Universitetet i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study program/specialization: Computer Science	Spring semester, 2016 Open / Confidential
Author: Dinesh Shivakoti	Dinesh Shivakoti (signature author)

Instructor:
Erlend Tøssebro
Supervisor(s):
Erlend Tøssebro

Title of Master's Thesis:

Automatic Detection and Extraction of Event Locations in News Report to locate in Map

ECTS:
30

Subject headings: Information Extraaction, GIS, Stanford CoreNLP, POS, Google Maps API	Pages: …55…… + Attachments/other: Source Code. Stavanger,…15/06/2016 Date/year

Automatic Detection and Extraction of Event Locations in News Report to locate in Map

Dinesh Shivakoti

Faculty of Science and Technology

University of Stavanger

June 2016

Abstract

In the present word of electronic media, there are hundreds of newspapers available online, it is really a tired some task to search for a specific event in this online newspapers.

This paper looked into possibility of automatic detection of accident events in news report and pointing out the event locations in the map as precisely as possible. Although many research has been done in data mining and extracting relevant and intended meaning of text data, many a times it is not sufficient alone. Various factors affect the quality of the extracted information, which includes correctness, relevance of the information. Also geo-locating a place based on the description present in the text data is still a challenge task. To know human intentions based on text data is an Artificial Intelligence problem. Much works have been done towards this field. Location ambiguity, lack of geographic information on web pages, language-based and country-dependent addressing styles, and multiple locations related to a single web resource are notable difficulties. Named Entity recognizer is always not working accurately in all languages as in case of Norwegian locations. Also pin pointing the location is a trivial because of the ambiguity in the names of places. Thus this paper studies these hurdles and application is designed to extract and filter relevant data from text document to point out the event in the map precisely.

Acknowledgements

I would like to thank Erlend Tøssebro, my supervisors for his valuable advises and contributions. And deepest gratitude goes to for his relentless support and insightful comments.

Preface

This thesis is submitted in partial fulfilment of the requirements to complete the Masterof Science (M.Sc.) degree at the Department of Electrical and Computer Engineering at the University of Stavanger (UiS), Stavanger Norway.

The work has been done under the supervision Erlend Tøssebro. It contains work done from February to June 2016.

The study is on event information extraction from news report using Stanford CoreNLP and geocoding. An attempt is made to extract the event locations and geocode those locations as preciously as possible.

This thesis might be helpful for those who think to work on information extraction and geocoding. It might pave a good ground for those who think of working with data mining and information extraction using Natural Language processing toolkit and geocode the extracted information based on the location.

Contents

Chapter 1 Introduction **1**
1.1 Background 1
1.2 Problem Statement 2
1.3 Problem Significance 2
1.4 Thesis Organization 3
Chapter 2 Related Work **4**
2.1 Geographic Information System 4
2.2 Information Extraction 6
2.2.1 Named entity extraction 6
2.2.2 Relation extraction 7
2.2.3 Event extraction 7
2.3 Natural Language Processor 7
Chapter 3 Choice of Natural Language Processor and Geo-Coding service **9**
3.1 Natural Language Processing Toolkits Choice 9
3.2 Geo-Coding Service Choice 12
Chapter 4 Design and Methodology **15**
4.1 Information Extraction 15
4.1.1 Detect the Occurrence of Accident Event 15
4.1.2 Extraction of Event and Location from Accident Report 16
4.1.3 Experiment 21
4.2 Geocoding Location Named Entity 22
4.2.1 Disambiguation of geocoding 23
4.2.2 MySQL Geographical Database 23
4.2.3 Geocoding case study 25
Chapter 5 Results and Analysis **30**
5.1 Experimental Set up 30
5.2 Result: Geocoding cases. 30
5.3 Result Comparison 36
Chapter 6 Conclusions and Discussion **39**
6.1 Future work: 40
Appendix A **43**

A.1 Haversine Formula 43
A.2 ‘near’ case pseudo code 43
A.3 ‘between/intersection’ case 43
A.4 Shortest Distance between Latitude and Longitude Pairs 44
A.5 Intersection between Coordinate Pairs of streets (lineStrings) 44
A.6 Google Geocoding 44
A.4 Stanford POS tags description 45

List of Figures

Figure 2.1: GIS collective. Re-printed from [10] 5
Figure 3.1: POS output from Stanford CoreNLP (screen shot of Stanford Online POS Tagger) 10
Figure 3.2: NER output from Stanford CoreNLP (screen shot of Stanford Online NER) 11
Figure 3.3: Dependency tree output from Stanford CoreNLP (screen shot of Stanford Online CoreNLP) 11
Figure 4.1: Information Extraction and Geocoding Process. 15
Figure 4.2: process to differentiate Accident Report and non-Accident Report 16
Figure 4.3: Design of Information Extraction process 17
Figure 4.4: Stanford NER some time fails to tag locations and street names. 17
Figure 4.5: Street Name Extraction Filter 18
Figure 4.6: Spatial relation Location extraction filter 19
Figure 4.7: Tunnel Name Extraction Filter 19
Figure 4.8: Road Name Extraction Filter 20
Figure 4.9: Actual accident location. Manually pointed by AIBN 20
Figure 4.10: Spatial Retation Extraction Filter 21
Figure 4.11: Information Extraction Process from News report 21
Figure 4.12: Geojson file example 24
Figure 4.13 : csv file example 24
Figure 4.14: creating database and importing cvs file to database. 25
Figure 4.15: Geocoding and Reverse-geocoding process 26
Figure 4.16: Geocoding and Reverse-geocoding process for between/intersection case 28
Figure 4.17: Geocoding and Reverse-geocoding process for ‘inside tunnel’ case 29
Figure 5.1: near case news report result for event on E6 road near Svinesundparken 31
Figure 5.2: near case news report result for event on E18 road near Fiskevollbukta 31
Figure 5.3: between case news report result for event between Cort Adelers gata and 33
Figure 5.4: between case news report result for event the intersection of Trondheimsveien .. 33
Figure 5.5: certain distance inside a tunnel case news report result. 34
Figure 5.6: certain distance inside a tunnel case news report result for event on Oslofjord 35
Figure 5.7: Event Location result comparison. (A) AIBN manually located near result, (B) automatically 36
Figure 5.8: Event Location result comparison. (A) AIBN manually located between result, (B) automatically located between result 37
Figure 5.9: AIBN manually located accident location 37
Figure 5.10: Automatically located result. 38

List of Tables

Table 5:1: difference in distance between automatic located and AIBN located events for near case. 32
Table 5:2: difference in distance between automatic located and AIBN located events for intersection case 34
Table 5:3: difference in distance between automatic located and AIBN located events for insidetunnel case 35

List of Abbreviations

AIBN	Accident Investigation Board of Norway
ERSI	Environmental Systems Research Institute
NLP	Natural language Processor
GIR	Geography Information Retrieval
GIS	Geography Information System
NER	Named Entity Recognizer
OSM	Open Street Map
POS	Parts of Speech
AI	Artificial Intelligence
HMM	Hidden Markov Model
GATE	General Architecture for Text Engineering
NER	Named Entity Recognizer
NE	Named Entity

Chapter 1 Introduction

1.1 Background

Global status report on road safety 2015 conducted by World Health Organization shows that road accident is the 9th leading cause of death globally across all age groups and by 2030 it is predicated to become the 7th leading cause of death [1]. It was found that more than 1.2 million people die each year on the world's roads with low and middle income countries having the highest number of death counts. The main cause of the accidents is the increase in the motor vehicles. Because of road traffic crashes, these countries lose about 3% of GDP annually. But as a matter of fact, most traffic crashes can both be predicted and can be prevented. The report has also pointed out that there are substantial evidences of road system improvement that are effective at making safer roads. Countries like Norway that have successfully implemented these road system improvements have seen corresponding reductions in road traffic deaths. The report by Accident Investigation Board of Norway (AIBN) shows a decrease in the number of accident in the recent years [2].

With the advancement in the Information Technology, the availability of news in the electronic format has certainly increased. This gave a call for automatic extraction of information from the news. Various research papers focus on extracting news from online newspaper, extracting news headlines. The following papers studied automatic information extraction from various news papers with diverse language had been studied extensively in recent years [3] [4] [5] [6]. A particular problem is studied in the thesis: automatically detecting accident occurrence in English News Report and extracting accident place to pin point the location in the map. This thesis only takes into account the accident events within Norway region.

