
- Lang, H. (2001, February 22). String Matching: Boyer Moore Algorithm. Retrieved October 12, 2011, from Fh-Flensburg: <http://www.inf.fh-flensburg.de/lang/algorithmen/pattern/bmen.htm>
- T. Lee, N.A. Bretana, and C. Lu. (2011, 12). "PlantPhos: using maximal dependence decomposition to identify plant phosphorylation sites with substrate site specificity." BMC Bioinformatics. Available: <http://www.biomedcentral.com/1471-2105/12/261#refs> [June 14, 2012].
- Lickfett, J., Ashish, N., Mehrotra, S., Venkatasubramanian, N., & Green, J. (2008). *The RESCUE disaster portal for disasters and emergency response*. In *the proceedings of the 5th international iscram conference*.
- Lim, B., Miranda, A., Trogo, J., & Yap, F. (2010). *Information extraction for eLegislation*. Undergraduate thesis, De La Salle University
- Makhoul, J. F. (1999, February). Performance measures for information extraction. In *Proceedings of darpa broadcast news workshop*.
- Masing, L. (1994). *Activities of volunteers in disaster management in the Philippines and their problems*. Available from <http://desastres.usac.edu.gt/documentos/pdf/eng/doc5796/doc5796-4b.pdf>
- Màrquez, L., Padró, L., Rodríguez, H. (2000). A machine learning approach to POS tagging, Machine Learning,
- McCallum, A. (2005, November). Information extraction: Distilling structured data from unstructured text. *Queue*, 3, 48–57. Available from <http://doi.acm.org/10.1145/1105664.1105679>
- Meier, P. (2009a, December 4). What is the Ushahidi platform? [Video file]. Available from <http://blog.ushahidi.com/index.php/2009/12/04/what-is-ushahidi/>
- Meier, P. (2009b, August 8). Proposing the field of crisis mapping. *iRevolution: From innovation to revolution*. Available from <http://irevolution.net/2009/08/08/proposing-crisis-mapping/>

- Meier, P. (2009c, April 10). Developing Swift River to Validate Crowdsourcing. *iRevolution: From innovation to revolution*. Available from <http://irevolution.net/2009/04/10/developing-swift-river-to-validate-crowdsourcing>.
- Meier, P. (2011, January 20). What is crisis mapping? An update on the field and looking ahead. *iRevolution: From innovation to revolution*. Available from <http://irevolution.net/2011/01/20/what-is-crisis-mapping/>
- Mitkov, R. (1998). Robust pronoun resolution with limited knowledge. In *ACL-36: Proceedings of the 36th annual meeting of the association for computational linguistics and 17th international conference on computational linguistics* (pp. 869–875). Morristown, NJ, USA: Association for Computational Linguistics. Available from <http://dx.doi.org/10.3115/980691.980712>
- Mohan, K. (2011, March 19). Facebook beats TV, newspapers in breaking news. *Financial Chronicle*. Approved from <http://www.mydigitalfc.com/news/facebook-beats-tv-newspapers-breaking-news-883>
- Pacific Disaster Center. (2005). Metropolitan Manila, the Philippines: Disaster risk management profile [Pdf file]. Available from <http://emi.pdc.org/cities/CP-Metro-Manila-08-05.pdf>
- Pettibone, J. (2002). Penn Treebank II Tags. Retrieved on October 14, 2011 from <http://bulba.sdsu.edu/jeanette/thesis/PennTags.html>
- Princeton. (2011, June 21). WordNet: A Lexical Database for English. Retrieved October 12, 2011, from Princeton: <http://wordnet.princeton.edu/>
- Rao, R. R., Eisenberg, J., & Schmitt, T. (2007). *Improving disaster management: The role of IT in mitigation, preparedness, response, and recovery*. Washington, D.C. : National Academies Press.
- RESTFB. (n.d.). <http://www.restfb.com>
- See, D. (2010, August 6). RP world's most disaster-prone – study. *Manila Bulletin*. Approved from <http://www.mb.com.ph/articles/271081/rp-world-s-most-disasterprone-study>
- Su, Y., Peng, J., & Jin, Z. (2009, 31 2009-april 2). Information quality assurance models for experts assessing in disaster management. In *Computer science and information engineering, 2009 wri world congress on* (Vol. 4, p. 33 -38).
- The Apache Software Foundation. (n.d.). Apache OpenNLP Developer Documentation. Retrieved October 12, 2011, from Apache: <http://incubator.apache.org/opennlp/documentation/manual/opennlp.html>
- Turmo, J., Ageno, A., & Catal`a, N. (2006, July). Adaptive information extraction. *ACM Computer Survey*, 38 . Available from <http://doi.acm.org/10.1145/1132956.1132957>

- United Nations Economic and Social Commission for Asia and the Pacific. (2010). *The Asia-Pacific disaster report 2010*. Available from <http://www.unescap.org/idd/pubs/Asia-Pacific-Disaster-Report%20-2010.pdf>
- Virtual University for Small States of the Commonwealth. (n.d.). Introduction to disaster management. Available from http://www.col.org/SiteCollectionDocuments/Disaster_Management_version_1.0.pdf
- Warfield, C. (2005). *The disaster management cycle*. Available from [http://www.gdrc.org/uem/disasters/1-dm cycle.html](http://www.gdrc.org/uem/disasters/1-dm%20cycle.html)
- Wong, T.-L., & Lam, W. (2007, February). Adapting web information extraction knowledge via mining site-invariant and site-dependent features. *ACM Trans. Internet Technol.*, 7. Available from <http://doi.acm.org/10.1145/1189740.1189746>

Appendix A. Penn Treebank POS Tagset

Table A-1 Word Level Tag Set (Pettibone, 2002)

CC	Coordinating conjunction e.g. and,but,or...
CD	Cardinal Number
DT	Determiner
EX	Existential there
FW	Foreign Word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List Item Marker
MD	Modal e.g. can, could, might, may...
NN	Noun, singular or mass
NNP	Proper Noun, singular
NNPS	Proper Noun, plural
NNS	Noun, plural
PDT	Predeterminer e.g. all, both ... when they precede an article
POS	Possessive Ending e.g. Nouns ending in 's
PRP	Personal Pronoun e.g. I, me, you, he...
PRP\$	Possessive Pronoun e.g. my, your, mine, yours...
RB	Adverb Most words that end in -ly as well as degree words like quite, too and very

RBR	Adverb, comparative Adverbs with the comparative ending -er, with a strictly comparative meaning.
RBS	Adverb, superlative
RP	Particle
SYM	Symbol Should be used for mathematical, scientific or technical symbols
TO	to
UH	Interjection e.g. uh, well, yes, my...
VB	Verb, base form subsumes imperatives, infinitives and subjunctives
VBD	Verb, past tense includes the conditional form of the verb to be
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner e.g. which, and that when it is used as a relative pronoun
WP	Wh-pronoun e.g. what, who, whom...
WP\$	Possessive wh-pronoun e.g.
WRB	Wh-adverb e.g. how, where why

Table A-2 Tag Set for Clauses (Pettibone, 2002)

S	simple declarative clause, i.e. one that is not introduced by a (possible empty) subordinating conjunction or a wh
SBAR	Clause introduced by a (possibly empty) subordinating conjunction.

SBARQ	Direct question introduced by a wh
SINV	Inverted declarative sentence, i.e. one in which the subject follows the tensed verb or modal.
SQ	Inverted yes/no question, or main clause of a wh

Table A-3 Phrase-level Tag Set (Pettibone, 2002)

ADJP	Adjective Phrase.
ADVP	Adverb Phrase.
CONJP	Conjunction Phrase.
FRAG	Fragment.
INTJ	Interjection. Corresponds approximately to the part
LST	List marker. Includes surrounding punctuation.
NAC	Not a Constituent; used to show the scope of certain prenominal modifiers within an NP.
NP	Noun Phrase.
NX	Used within certain complex NPs to mark the head of the NP. Corresponds very roughly to N
PP	Prepositional Phrase.
PRN	Parenthetical.
PRT	Particle. Category for words that should be tagged RP.
QP	Quantifier Phrase (i.e. complex measure/amount phrase); used within NP.
RRC	Reduced Relative Clause.
UCP	Unlike Coordinated Phrase.
VP	Verb Phrase.
WHADJP	Wh -adjective Phrase. Adjectival phrase

	containing a wh-adverb, as in how hot.
WHA VP	Wh -adverb Phrase. Introduces a clause with an NP gap. May be null (containing the 0 complementizer) or lexical, containing a wh-adverb such as how or why.
WHNP	Wh - Wh-noun Phrase. Introduces a clause with an NP gap. May be null (containing the 0 complementizer) or lexical, containing some wh-word, e.g. who, which book, whose daughter, none of which, or how many leopards.
WHPP	Wh -prepositional Phrase. Prepositional phrase containing a wh-noun phrase (such as of which or by whose authority) that either introduces a PP gap or is contained by a WHNP.
X	Unknown, uncertain, or unbracketable. X is often used for bracketing typos and in bracketing <i>the...the</i> -constructions