This thesis can be broadly sub categorised into three main sub tasks. First task involves monitoring the news streams constantly in order to identify the article source that reports accidents. This is a classification problem which involves classifying each article source as reporting an accident or not reporting an accident. This task is carried out manually as there are limited numbers of source that reports news in English language. Second task is identifying accident event in the article and extracting this accident event from each news report with its location. In this sub task, spatial relation or extract distance approximation words are extracted as "Near", "On", "Between" and "Intersection". These words help in pointing the event location in the map. For example, "On 5 May 2014 the trailer of a Swedish-registered heavy goods vehicle loaded with timber overturned when it was entering a four-lane motorway on the E6 road near Svinesundparken in Østfold County". Here the task is to extract accident keywords as "overturned", and locations "E6" and "Svinesundparken" with

the spatial relating word as “Near”. The final task is mapping the geographic locations accurately in map. As discussed later, all these sub tasks are different from each other and have their own challenges. The first sub task is more of a classification problem with requires refining sources and monitoring constantly the source of accident news. Although tweeter tweets are the best and fastest means for getting news in real time but they are not the best choice for the purpose of this thesis. Tweeter tweets has very short description for the event location. Thus they are not sufficient to get a precise location. So for this thesis we choose to use the Accident Investigation Board of Norway **(**AIBN) accident reports as the main source of accident report source. The AIBN accident news reports describe the event location along with the spatial relationship to the location. The second sub task is an information extraction problem and for this task Stanford Natural Language Processor (CoreNLP) [7] is used along with some filters based on sequential rules for extraction. A greedy algorithm is designed for to map this extracted location on a map with accuracy. Finally, these tasks are combined to solve the problem of this thesis.

1.2 Problem Statement

Given a collection of news report text documents, our task is to find the existence of accident event with its location in the news report and extract them. Extracting meaningful information from the text document is a challenging process. Extracting accident events means extracting the accident keywords. Keywords are the sensitive verb words as “collision”, “overturned”, “skid”, “drove off” , “caught fire” etc from the news report which indicates that the accident events has occurred. In the later part of the thesis we will discuss the methods to extract these keywords. Unlike general web search, location-based search is a nontrivial task as it involves ambiguity in the location. If a newspaper article contains the words Denzel Washington, a good Named Entity Recognizer (NER) must label “Denzel Washington” as a person and “Washington” as a location. NER is discussed in the following chapters. Much research has been done towards this field to bring out the ambiguity in place names. It is difficult to bring the intention of the speaker in the text. Norwegian local places are hard to extract using NER as many of the places do not come under the circumference of NER. Location words are location named entities and there are many software packages that provides the facilities to extract location entities from the text.

1.3 Problem Significance

Mapping accident locations in map can be useful for many applications. As for instance, it can helps drivers to see the accident prone areas and act accordingly. It also enables roadways accident department to take preventive actions to alert drivers driving in the accident prone areas.

Because of the advancement in the informational technology, accident news reports are usually from electronic news articles. Although it is possible to collect such reports manually by reading all the accident related news from a particular region, but it is a difficult, time consuming and labour intensive task. It is thus useful to develop automated techniques to extract such reports automatically to find details about the cause, consequence of the accident .Thus identifying accident hot spots will definitely help in dealing with the accident.

1.4 Thesis Organization

The thesis is organized in the following way

Chapter 2 covers basic background theory needed to understand this thesis.

Chapter 3 discuss the choice for Information extraction toolkit and geocoding service.

Chapter 4 has a detailed discussion on the design and approaches needed to build frameworks to work on the thesis.

Chapter 5 discusses about test results, it also discuss about performance analysis with results comparison with manually pointing the location.

Chapter 6 concludes the thesis, discuss the hurdles faced, also indicating some future enhancements and suggestions for further work.

Chapter 2 Related Work

In this chapter, we review related work in Geographic Information System, information extraction, dependency tree that has been studied in data mining, natural language processing, and information retrieval, respectively.

2.1 Geographic Information System

A geographic information system (GIS) is a computer system to capture, store, check, and display data related to positions on Earth's surface i.e. GIS helps to visualize geographic information on a map. GIS can show many different kinds of data in one map [8]. This enables people to more easily sees, analyze, and understand patterns and relationships [9]. GIS connects what with the where. It helps to connect geography with data which in turn drive decision-making in real world applications. Some of the largest problems of our planet as climate change, natural disasters and population dynamics, accident locations are best understood spatially. These problems can be solved through spatial analysis. Spatial analysis gives perspective in understanding relationships between spatial and attribute data.

According to Michael DeMers, GIS is a collection of software, hardware and people [10] as illustrated in Fig 2.1. The essential parts of a GIS listed by DeMers are

- Data and information
- Computers, input and output technology and software
- Geographic and related concepts that drive the analysis
- People, such as operators, managers, consultants, vendors, etc.
- Institutions and organizations within which the GIS exist.

Thus the goal of any GIS is to visualizations data, which is in fact the essential key to explore and find solution to the nature of the data set, and patterns that the dataset revels as related to the current problem[11]. Therefore, data is the backbone of any GIS. GIS database is called 'Geodatabase'. A geodatabase is a database which gives reference to locations on the earth and these references are pointed in the map as the location of the place. Geodatabases can be:

- Vector data which is spatial data represented as features (points, lines and polygons)
- Raster data which is cell-based data as aerial imagery, 2D & 3D maps and digital elevation models.

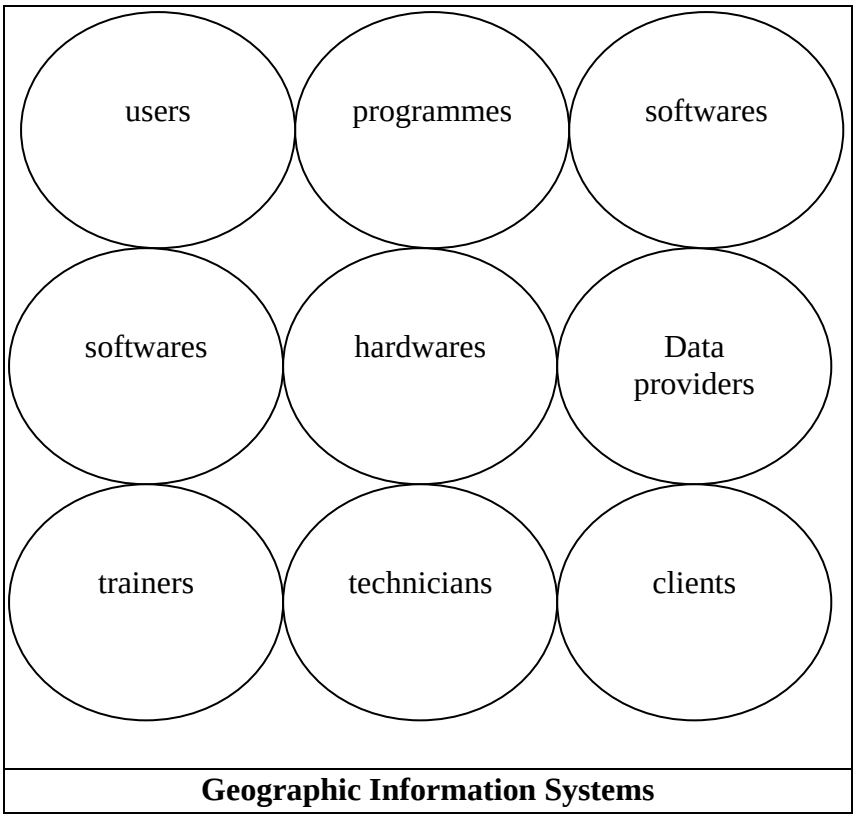


Figure 2.1: GIS collective. **Re-printed from** [10]

Because of importance and usefulness of GIS, much research has been done in extracting information from texts related to specific geo-locations. It had been estimated that 80% of an organization data has geographic location [12]. This shows the importance of GIS. Geographic Information Systems connects what with the where.

GIS is a part of geographic information retrieval (GIR). As defined by Larson in paper [13] , GIR is a result of combined research in Information Retrieval (IR), user interfaces, GIS and Database Management System. GIR has recently gained a lot of attention in everyday communication as it allows users to constrain queries to specific locations, while traditional search engines do not give extra notice on the geographical information. People often use place names and spatial relations to describe where they are, to give navigational instructions, to inform the location of events [14]. Services that provide geographic information using places names, such as navigation and routing systems, transportation timetables, environmental forecasting, map-based websites, and Web search engine. Although many efforts are made by scientists, services that provide geographic information still cannot unambiguously distinguish and perform enough spatial reasoning with names of places by human expressions. Place names frequently occur in text documents with geographic context. People desire geographically-oriented information for instance

- Natural resources managers who would like to retrieve information pertinent to specific areas.
- Earth scientists who would like to locate publications which discuss certain locations.
- Historians who would like to retrieve documents about specific areas.
- Journalists who would like to locate documents pertinent to current events.

- Tourists who would like to locate hotels and other relevant information to areas being considered for trips.

In searches such as those above, "relevant" documents may cross many subject areas and be of many different types. This often makes a comprehensive search difficult.

Geographic location is the shared attribute which can draw together these disparate subjects and document types to allow a cohesive search. In GIR a geographic query is a structured triplet of <*theme*><*spatial relationship*><*location*> and whose main aim is to improvise information retrieval based on geographic specific focusing on access to documents that are not structured.

As with the increase in importance of geographic features for web resources, much work has been performed to improve the accuracy of detecting web location and estimation. At present GIR research are done in major directions as: 1) exploiting geographic information sources, 2) identifying and disambiguating names of places, and 3) developing effective computation approaches. Tremendous amount of work has been done which focus on identifying geographic references and disambiguating names of places. These are referred to as geoparsing and geocoding respectively.