A-4 Punctuation Tags (Pettibone, 2002)

Punctuation Tags
#
\$
"
(
)
,
.
:
``

Appendix B. JAPE Rules for Filipino Extraction

```
Phase: FileExtractor-Sample
Input: Lookup
Options: control = appelt debug = true

Rule: DisasterDitoSaLocation
(
    ({Lookup.majorType == "disaster"}):ruleDisaster
    {Token.string == "dito sa"}
    ({Lookup.majorType == "location"}):ruleLocation
): ruleDL
-->
:ruleDL.DisasterExtraction = {
    disaster = :ruleDisaster.Lookup.majorType,
    location = :ruleLocation.Lookup.majorType
}

Rule: DisasterSaLocation
(
    ({Lookup.majorType == "disaster"}):ruleDisaster
    {Token.string == "sa"}
    ({Lookup.majorType == "location"}):ruleLocation
): ruleDL
-->
:ruleDL.DisasterExtraction = {
    disaster = :ruleDisaster.Lookup.majorType,
    location = :ruleLocation.Lookup.majorType
}

Rule: DitoSaLocation
(
    {Token.string == "dito sa"}
    ({Lookup.majorType == "location"}):ruleLocation
):ruleL
-->
:ruleL.Location{
    location = :ruleLocation.Lookup.majorType
}

Rule: SaLocation
(
    {Token.string == "sa"}
    ({Lookup.majorType == "location"}):ruleLocation
):ruleL
-->
:ruleL.Location{
    location = :ruleLocation.Lookup.majorType
}
```

```

Rule: LocationCity
(
    ({Lookup.majorType == "location"}):ruleLocation
    {Token.string == "city"}
):ruleL
-->
:ruleL.Location{
    location = :ruleLocation.Lookup.majorType
}

Rule: MataasAngTubigSaLocation
(
    {Token.string == "mataas ang tubig sa"}
    ({Lookup.majorType == "location "}):ruleLocation
):ruleMT
-->
:ruleMT.DisasterExtraction {
    disaster = "Flood"
    location = :ruleLocation.Lookup.majorType
}

Rule: LakasNgDisaster
(
    {Token.string == "lakas ng"}
    ({Lookup.majorType == "disaster"}):ruleDisaster
):ruleD
-->
:ruleD.DisasterExtraction {
    disaster = :ruleDisaster.Lookup.majorType
}

Rule: MulaSaOrgLoc
(
    {Token.string == "mula sa"}
    (
        {Lookup.majorType == "organization"}|
        {Lookup.majorType == "location"}
    ):ruleOrgLoc
):ruleOL
-->
:ruleOL.SourceExtraction{
    source = :ruleOrgLoc.Lookup.majorType
}

Rule: Time
(
    {Lookup.majorType == "TimeReference"}
):ruleTime
-->
:ruleTime.Time{
    time = :ruleTime.Lookup.majorType
}

Rule: Date

```

```

(
    ({Lookup.majorType == month}):ruleMonth
    {Token.string == "/" }
    ({Loopup.majorType == numeric}):ruleDay
    (
        {Token.string == "/" }
        ({Loopup.majorType == numeric}):ruleYear
    )?
):ruleDate
-->
:ruleDate.Date{
    Date = :ruleMonth.Lookup.majorType
    Day = :ruleDay.Loopup.majorType
    Month = :ruleMonth.Loopup.majorType
}

Rule: MayPaparatingNaBagyoSa
(
    {Token.string == " MayPaparatingNaBagyoSa "}
    ({Lookup.majorType == "location "}):ruleLocation
):rulePB
-->
:rulePB.DisasterExtraction {
    disaster = "Typhoon"
    location = :ruleLocation.Lookup.majorType
}

Rule: Sunog
(
    {Token.string == "sunog"}
):ruleS
-->
:ruleS.DisasterExtraction{
    Disaster = "sunog"
}