2.2 Information Extraction

Information extraction (IE) is usually defined as the process of selectively structuring and combining data that are explicitly stated or implied in one or more natural language documents[15]. Much of today's world information is found in natural language text but getting meaningful information from it is a challenging task. IE aims at extracting this information from unstructured natural language text and convert to a structured form suitable for computer manipulation [16]. IE commonly have three basic tasks: named entity extraction, relation extraction, and event extraction.

2.2.1 Named entity extraction

Early 1990's showed the beginning of named entity extraction where focus was primarily on extraction from journal articles. NER locates and extracts entities in natural language text and identifies its types [16] [17]. Later studies showed research in diverse domain as in biomedical entity extraction [18] [19] while other study focus on general entities by extracting content automatically [20].

NER use Gazetteer and regular expressions to extract entities. But using regular expressions is not a good choice as it has to be tuned manually to accommodate new domains and this could be difficult and tedious. Thus named entity extraction using Hidden Markov Model (HMM) are proposed to recognize and classify names, times and numerical quantities [21]. Zhou in

the paper Named entity recognition using an HMM-based chunk tagger [22] propose use of both linguistics based approach and machine learning based in extracting entities.

2.2.2 Relation extraction

Relation extraction detects and characterizes the semantic relation among extracted named entities. The relation extraction requires detection and characterization of relations between various entities present in the text. Some of the types of relations are:

- Organization- names of organization,
- At- to defines location relationship
- Near- to identify relative location
- Physical- to identify place name.

Relation extraction uses the text between the two entities such as Part-Of-Speech (POS) tagging and dependency tree relationships to construct the connection between the entities [16]. POS and dependency tree are discussed in the later part of the thesis.

2.2.3 Event extraction

Event extraction detects the presence of the events that the entities participated in and identifies their types. For instance, Automatic Content Extraction [20] defines five types of events: Interaction, Movement, Transfer, Creation, and Destruction. Event extraction is a non trivial problem as an event is usually expressed indirectly in a sentence. With respect to this thesis event means accident event. Accident events are grouped as keywords and they are extracted based on some rules.

2.3 Natural Language Processor

Report from Georgetown–IBM [23] experiment in 1954 showed the beginning of machine learning which was developed jointly by the Georgetown University and IBM. This was a small scale public demonstration of a Russian-English machine translation. Only with 250 words and six ‘grammar’ rules, the project marked the beginning of automatic translation of languages. NLP is a combined field which involves computer science, artificial intelligence, and computational linguistics which is involved with the interactions between computers and human languages. NLP is an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to do useful things [24]. Introduction of dependency tree is one of the noticeable advancement in NLP in recent years. A dependency tree describes the semantic relationships that exist between pair of words in a sentence. The sub-trees produced by the dependency trees are helpful in classification. Dependency tree has also been applied in automatic

information extraction using various models. Mark Stevenson in his paper "Dependency Pattern Models for Information Extraction [25] compared a variety of pattern models and found that the best performance was observed from the models which use the majority of relevant portions of the dependency tree without including irrelevant sections. Grishmanin explained the Tree-Based Pattern representation for Japanese information extraction where a pattern is denoted as a path in the dependency tree of a sentence [26]. Tree-based patterns are found superior to the patterns derived from un-annotated text. Some research were made to introduce dependency tree in relation extraction by capturing the shortest path between the two entities in the dependency tree kernel [27].

There are many open source NLP tools to construct a dependency tree, parts of speech tagging and named entity recognition available. Some of them are as Stanford's Core NLP Suite [7] suitable for processing English, Chinese, and Spanish. This includes tools for tokenization, part of speech tagging, grammar parsing (identifying things like noun and verb phrases), and named entity recognition. Natural Language Toolkit [28] is suitable for python programming language. Similar to the Stanford library, it includes capabilities for tokenizing, parsing, and identifying named entities as well as many more features. Apache OpenNLP [29] uses a different underlying approach than Stanford's library, the OpenNLP project is an Apache-licensed suite of tools to do tasks like tokenization, part of speech tagging, parsing, and named entity recognition.

Chapter 3 Choice of Natural Language Processor and Geo-Coding service

This thesis aims at extracting information from the accident news report and locating the accident event in map as precisely as possible. So it is important to make a wise choice for information extraction toolkit as well as the geocoding service. So this chapter describes the most popular NLP toolkits and popular Geo-coding services available and the reason for choosing specific NLP toolkit and geocoding service.

3.1 Natural Language Processing Toolkits Choice

The processing power of NLP is based on the various evaluation criteria which are discussed in [30]. The Evaluations criteria must be designed in a way to address issues which are relevant to the specific task domain of the NLP system in study, thus the NLP systems for different task requires different evaluation criteria. While evaluating NLP systems more attention should be given on the "environmental" factors that are associated with NLP systems in actual use. A list of functional requirements for NLP toolkits are sentence segmentation, tokenization, Part-Of-Speech (POS) Tagger, lemmatization, and co-reference resolution. These functionalities are important criterion in evaluation of NLP toolkits.

Popular NLP toolkits available are:

- Stanford CoreNLP [7]
- Natural Language Toolkit (NLTK) [28]
- Apache OpenNLP [31]F

The choice of the NLP toolkit is based on the efficiency of tagging location entities, POS tagging, programming language it supports and ease of implementation. Apache OpenNLP is not preferred as it does not take into account the feature of POS tagger while identifying named entities, thus Apache OpenNLP do not perform well as Stanford CoreNLP. It sometimes fails to detect simple single tokens. NLTK is not selected for this thesis as it is requires knowledge of Python while the author being more familiar with Java programming language. Thus Stanford CoreNLP is chosen for this thesis as it supports Java programming language, it can be easily obtained and is easy to implement. Also it has accuracy rate of about 92.99% for tagging English news [32].

As Stanford CoreNLP toolkit is be used in the thesis, it is important to describe its working principle. Also to understand the principle of natural language processing it is necessary to understand how the Stanford CoreNLP works. Stanford CoreNLP has Parts of Speech tagger and Named Entity Recognizer (NER). NER labels sequences of words in a text which are the names of things, such as person and company names, or gene and protein names. It comes with well-engineered feature extractors for NER, and many options for defining feature extractors. Included with the download are good named entity recognizers for English, particularly for the 3 classes (PERSON, ORGANIZATION, and LOCATION).

Stanford NER is also known as CRFClassifier. The software provides a general implementation of (arbitrary order) linear chain Conditional Random Field (CRF) sequence models. As per the requirement of this project, only Location class is used out of the three classes [7]. Figure 3.2 shows the en example of Stanford NER.

Stanford POS Tagger is a piece of software that reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc., although generally computational applications use more fine-grained POS tags like 'noun-plural'. This software is a Java implementation of the log-linear part-of-speech taggers. Fig 3.1 shows the working principles of Stanford POS [7].

Stanford CoreNLP provides a set of natural language analysis tools. It can give the base forms of words, their parts of speech, whether they are names of companies, people, etc., normalize dates, times, and numeric quantities, and mark up the structure of sentences in terms of phrases and word dependencies, indicate which noun phrases refer to the same entities, indicate sentiment, extract open-class relations between mentions, etc. Stanford CoreNLP is fast, reliable analysis of arbitrary texts. It has the overall highest quality text analytics and support for a number of major (human) languages providing Interfaces available for various major modern programming languages [7].

The below figure shows the output from Stanford CoreNLP for the input text “A truck overturned when it was entering a four-lane motorway on the E6 road near Svinesundparken in Østfold county.”

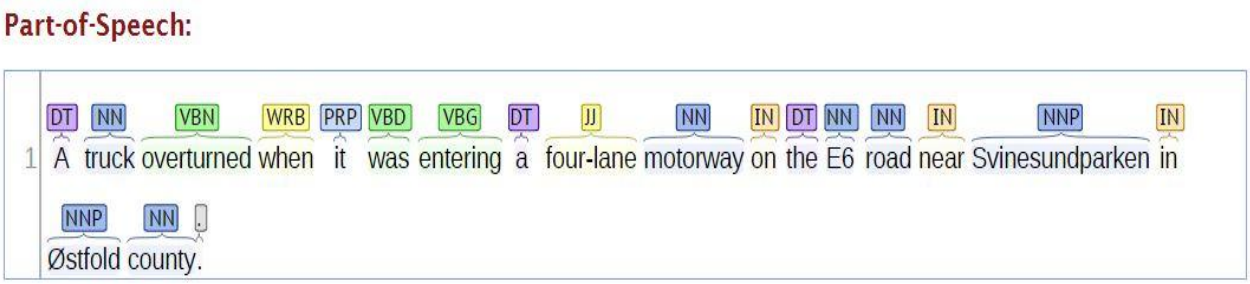


Figure 3.1: POS output from Stanford CoreNLP (screen shot of Stanford Online POS Tagger)

In the above figure Stanford POS tags the words of the sentence based on the parts of speech of the word. IN stands for Preposition or subordinating conjunction, NN stands for Noun, singular or mass, NNP stands for Proper noun, singular. All the place names are tagged as NNP in the text as Oslo_NNP. Appendix A.4 describes in the full forms in detail.