```

Appendix C. TFIDF Trigger Words

TFIDF - English Words					
aftershocks	warning	issued	kilometers	filipino	floods
alarm	warnings	lives	heavy	fire	flood
blaze	city	living	victims	firefighters	storm
broke	people	located	around	fires	town
buried	landslide	lying	over	geological	government
calamity	areas	magnitude	killed	happened	water
control	area	materials	damage	hurt	some
depth	houses	members	pagasa	investigation	village
earthquake	rains	miners	time	university	teams
earthquakes	rain	mining	daily	property	site
epicenter	landslides	mountain	scale	quake	sites
evacuated	barangay	mud	schools	razed	slopes
fault	families	ocd	seismology	reached	smoke
faults	residents	phivolcs	showed	relocation	soil
tremor	solidum	triggered	strength	tsunami	students
volcanology					

TFIDF - Filipino Words					
lindol	sugatan	nasunog	karamihan	gawin	dam
aabot	sumiklab	natabunan	kilometro	gaya	datos
aftershocks	tadios	nawawala	klase	gitna	department
agad	tahanan	ndrrmc	km	gobyerno	director
ahensya	talaga	ngayon	kumalat	guihulngan	disaster
alam	taon	ngayong	kuryente	gusali	dost
alarma	taong	opisyal	lahat	habagat	eastern

amihan	timog	oriental	lalo	habang	editorial
anyos	tinatayan g	pagasa	layong	halaga	nasugatan
apektado	trabaho	pagbaha	ligtas	hanggang	kapag
apoy	trahedya	pagguho	limang	hangin	sinasabing
arian	tulad	pagyanig	lindol	hanging	siya
awtoridad	tulong	pahayag	linggo	hilagang	siyang
babala	tuloy	pamahalaan	lokal	hulaan	solidum
bakit	tuluyang	pamamagitan	loob	iligan	subalit
balitang	tumama	panahon	lugar	ilocos	narekober
bandang	tuwing	pangasinan	lumikas	ilog	nasalanta
bangkay	umaabot	pangulong	lungsod	imbestigasyon	naitala
barangay	umabot	pantukan	lupa	inaasahang	naitalang
bata	umaga	partikular	maaaring	insidente	nakaraang
batanes	umano	patay	mabilis	kababayan	namang
bawat	umanong	pati	magiging	kabilang	namatay
bayan	bagyo	patuloy	magkaroon	kagabi	nangyari
bfp	sunog	phivolcs	magnitude	kahoy	naramdam an
biktima	katao	pinakahuling	makaraang	kailangan	nasaktan
bilang	kanilang	pinoy	malalakas	kalamidad	sasakyan
bilis	baha	pinsala	mamamayan	kalat	senado
blues	alas	pnp	mataas	kalsada	sentro
buhay	bahay	police	matinding	kamakalawa	nasawi
bumbero	ulan	posibleng	mayroon	kanluran	lalawigan
bundok	tubig	probinsya	milyong	kanyang	ilang
bunsod	kahapon	problema	mmda	epekto	sinabi

buwan	bagyong	protection	naabo	gabi	nasabing
central	araw	publiko	naapektuhan	gamit	lamang
council	luzon	sakit	naapula	ganap	bansa
daan	bahagi	sakuna	naganap	bago	pamilya
dagat	malakas	samantala	naging	upang	matapos
dakong	oras	sanhi	nagsimula	area	residente

Appendix D. Extractor Seed Words

Extractor Seed Words - Disaster Words				
flooding	malindol	rainstorm	nasusunog	binahang
tubig-baha	landslide	sleet	sinusunog	unyielding rain
nbaha	earthfall	spate	apoy	overflowed creek
flash flood	mudslide	spit	umaapoy	pag-ulan
raining	rockslide	torrent	inaapoy	wildfire
flood	moderate to heavy rainfall	volley	maapoy	rainy
flashfloods	umuuln	bagyo	storm	bad weather
floods	moderate-heavy rains	binabagyo	typhoon	heavy rain
flooded	pagbaha	bumabagyo	cyclone	bahang
fire	lakas ng ulan	ulan	downpour	raining hard
blaze	inclement weather	ulanan	gale	malakas ang ulan
charring	heavy rains	umuulan	gust	floodings
combustion	binaha	inuulan	hurricane	rain
conflagration	rains	inuulanan	monsoon	rainshowers
devouring	thunderstoms	maulan	precip	uulan
embers	downpour/flood	mabagyo	precipitation	bumaha
smoke	rain showers	baha	tempest	inulan
flames	thunderstorm	bumabaha	windstorm	heavy rain fall
flare	strong rain	mabaha	raining non stop	masamang panahon
heat	heavy tain	binabaha	seism	nilindol
incandescence	hard rain	earthquake	seismism	lumilindol
inferno	nagksunog	macroseism	temblor	cloudburst
phlogiston	bahaing	microseism	trembler	drencher
pyre	floodway	quake	undulation	drizzle