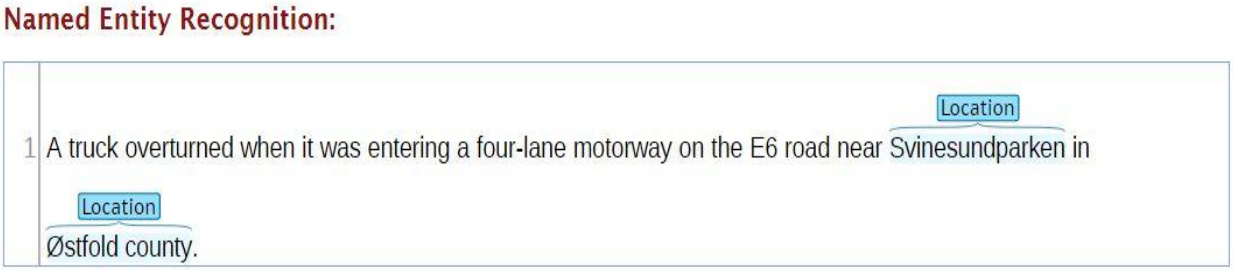


Figure 3.2: NER output from Stanford CoreNLP (screen shot of Stanford Online NER)

Basis dependencies represent a simple description of the grammatical relationships in a sentence. It can help to extract textual relations. Sanford dependency tree represents all sentence relationships uniformly as typed dependency relations [33].

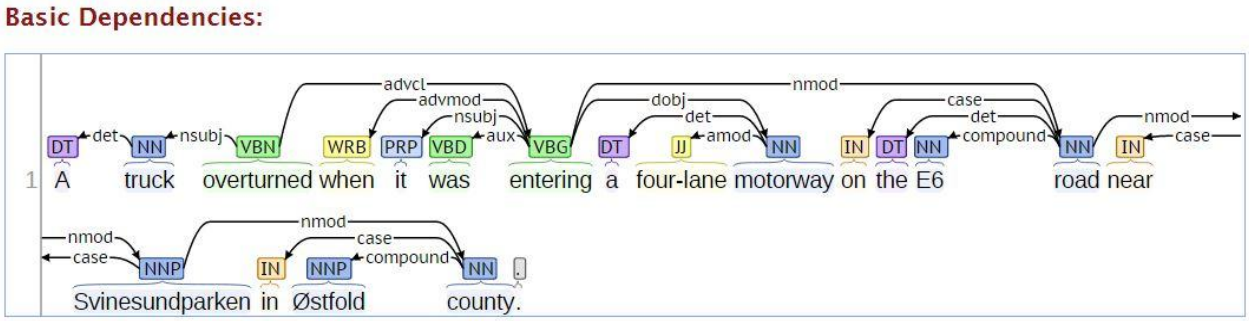


Figure 3.3: Dependency tree output from Stanford CoreNLP (screen shot of Stanford Online CoreNLP)

Stanford CoreNLP works very well for one sentence to give a dependency tree among the words in the text. But when it comes to extract and produce dependency tree of more than two sentences, Stanford CoreNLP fails to produce the desired result. As for in our case we intend to pin point the location in the map by extracting all the relevant location details found in the news report which are related to the event. Stanford CoreNLP cannot produce the relation with two sentences making it unfeasible for us to use the dependency tree produces by it. As for instance for the same news report input "A truck overturned when it was entering a four-lane motorway on the E6 road. The accident took place near Svinesundparken in Østfold County.", the dependency tree is explained in figure 3.4 which clearly shows that the Stanford CoreLNLP could not bring out the relation among words of more than one sentence.

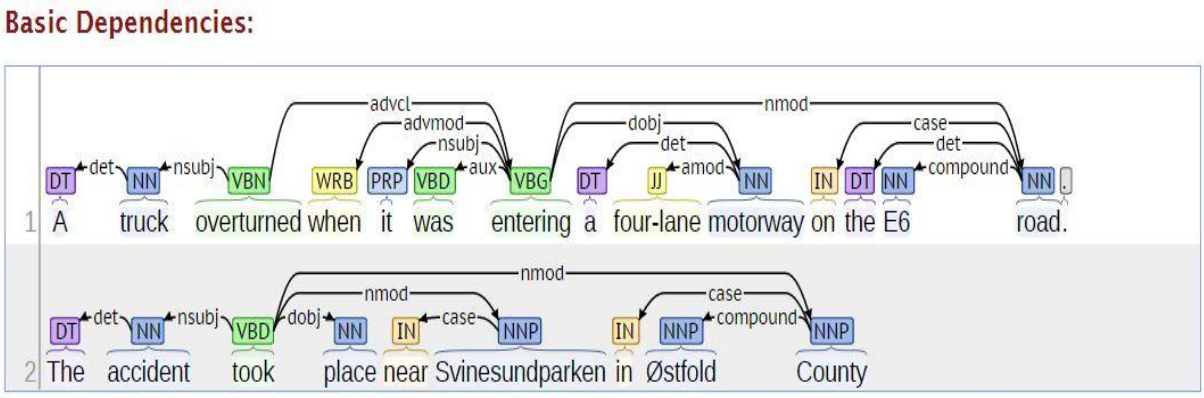


Figure 3.4: Basic Dependency tree output from Stanford CoreNLP for more than one sentence.

3.2 Geo-Coding Service Choice

Geocoding is the process of converting addresses (like "Jernaldervien 57, Stavanger, NO") into geographic coordinates (like latitude 58.94454 and longitude 5.692862), which can be used to place markers on a map, or position the map. Geocoding gives geographic name an actual location in the map based on the relevant information provided to that location [34].

Some of the popular geocoding software packages available today are

- ESRI PC-based Geocoder
- Google Earth- web based
- Google Maps API-web based
- Yahoo Maps API – web based

These services also provide reverse geocoding functionality. The choice of the geocoding service is based on the cost and the queries per day provided. Environmental Systems Research Institute (ESRI) [9] provides a 60 day free trial of ArcGIS software. Pricing of the full version varies depending upon the user. Google Earth provides paid service. Although Yahoo Maps API free service but the geocoding service is limited to 5,000 queries per IP address per day [35].

So Google Maps API is chosen for both geocoding and reverse geocoding services as the service is free of cost and with no limit on queries per day.

The geocoding service is provided by Google Maps Geocoding API directly via HTTP request. The Accessing of the Google Geocoding service is asynchronous so the service is accessed within the code with the help of google.maps.Geocoder object. The Geocoder.geocode method initiates a request to the geocoding service, passing it a GeocoderRequest [34]. When an address is successfully geocoded, a Geocoding Responses is returned with multiple objects as formatted address, partial match address, geometry containing location with geocoded latitude, longitude value, route type, intersection (only for major national roads) etc. This thesis aims at getting the accident location as precisely as possible. Many times there is a need to get the intersection of the local streets and for that the full geometry of the road is required. But Google geocoder only returns a one latitude longitude coordinate pair. Thus Google geocoding service is suitable only for getting a latitude longitude pair. In the later part this importance is discussed.

Geocoding is a time and resource consuming task. To use the Google Maps Geocoding API, an API key is required [34]. Input to the Google Geocoding service is location parameter while the output format can be either json or xml. Required parameters in a geocoding request are:

- Address: the street or location which is to be geocoded
- Key: application's API key which identifies the application for purposes of queries permitted.

The JSON out from Google Geocoding service for an address ‘Munkedamsveien’ looks like

```
{
  "results" : [
    {
      "address_components" : [
        {
          "long_name" : "Munkedamsveien",
          "short_name" : "Munkedamsveien",
          "types" : [ "route" ]
        },
      ],
      "formatted_address" : "Munkedamsveien, Oslo, Norway",
      "geometry" : {
        "location" : {
          "lat" : 59.9124126,
          "lng" : 10.72755
        },
        "location_type" : "GEOMETRIC_CENTER",
        "viewport" : {
          "northeast" : {
            "lat" : 59.9148377,
            "lng" : 10.7322764
          },
          "southwest" : {
            "lat" : 59.91077809999999,
            "lng" : 10.7215481
          }
        }
      },
      "place_id" : "ChIJ9SAfZIBuQUYRMjTFgoj_mO4",
      "types" : [ "route" ]
    }
  ],
  "status" : "OK"
}
```


Figure 3.5: Google Geocoding json output

As shown in above figure the JSON response has two main elements:

- "status": 'OK' means the geocoding was successful and return at least one geocode result, 'ZERO_RESULTS' means a successful geocode but no result returned [34]

- "result": has an array of address and geometry information of geocoded results. Normally only one result is returned but the geocoder may return more than one result in case of ambiguous address query [34].

In the above figure 3.5, the street address 'Munkedamsveien' is of types 'route'. But the json outputs only one coordinate's pair for the street route. But in order to find an

approximate distance such as ‘near a road’, ‘on the road’ or ‘between two roads’, a full geometry (coordinates) of the street is required. The Google Geocoding service is suitable in case when only one pair of coordinate is required i.e. for geocoding place or locations but not when a full geometry of the street or road is needed.

To the best of authors knowledge there is no service that returns the full geometry coordinates of a street. Thus a separate database is designed from the openstreetmap (OSM) shapefile in order to extract the full street geometry. The database design is discussed in Section 4.2.2.

Reverse geocoding [34] service is also provided by Google Maps API. Reverse geocoding is the process of converting a location on a map i.e. converting latitude, longitude coordinates or place ID into human-readable address. Place ID is a unique identifier for that place. Reverse geocoding is the opposite process of geocoding. Like geocoding, reverse geocoding requires parameter as:

- Latlng: latitude and longitude values which specify the location and convert into human-readable address format

OR

- Place_id: Place unique identifier of that place to retrieve its human-readable address[34]
- key : Google Maps API key

Reverse geocoding also has two main response elements similar to geocoding. They are:

- ‘result’: an array of most closest addressable location with a certain tolerance.
- ‘status’: ‘OK’ means a successful reverse geocoding with no errors occurred and it returns as least one address, ZERO_RESULTS’ means a successful reverse geocoding but no result returned[34].

Thus reverse geocoding produces result on the basis of the input parameters passed on it. One thing to be noted is that reverse geocoding process is an estimate. Thus geocoder returns the closest addressable location within an approximation.

Chapter 4 Design and Methodology

This chapter describes the design principles and methods used in automatic text extraction and geocoding and reverse geocoding to locate the event in the map.

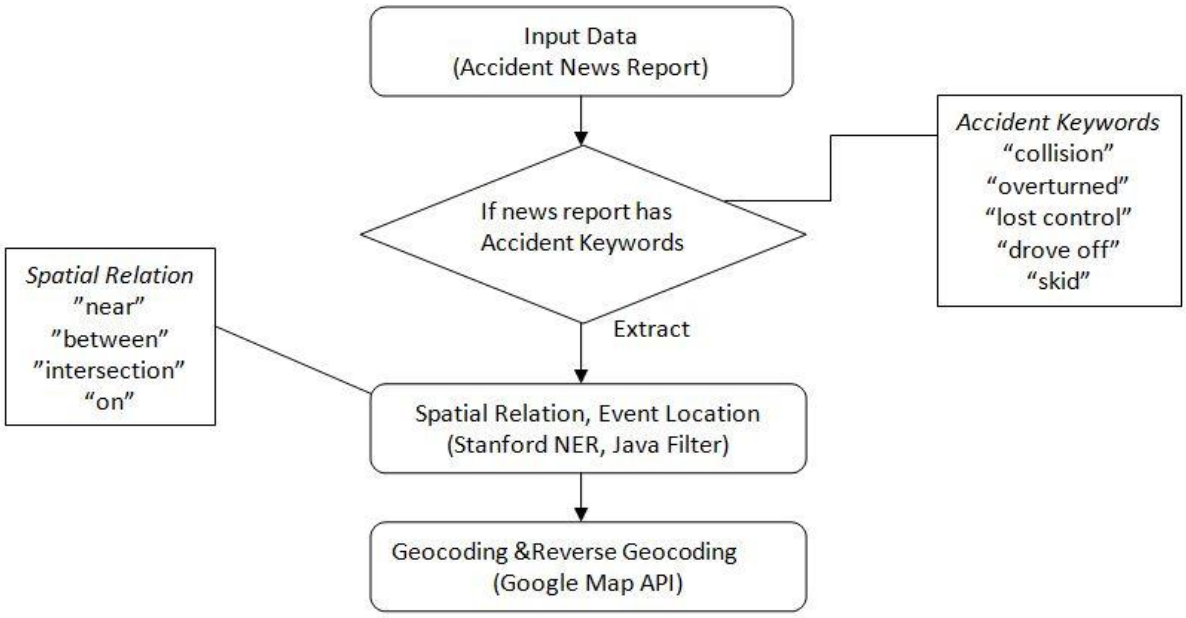


Figure 4.1: Information Extraction and Geocoding Process.

The design process has two main phases. First is the information extraction phase which detects and retrieves the accident locations and the spatial relation. Accident locations are the places where an accident event occurred and spatial relations are connecting words as 'near', 'between','intersection' etc which helps in approximating the accident event. Second is the geocoding and reverse geocoding phase. In this phase geocoding and reverse geocoding is done based on the spatial relation with the location. Both the phases are discussed in detail in this chapter.

4.1 Information Extraction

The information extraction phase in this project has two main parts. First part detects and filters the news report that contains accident information in it. And the second part extracts the location entity and the spatial relation of the location in the news.

4.1.1 Detect the Occurrence of Accident Event

With the advancement in information technology, there is a tremendous boost in the digital media including newspaper. There are more than hundred leading national news papers around the world. In Rogaland (Norway) alone there are around 20 daily newspapers [36]. It

needs lot of computational time to process all the newspapers for the presence of accident events. Thus there is a need to filter the newspaper data to work with this project. That is to process only those news reports that has accident event in it. In this step, we study how to classify Accident Report news and non-Accident Report news which is a classification problem.

The Following code describes the Accident News Detection system.

```
processAccidentNewsDetection(News_Report) {
if (News_Report contains(Accident_Keywords && Accident_Location) {
Accident Incident Detected in News Report
  }
   Else {
        No accident Detected in the News Report
        }
Return News_Report.
}
```

Figure 4.2: process to differentiate Accident Report and non-Accident Report

The efficiency of the above method depends on the Accident Keywords that are used to filter and separate the news report to the ones that has accident news report and those that do not. A list of Accident keywords that are used as filter to sieve the news report are: collided, accident, skid, collision, lost control, ran off, and overturned. More words can be added to this list to increase the news report capturing power.

4.1.2 Extraction of Event and Location from Accident Report

After classifying the news report as Accident Report news and non-Accident Report news, the next step is to extract the location entity and spatial relation of the location entity from the Accident Report news. The problem studied in the step is defined as follows.

Problem definition: Given an Accident Report news, extract the Accident Location Named Entities and the spatial relation.

Thus the above problem is clearly an information extraction problem. The information extraction problem has been studied by many researchers and many existing techniques are reported in the previous chapter, one of the most notable being Stanford CoreNLP. Despite the progress made, we will explain how the existing solution does not work well in this thesis. To work with the project, a novel technique is proposed which is a combination of rule generation (supervised) and pattern data mining (unsupervised).

The following figure shows the proposed technique.

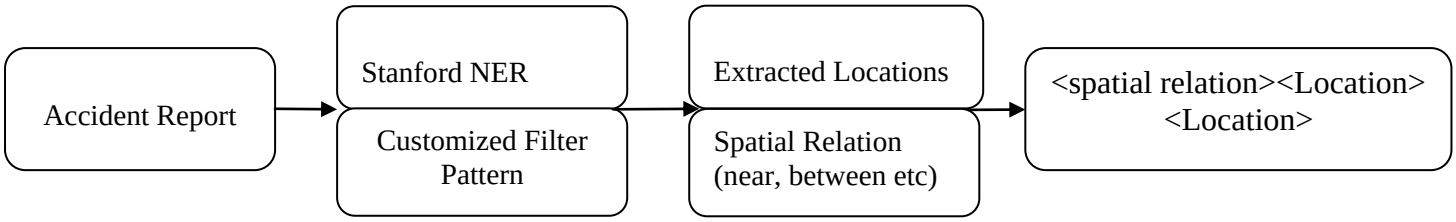


Figure 4.3: Design of Information Extraction process

Stanford CoreNLP consists of NER and POS tagger. Extracted Accident Report is passed to Stanford NER. It can extract named entities such as people, location, and organizations from an unstructured data. But in this project we only need the location named entity. Figure 4.4 explains how Stanford NER extracts the named entity from a data.

Problem with Stanford CoreNLP is that it can very efficiently build a dependency tree for one sentence and extract relevant paths to form the sequence data for mining and learning. But it fails to build the dependency tree to form paths for more than one sentence as described in Figure 3.4. Thus Stanford dependency tree parser does not work in our case. To deal with the issue, costumed pattern mining is performed which acts as filter to extract relevant information from the data.

Customized Filter Pattern is the process of extracting specific patterns from the document. Various filters are made to put the data into categories which are later processed to derive information from them. Filters are of following types:

- Street Name Extraction Filter
- Location Extraction Filter
- Road Name Extraction Filter
- Tunnel Name Extraction Filter
- Spatial Relation Extraction Filter

Street Name Extraction Filter

One of the main of this project is to point the accident location as precise as possible in the map. It has been seen through research of accident news report trend, most of the time the whole paragraph of the news report describes the precise location of the event. Accident location can be detected from one sentence but it may or may not be precise. Thus in order to find out the most precise location of accident from news report, the problem cannot be solved through sentence level.