scorching	moderate to heavy rains	quaker	upheaval	flurry
sunog	umulan	seimicity	lindol	rainfall
ulaaaaan	maulang	non stop rain		

Appendix E. List of News Websites

List of News Websites	
http://newsinfo.inquirer.net/	http://www.malaya.com.ph/index.php/news/national
http://www.abs-cbnnews.com/nation	http://www.mb.com.ph/
http://www.abs-cbnnews.com/weather	http://www.newsbreak.ph
http://www.balitapinoy.net	http://www.sunstar.com.ph/manila/local-news
http://www.gmanetwork.com/news/	http://www.manilatimes.net
http://www.interaksyon.com	http://www.abante.com.ph
http://www.journal.com.ph	

Appendix F. List of Facebook Pages

Facebook Pages		
Abs-Cbn	Official Gazette of the Republic of the Philippines	CNN International
ABS-CBN Ad Congress 2011	PAGASA	Department of Health (Philippines)
abs-cbnNEWS.com	Patrol ng Pilipino	Dost_pagasa
ABSCBNdotcom	Patrol Ng PilipinoTV	DZMM TeleRadyo
Ako ang Simula TV	Philippine Daily Inquirer (Official)	Failon Ngayon
ANC 24/7	Philippine News	GMA News TV
ANC Talkback	TV 5	IBaBalita
Asia News Network	TV Patrol	Manila Bulletin Newspaper
Bandia	UBE Txt FIRE	MMDA
Bayan Mo, iPatrol Mo: Ako ang Simula	Umagang KayGanda (Official Page)	Mornings@ANC
Bida KaPamilya	Unang Hirit	Newsbreak
Breaking News	Weather Watch	ABC News
Ces Drilon	Philippine Disaster Recovery Foundation	TV5 Manila
Choose Philippines	Disaster of the Philippines	PH Red Cross
CCCP Movement	MDDA PH	Disaster News
Philippine News Net	SunStar Philippine News	Disaster News Today
Philippine News	Philippine News Daily	ABSCBN Network
Yahoo! Philippines	PH Typhoon News	Occupy Philippines

Appendix G. List of Trusted Twitter Accounts

Twitter Accounts of Trusted Users			
1PhilippineNews	govph	dolandcastro	nadiatrinidad
1rgcruz	GovPH_PCOO	DOTPhilippines	newsphilippines
AbnerMercado	gretchenfullido	DYANCASTILLEJO	Nikobaua
ABS-CBN News	gretchenmalalad	DZMMTeleRadyo	ninacorpuz
ABSCBN	HenryOmagaDiaz	ernie_manio	noynoyaquino
ABSCBNBreaking	iamalexsantos	espiecabral	PAGASAFFWS
abscbndotcom	iammarielozano	Failon_Ngayon	Patrol_Pilipino
abscbnnewstv	iampinkywebb	FrancisFaulve	Paulhenson
AKOANGSIMULAtv	InaReformina	gingerconejero	PhCHED
ANCALERTS	inquirerdotnet	gingreyes	Philippine_News
anjo_bagaoisan	jacquemanabat	glendamgloria	philippinenews
applesjalandoni	jasmin_romero	gmanews	philippinenews1
arleneburgos	jeffcanoy	philippinenews7	scoopbox
atomaraullo	jennyfreyes	philredcross	sheryllmundo
ayeemacaraig	JINGCASTANEDA	PHNews0verviews	shiereyes
Balitaph	JojoMalig	phnewshd	sol_aragones
BalitaPilipinas	Jorge_Carino	pia_gutierrez	stephgosk
bandila	juLiusbabao	piahontiveros	TempoPHL
bayanmo	Karen_DaviLa	pinoy_balita	thepocnews
Climate	KellyO	PTVph	tjmanotoc
Bernadette_ABS	kuyakim_atienza	radypatrol42	TXTFIRE
chiarazambrano	lundahumilla	RedCross	TVPatrol
cnnbrk	Magic899	rongagalac	willardcheng
daily	manila_bulletin	ryan_chua	zenhernandez
DavidChalian	marietonpacheco	S23Channel	ZyannAmbrosio
DepEd_PH	MBulletin3D	salamatdok	