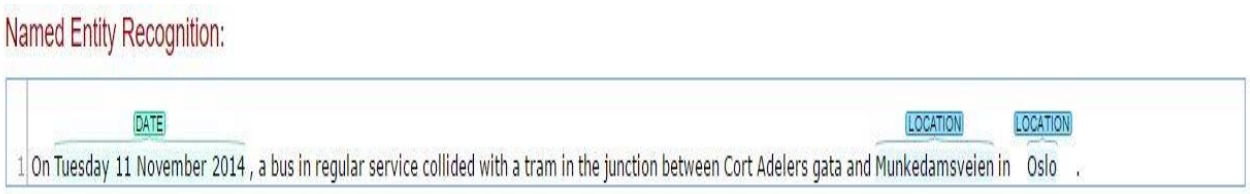


Figure 4.4: Stanford NER some time fails to tag locations and street names.

For the news report published in AIBN Annual Report from 2014 “On Tuesday 11 November 2014, a bus in regular service collided with a tram in the junction between Cort Adelers gata and Munkedamsveien in Oslo.” Stanford NER extracts *Tuesday 11 November 2014* as the Date class, *Munkedamsveien* and *Oslo* as the Location class. Stanford NER recognizes country, capital, and city names very efficiently. But it sometimes fails to recognize regional location names, more preciously it is less robust for regional locations names. As for example in the above news report, Stanford NER fails to recognize Cort Adelers gate as location name. Thus, Stanford Named Entity recognizer is always not working accurately in all languages link in case of Norwegian place names. Places like “Aasta Hansteens vei” and “Cort Adelers gata” which are separated by a space are sometime recognized as two different places and sometime it does not recognize at all as shown in Figure 4.4. Through manual research, it was found that in Oslo alone out of 1839 street names, 230 street names are separated and are in two parts. In order to get accurate occurrence of event we need this places to be tagged as one. A Street Name filter is designed to extract such places and street names.

Norwegian street names often ends with “vei” or “gata”. The idea is to capture the presence of these keywords and extract the street name associated with it. Thus to deal with this task, Stanford POS is used which tags and filter various spatial connecting words that bring a meaningful sense with the event and the place name. For example in Fig 4.2, “Cort Adelers gata”, is tagged as “Cort_NNP Adelers_NNP gata_NN”. That is if the word before “gata” is tagged as NNP in the sentence, it is very much likely that it is a place name.

The Following code describes the Street Name filter.

```
function Street_Names_Filter (NewsReport) {
   StreetNames = NewsReport_WithTags.split("\\s");
        if (StreetNames contains("gata") or ("vei")) {
          Extract it and the word before it that has tag _NN or_NNP.
                        }
                }
                return RoadName;

        }
```

Figure 4.5: Street Name Extraction Filter

Location Extraction Filter

This filter extracts only the locations that has a respective spatial relation it the news report. For instance for the report type ‘near Svinesundparken’, ‘between Cort Adelers gata and Munkedamsveien’, this filter extracts the location that has the near and between relation.

```
Function SpatialRelatingLocationFilter(NewsReport) {
  News_Report = NewsReport split("\\s");
        if (News_Report contains("near")) {
                Extract word before it that has tag _NN or_NNP.
       }
return Location;
}
```

Figure 4.6: Spatial relation Location extraction filter

Tunnel Extraction Filter

Another problem in extracting location entity is capturing locations like tunnel. In the news report it is often reported that an event occurred near a tunnel as Gudvangatunnelen. Stanford NER efficiently extracts such tunnels but if the tunnel name is of type “Oslofjord tunnel”, it can only extract the place Oslofjord. Oslofjord is a vague place and not precise to point the event location that took place. Instead it is expected that Oslofjord is captured as Oslofjord tunnel. So a filter is needed to extract such tunnel names.

The following code explains the tunnel extraction filter.

```
function Tunnel_Extraction(News_Report) {
  News_Report = NewsReport_WithTags.split("\\s");
  if (News_Report contains("tunnel")or ("tunnelen")) {
    Extract it and the word before it that has tag _NN or_NNP.
   }
return Tunnel_Name;
}
```

Figure 4.7: Tunnel Name Extraction Filter

Road Name Extraction Filter

In order to get the most precise location of accident it is a must to extract the road and street names. Stanford NER does not provide road names entity recognizer. Also extracting road names with the help of POS is not a smart choice as it will then require lots of tag to process to get the desired road names. Road names are tagged as NN (Noun, singular or mass) by the POS for example E6_NN and it can easily be confused with other NN tags in the sentences as timber_NN, trailer_NN, vehicle_NN.

Norwegian road network can be European route (European abbr. E), Norwegian national road (Riksvei/Riksveg abbr. Rv), Norwegian County Road (Fylkesvei in Norwegian language abbr. Fv). Thus a costumed pattern can be designed to detect and extract these road

name present in the accident report. Also this road name defines the first precise location choice in locating the accident event. (We are working with a data base to we need to made a search that match the database, so we need to substitute the road names and remove and keep only the number in case of the national and county roads.)

The following code describes the road name Extraction filter.

```
function RoadNameExtraction(News_Report) {
  News_Report = NewsReport_WithTags.split("\\s");
   if (News_Report contains("road")) {
            Extract word before it that has tag _NN or_NNP.
           }
Return Road_Names;
}
```

Figure 4.8: Road Name Extraction Filter

Spatial Relation Extraction Filter

Spatial Relations are those terms like ‘near’, ‘between’, ‘intersection’ which helps define the position of the event when combined with the location. For example, an accident report states ‘An accident took place near Svinesundparken on E6 road’. Then the event location must not be far away from the place.

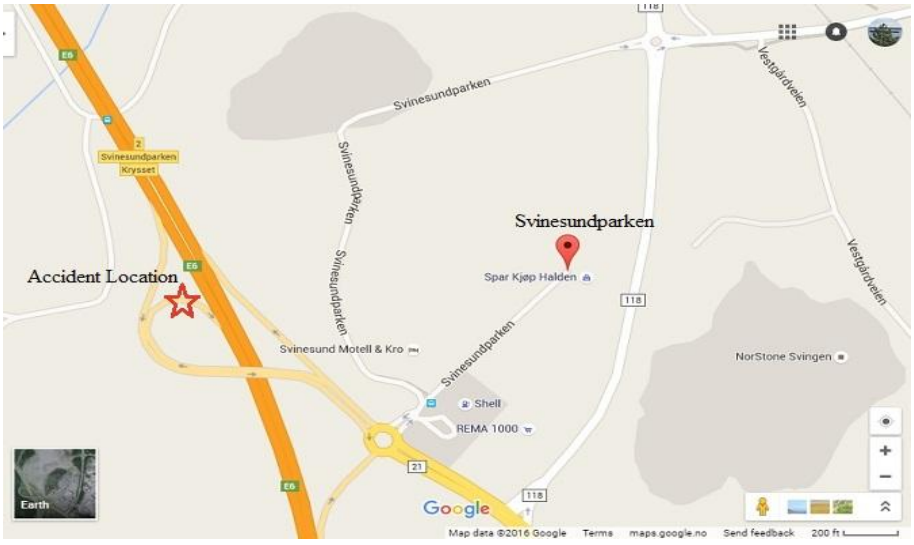


Figure 4.9: Actual accident location. Manually pointed by AIBN

In the above figure 4.6, accident location is marked by a red star on E6 road. The accident location is near Svinesundparken, thus the spatial relation helps how to locate the accident place in the road or street.

The following code explains the Spatial Relation Extraction filter:

```
Function Spatial_Relation_Filter(NewsReport) {
  Spatial_relation_words = NewsReport.split("\\s");
 if (Spatial_relation_words contains("intersection")or ("between") or ("near")) {
        Extract it the word
          }
return Spatial_relation_word;
}
```

Figure 4.10: Spatial Retation Extraction Filter

The whole process of information extraction from text is explained in the algorithm.

```
Function informationExtraction(News_Report) {
   if(Accident Detection )    {                  // events as accident, collision etc
         POStagged := StanfordPOS(paragraph)   // parts of speech tagging
         NERtagged :=StanfordNER(paragraph)  // NER tagging
        StreetName:=StreetNameExtractionFilter(POStagged) //Extract Street Names
        LocationName:=Location Extraction Filter(POStagged) //Extract most precise location names
        RoadName:=Road Name Extraction Filter(POStagged) // Extract road names
        TunnelName:=Tunnel Name Extraction Filter(POStagged) //Extract tunnel name
        SpatialRelation:=Spatial Relation Extraction Filter(POStagged) // Extract spatial relating words
         }
        Else {
            Non-Accident news report
               }
  }
Repeat for all Paragraph
```

Figure 4.11: Information Extraction Process from News report

4.1.3 Experiment

4.1.3.1 Data Collection

This thesis only considers the accident report published in English language. There are around 20 online newspapers in Rogaland area alone. But majority of the newspapers are in Norwegian language. So to test the information extraction filters which are described in

Section 4.1.2 , we manually collected the accident report from Accident Investigation Board Norway [37] and Google News [38]. The data collected is in text format. This thesis is mainly focused on the accident events within Norway. The reason of choosing AIBN accident report is because the accident report describes the place and location of events enough to point the location in map as precise as possible. Although accident news reported in twitter is available online in real time, they cannot be used for the purpose of this thesis because they do not describe the accident event locations enough. Thus tweeter tweets are not the best choice data source for this thesis.

4.1.3.2 Result from Information Extraction

To test Stanford NER for this thesis, 1839 street names from Oslo are manually collected. These street names are mostly of two types. First type consists of street names with only one token like Gimleveien, Gjøvikgata and the second type consists of street name with two more than two tokens like Fjørtofts gate , Cort Adelers gata, General Ruges vei etc. Out of 1839, there are around 1100 street names of first type and the rest are type two street names. Stanford NER successfully tagged the first type as location because the Stanford NER is designed to process words one token at a time. But for the type two street names it fails to recognize them as locations.

The same street names are tested in the Street Name Extraction Filter explained in Section 4.1.2. The purpose of this filter is to extract the type two street names. So the type one street names are not tested by this filter as they are already recognized by the Stanford NER as location as expected. This filter successfully extracted the type two street names that ends with ‘vie’,‘veien’, ‘gata’ and ‘gate’. There are 230 street names of type two that ends with ‘gate’ and 282 with ‘gate’. Street Name Filter manages to capture them. Street naming system in Norway is diverse. Street names may end with ‘vie’,‘veien’, ‘gata’ , ‘gate’, ‘plass’, ‘allé’ etc. Thus a very good and stronger filter is required to extract all these street names.

4.2 Geocoding Location Named Entity

Geocoding converts addresses into geographic coordinates. It gives geographic name an actual location in the map based on the relevant information provided to that location. [23]

After extracting location and the respective spatial relation from the Accident News Report, the next step is to map these Location NEs to geographical location based on the < spatial relation ><location> pair.

4.2.1 Disambiguation of geocoding

Geocoding is an easy task only if the mapping relationship is one-to-one between the Location NE and geographical location, but unfortunately this is not true. The relationship is many-to-one in many cases. A search is made in geographical database to find the location matches the Location Named Entity. Geographical database such as GeoNames has severe problem of ambiguity. As for instance search "London" in GeoNames database results in 687 results in more than 9 countries. To solve this problem for this project geographical database search is restricted Norway only i.e. reducing the search domain. But even at country level there exist some ambiguity in Location names especially in the street names. For example for a street name "Kongsgata" in Norway, geographical database search give us two location, one at Kongsgata, Stavanger and the other at Kongsgata, Ålgård [13].

To work with this thesis a geographical database for the Norwegian roads and places is designed.

4.2.2 MySQL Geographical Database

OpenStreetMap provides free access to all the latest geographical database of the world including a country's road network. To design the database we download the osm file of Norway which provided freely by Geofabrik [39]. Geofabrik is a German based consulting and software development firm specializing in OpenStreetMap services. The osm file contains a shapefile(.shp). A shapefile [40] is an Esri [41] vector data storage format which stores the location, shape, and attributes of geographic features. It is stored as a set of related files and contains one feature class. According to Esri a shapefile has three mandatory files

- feature geometry (`.shp`, shape format)
- a positional index of the feature geometry (`.shx`, shape index format)
- columnar attributes for each shape (`.dbf` , attribute format)

The osm file has shapefile of buildings, landuse, places, points, roadways, waterways. For the geocoding purpose of this thesis only roadways shapefile of Norway is required. Place geocoding is done with the help of Google Map API.

In order to design the route network database, the roadways shapefile (.shp) is first converted into geojson. A geojson file looks like


```
{"name":"roads","type":"FeatureCollection",
"features":[
    {"type":"Feature","geometry":{"type":"LineString",
     "coordinates":[[5.3356843,60.3907194],[5.3358712,60.390621]},
     "properties":{"osm_id":"1227","name":"Kalfarveien","ref":"E6","type":"primary",
     "oneway":1,"bridge":0,"tunnel":0,"maxspeed":50}
    }
   ]
}
```

Figure 4.12 : Geojson file example

From the above geojson information only ‘coordinates’, ‘osm_id’, ‘name’, ‘ref’ properties are extracted to create the geographic database for this thesis. The reason for choosing only this information’s is that it is sufficient to get the accident event location with the chosen information. For example for the accident event that took place on E6 road Near Svinesundparken in Østfold County, E6 is the ‘ref’, Svinesundparken in ‘name’, latitude longitude details for Svinesundparken are found in ‘coordinates’

Geojson Processing:

The database is creates in MySQL [42]. The reason for choosing MySQL database is that it provides easy implementation of spatial data types which correspond to OpenGIS classes. Some of these types hold single geometry values as geometry, point, linestring polygon. MySQL supports two types of spatial data formats:

- Well-Known Text (WKT) format : for exchanging geometry data in ASCII form
- Well-Known Binary (WKB) format: for exchanging geometry data as binary streams represented by BLOB values containing geometric WKB information [42].

In order to export the geojson information into MySQL, first it needs to be processed in the format supported my MySQL. Geojson file is processed to get only the desired information in CSV format which can be easily be imported into MySQL database. As csv file looks like:

osm_id	Name	ref	LineString
1227	Kalfarveien	E6	LinsString (5.3356843 60.3907194, 5.3358712 60.390621)

Figure 4.13 : csv file example

Now the file is ready to be imported in the MySQL database. The following Figure 4.14 describes the procedure for creating a database and importing the csv file that contains the road geometry details of Norway.

Create Table MySQL Table:

```
create table norway_route_info (osm_id int primary key, name varchar(255), ref
nvarchar(255),    Route (linestring);
```

Import csv File

```
load data local infile ' Path of csv file'
into table `Norway_route_info`
fields  terminated by  ';'
lines terminated by '\n'
(osm_id, name, ref,@Route)
set wkt = LineStringFromText(@Route);
```

Figure 4.14: creating database and importing cvs file to database.

4.2.3 Geocoding case study

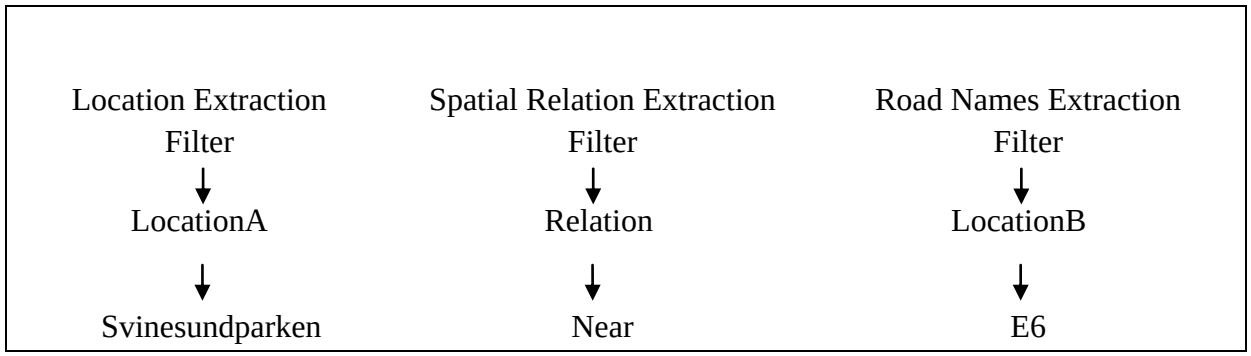
After the database is ready, the next step is geocoding. In order to locate the accident events precisely in the map, the following cases are worth discussing.

1. ‘near’ case:
2. ‘between / intersection’ case:
3. ‘inside tunnel’ case:

1. ‘Near’ case:

A near case has a Location and spatial relation to the other location. For this case the locations are extracted from the Location Extraction Filter and Road Names Extraction Filter while their spatial relation is extracted from the Spatial Relation Extraction Filter as discussed in section 4.1.2.

Example: For an Accident news like ‘An accident occurred on E6 road Near Svinesundparken in Østfold County’, extraction is done in the following way



LocationA is geocoded with the help of Google Map API service provided by Google. LocationB is a road. So a fully geometry is required and this cannot be achieved by Google Maps API geocoding service. So for LocationB a SQL query is made on the MySQL database. Then using the haversineFormula [43] the shortest distance is calculated between the coordinate pair's of geocoding and those from query on database. The coordinate pair from the sql query with the shortest distance with the coordinate obtained from geocoding is taken and reverse- geocoded into the map with Google static map. The process is explained in the following Figure 4.15.