DILG	MMDA	DOHgovph	MMDAtweets
------	------	----------	------------

Appendix H. POS tagger performance measure, large text full

Bangkok (The Nation/ANN) - Despite the weaker external environment, growth in Southeast Asia is expected to remain robust, supported by strong domestic demand and reconstruction in flood-affected areas, the Asian Development Outlook Supplement released yesterday said.

"Growth in Thailand is expected to be 5.5 per cent or higher this year, accelerating from almost zero per cent growth last year as a result of last year's devastating floods," Haruhiko Kuroda, president of the Asian Development Bank (ADB), told a press conference on the sidelines of yesterday's Thailand-ADB-IMF conference.

Disruption of supply chains caused by last year's devastating floods has been overcome as mentioned by Finance Minister Kittiratt Na-Ranong, said Kuroda, referring to Kittiratt's comments on strong recovery in automobile production and expected quick recovery of the electronics sector.

Private consumption and investment is very strong in Thailand, said Kuroda.

Asked about the outlook for gross domestic product growth in Myanmar, Kuroda said the country had been isolated from the rest of the world for the past three decades.

"Now Myanmar has integrated much with Asean and the global economy, and we expect substantial growth over the coming years," he said.

Myanmar, however, has not made sufficient investments in infrastructure projects, education and healthcare. Investment in these areas are much needed now, he said. And the capacity of the Myanmar government must be enhanced, he suggested.

ADB and World Bank will provide technical assistance and policy advice to Myanmar but financial support must be provided only after the country clears debts owed to the two institutions, he said.

According to ADB's report, vibrant domestic demand and private investment appeared to drive Southeast Asian growth in the first half of this year. The region's economies expanded 4.3 per cent in the first quarter after a weak 2.9-per-cent growth in the last quarter of 2011. It was mainly due to the strong rebound in Thailand and the robust growth in the Philippines.

The open economies of Singapore and Malaysia, however, posted slower growth in the first quarter as a result of the weaker external demand. Vietnam's first quarter growth was also much lower than expected, pulled down by a contraction in construction and flat industrial production, said the report.

Meanwhile, Kuroda said that China's economy would report 8.2-per-cent growth, though slowing down from April's forecast of 8.5 per cent. Next year, China will grow 8.7 per cent. China is Thailand's largest export market.

He expected the Chinese government to launch fiscal stimulus following monetary policy easing lately. South Korea has already implemented public spending to boost growth, he said.

India's growth forecast has moderated as high inflation and trade deficit make it difficult to ease monetary policy to stimulate demand, according to the report.

Developing Asia as a whole will grow 6.6 per cent in 2012 and 7.1 per cent in 2013, lower than the 6.9 per cent and 7.3 per cent forecast in ADB's Asian Development Outlook published in April, said Kuroda.

Kuroda said that he agreed with the consensus view that the euro zone would show a slight minus growth this year and slight positive growth next year. Economic situations could change when ADB issues its updated economic forecast in September, he added.

(Source: <http://ph.news.yahoo.com/asean-economies-poised-growth-despite-global-difficulties-055002917.html>)

Appendix I. Sample File of Facebook Posts

```
<Facebook-0327050813.txt size = 6>
  <facebook>
    <user> Maoche Gwapo </user>
    <time> Tue Mar 27 04:54:39 PHT 2012 </time>
    <post> help us hit 1K fllowr for @pooh_tiki and 100K
folwers for @pooh_tik and get the chance to b 1 of d 2 follwrs
who will win 2 4'dbest concert tckts each on MAY 4,5.11&12
at the MUSIC MUSEUM :)
lets hit follow and retweet :) on twitter :)
follow me @maogwapo08
null </post>
  </facebook>
  <facebook>
    <user> ABS-CBNnews.com </user>
    <time> Tue Mar 27 04:33:24 PHT 2012 </time>
    <post> Top story: Ground Zero Manila: Will you
survive a nuke blast?
MANILA, Philippines - Amid the tension over North
Korea's long-range rocket launch that has the Philippines in its
trajectory, ever wondered if you will survive a nuclear
strike?... </post>
  </facebook>
  <facebook>
    <user> ABS-CBNnews.com </user>
    <time> Tue Mar 27 04:33:21 PHT 2012 </time>
    <post> [In Depth] Ground Zero Manila: Will you
survive a nuke blast?
MANILA, Philippines - Amid the tension over North
Korea's long-range rocket launch that has the Philippines in its
trajectory, ever wondered if you will survive a nuclear
strike?... </post>
  </facebook>
  <facebook>
    <user> ABS-CBNnews.com </user>
    <time> Tue Mar 27 04:33:10 PHT 2012 </time>
    <post> Top story: Corona prepared to lose everything
for principles
MANILA, Philippines -Ê Chief Justice Renato Corona on
Monday said he is prepared to "lose everything" in his fight to
uphold his principles. </post>
  </facebook>
  <facebook>
    <user> Jonathan Renier Verzosa </user>
    <time> Tue Mar 27 03:44:26 PHT 2012 </time>
    <post> Sign up here!
http://www.jrbvrnl.ishareinternational.com/
null </post>
  </facebook>
  <facebook>
    <user> ABS-CBNnews.com </user>
    <time> Tue Mar 27 03:34:57 PHT 2012 </time>
```


<post> [In Depth] Corona prepared to lose everything
for principles
MANILA, Philippines -Ê Chief Justice Renato Corona on
Monday said he is prepared to "lose everything" in his fight to
uphold his principles. </post>
</facebook></Facebook-0327050813.txt>

Appendix J. Sample News Article

<News-0713162223.txt>

<news>

<title> Magnitude 4.9 quake hits Biliran, parts of Visayas | Inquirer
News </title>

<body> Home > Newsinfo > Cebu Daily News > CDN - Visayas >
Magnitude 4.9 quake hits Biliran, parts of Visayas

Magnitude 4.9 quake hits Biliran, parts of Visayas

7:18 am | Friday, July 6th, 2012

Tweet

TACLOBAN CITY, Leyte — Several residents of the capital town of Naval in Biliran ran to the streets on Wednesday night after a 4.9 magnitude earthquake hit the municipality and some parts of Leyte.

Flor Jackson of the Philippine Information Agency (PIA) based in Naval told the Inquirer that several residents scampered out of their houses.

“They fear being hit by falling cabinets and other decor inside their houses,” she told the Inquirer in a text message.

The Office of Civil Defense in Eastern Visayas has not received a report of any damage caused by the earthquake, said Cherelyn Lubang, OCD-8 administrative officer.

The earthquake, which was tectonic in origin, was recorded at 7:09 p.m., said Myra Dolina of the Philippine Institute of Volcanology and Seismology in Tacloban. Its epicenter was traced 3 km south of Naval, she added.

Dolina said the earthquake was recorded at Intensity 4 in Naval, Kawayan, Culaba and Biliran, all in Biliran; and in towns of Calubian and Leyte, both in Leyte.

Intensity 3 earthquake was felt in Kananga town in Leyte; Intensity 2 in Palompon and Matag-ob towns as well as Tacloban, also in Leyte.

The earthquake was recorded at Intensity 1 in Cebu City. /INQUIRER

Join us on

</body>

<time> </time>

<language> </language>

<disaster> </disaster>

<url> <http://newsinfo.inquirer.net/224379/magnitude-4-9-quake-hits-biliran-parts-of-visayas?ModPagespeed=off> </url>

</news>

</News-0713162223.txt>

Appendix K. Sample List of Twitter Posts

```
<Twitter-0705192437.txt size = 3>  <twitter>
  <id> 219991576947068928 </id>
  <user-account>
    <user> basha_marie </user>
    <followers> null </followers>
    <following> null </following>
    <status-count> null </status-count>
    <account-age> null </account-age>
  <location> null </location>
</user-account>
  <time> Tue Jul 03 11:10:53 CST 2012 </time>
  <tweet> "@ANCALERTS: Caloocan Mayor Recom Echiverri
suspends classes in ALL levels in Caloocan City -- Great
call.Traffic & flood almost everywhere. </tweet>
  <language> null </language>
  <disaster> null </disaster>
  <type> null </type>
  <disaster location> null </disaster location>
</twitter>  <twitter>
  <id> 219977652851843074 </id>
  <user-account>
    <user> reineerreyes </user>
    <followers> null </followers>
    <following> null </following>
    <status-count> null </status-count>
    <account-age> null </account-age>
  <location> null </location>
</user-account>
  <time> Tue Jul 03 10:15:34 CST 2012 </time>
  <tweet> flood's worse in caloocan due to recom's
ongoing drainage 'improvement' project #AngGalingMoMayor #dafuq
</tweet>
  <language> null </language>
  <disaster> null </disaster>
  <type> null </type>
  <disaster location> null </disaster location>
</twitter>  <twitter>
  <id> 219961964510715904 </id>
  <user-account>
    <user> icecroft </user>
    <followers> null </followers>
    <following> null </following>
    <status-count> null </status-count>
    <account-age> null </account-age>
  <location> null </location>
</user-account>
  <time> Tue Jul 03 09:13:13 CST 2012 </time>
  <tweet> @mmda hello po, would you know if there's
flood in r. papa, caloocan? thanks! </tweet>
  <language> null </language>
  <disaster> null </disaster>
```

```
<type> null </type>  
<disaster location> null </disaster location>  
</twitter></Twitter-0705192437.txt>
```

Appendix L. Sample List of Annotated Twitter Posts

```

<Annotated-0707153451.txt size = 3>
  <twitter>
    <id> 219991576947068928 </id>
    <user-account>
      <user> basha_marie </user>
      <followers> 0 </followers>
      <following> 0 </following>
      <status-count> 0 </status-count>
      <account-age> null </account-age>
    <location> null </location>
    </user-account>
    <time> Tue Jul 03 11:10:53 CST 2012 </time>
    <tweet> "@ANCALERTS: Caloocan Mayor Recom Echiverri
suspends classes in ALL levels in Caloocan City -- Great
call.Traffic & flood almost everywhere. </tweet>
    <language> English </language>
    <disaster> Yes </disaster>
    <type> flood </type>
    <disaster location> Caloocan City </disaster
location>
  </twitter>  <twitter>
    <id> 219977652851843074 </id>
    <user-account>
      <user> reineerreyes </user>
      <followers> 0 </followers>
      <following> 0 </following>
      <status-count> 0 </status-count>
      <account-age> null </account-age>
    <location> null </location>
    </user-account>
    <time> Tue Jul 03 10:15:34 CST 2012 </time>
    <tweet> flood's worse in caloocan due to recom's
ongoing drainage 'improvement' project #AngGalingMoMayor #dafuq
</tweet>
    <language> English </language>
    <disaster> Yes </disaster>
    <type> flood </type>
    <disaster location> caloocan </disaster location>
  </twitter>  <twitter>
    <id> 219961964510715904 </id>
    <user-account>
      <user> icecroft </user>
      <followers> 0 </followers>
      <following> 0 </following>
      <status-count> 0 </status-count>
      <account-age> null </account-age>
    <location> null </location>
    </user-account>
    <time> Tue Jul 03 09:13:13 CST 2012 </time>

```

```

        <tweet> @mmda hello po, would you know if there's
flood in r. papa, caloocan? thanks! </tweet>
        <language> </language>
        <disaster> No </disaster>
        <type> </type>
        <disaster location> </disaster location>
</twitter></Annotated-0707153451.txt>

```

Appendix M. Resource Persons

Mr. Ralph Vincent J. Regalado

Thesis Adviser, Faculty Member

College of Computer Studies

De La Salle University

regalador@dlsu.ph

Appendix N. Personal Vitae

Mr. Herman C. Cheng Jr.

1527 V. Mata St., Nagtahan, Manila

(0917)8519554

herman_chengjr@dlsu.ph

Ms. Jenina L. Chua

739 – I A. Bonifacio St., Balintawak, Quezon City

(0927)3200486

jenina_chua@dlsu.ph

Mr. Justin L. Co

1266 C.M. Recto, Binondo, Manila

(0922)8555861

justin_co@dlsu.ph

Mr. Angelo Bruce L. Magpantay

5445EB Currie St., Barangay Palanan, Makati City

(0917)5835417

angelo_magpantay@dlsu.ph