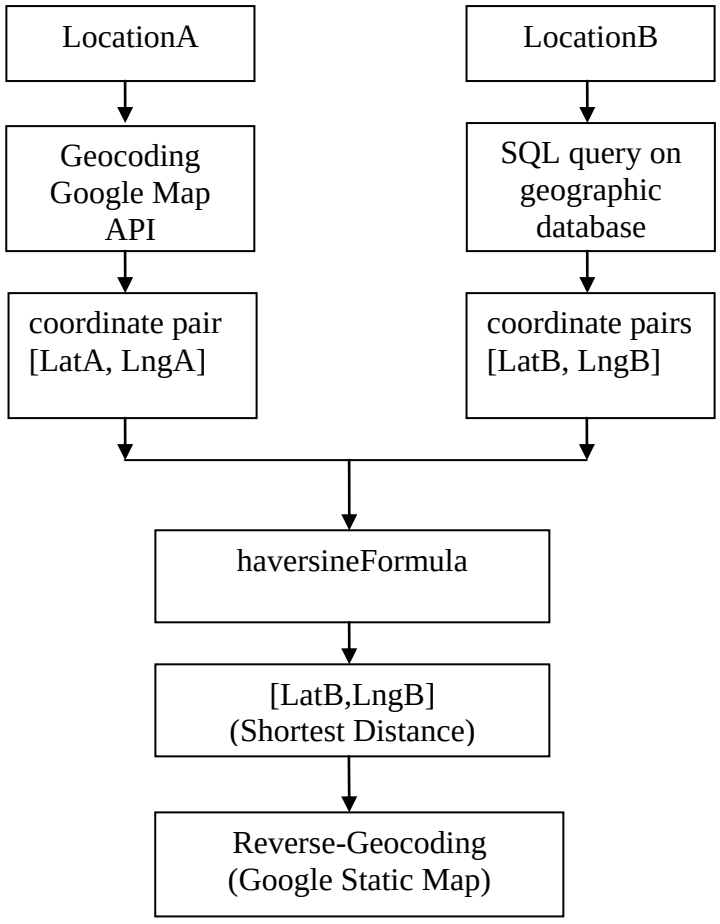


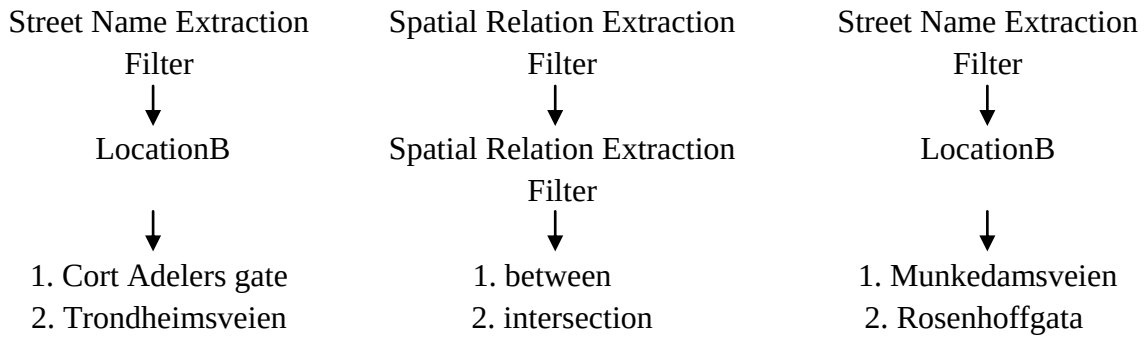
Figure 4.15: Geocoding and Reverse-geocoding process

2. ‘Between / Intersection’ case:

This case is different from the near case as instead of finding the near point task here is to find the intersection of two streets or roads. This case also has a Location and spatial relation to the other location. Locations are extracted from Street Names Extraction Filter while their spatial relation is extracted from the Spatial Relation Extraction Filter as discussed in section 4.1.2. The following examples shows the between and intersection accident events

Example:

- between Cort Adelers gate and Munkedamsveien in Oslo.
- intersection of Trondheimsveien and Rosenhoffgata



In this case Google Map API cannot be used for geocoding as instead of a point coordinate pair; linestring geometry is required in order to find the intersection of the streets. So for both LocationA and LocationB SQL query is made on the database to extract their geometry linestring. Then we find the intersecting coordinate pair. The intersecting points can be more than one but these points are within 2-5 metres. So the first intersection point coordinate is taken which is then reverse- geocoded into the map with Google static map. The process is explained in the following Figure 4.16

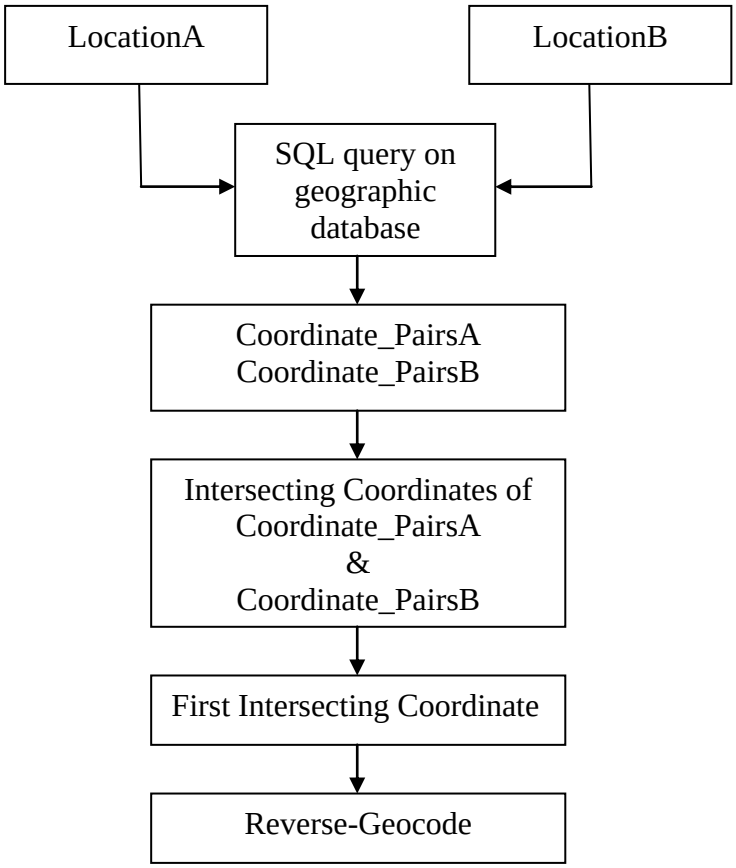


Figure 4.16: Geocoding and Reverse-geocoding process for between/intersection case

3. 'inside tunnel' case:

The main of this case is to locate the accident that took place certain distance inside a tunnel. Many a times the accident event is not described clearly especially when accidents happen inside a tunnel. For instance for an event of type "*A truck caught fire about 2.5 km into the 11.4 km long Gudvanga tunnel*", the location is not clear. Accident location can be 2.5 km from both ends of the tunnel. This case deals with such accidents by pointing out the approximate distance from both the ends. Thus two points are located. When the accident approximation distance is not mentioned in the text, the event location can be pointed in the start or end or middle of tunnel.

Tunnel name is extracted from Tunnel Names Extraction Filter while the approximate distance is extracted from the Tunnel distance Extraction Filter as discussed in section 4.1.2.

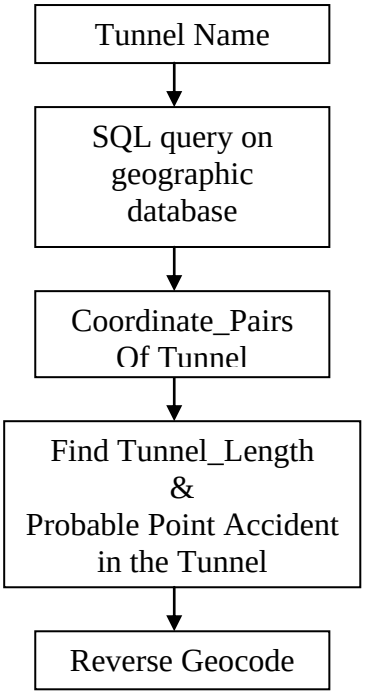


Figure 4.17: Geocoding and Reverse-geocoding process for 'inside tunnel' case

This case sometime gives a near precise location of event while sometimes give approximate results. It entirely depends on the description of the accident event that occurred certain distance inside the tunnel or near the entrance of the tunnel or on the exit. For an event that occurred inside some distance in the tunnel, two results are pointed in map as the accident location can be at certain distance inside the tunnel from both the ends. For the rest of types where accident distance is not specified, the accident location can be pointed at the entrance, at the exit or at the middle of the tunnel.

Chapter 5 Results and Analysis

This chapter discuss the results obtained from the geocoding cases in Chapter 4. AIBN accident reports are used as test case for this thesis. Out of the 40 accidents report published by AIBN, 28 reports has the near case, 4 reports showed intersection/between case and the rest are 'inside the tunnel' case.

5.1 Experimental Set up

The following tools should be installed:

- Stanford-ner-2015-12-09
- Stanford-postagger-full-2015-04-20
- Google Maps Geocoding API
- MySQL 6.3
- Vivid Solutions Geometry

The program is written in Java programming language in Eclipse IDE.

5.2 Result: Geocoding cases.

The extracted information from the Section 4.1.2 is processed my section 4.2.3 to get the result for various geocoding cases.

1. **'near' case results:**

In this case some of the past accident news report are used as test cases which indicates the near case. Some results are shown below.

News Report : 'On 5 May 2014 the trailer of a Swedish-registered heavy goods vehicle loaded with timber overturned when it was entering a four-lane motorway on the E6 road near Svinesundparken in Østfold county.'

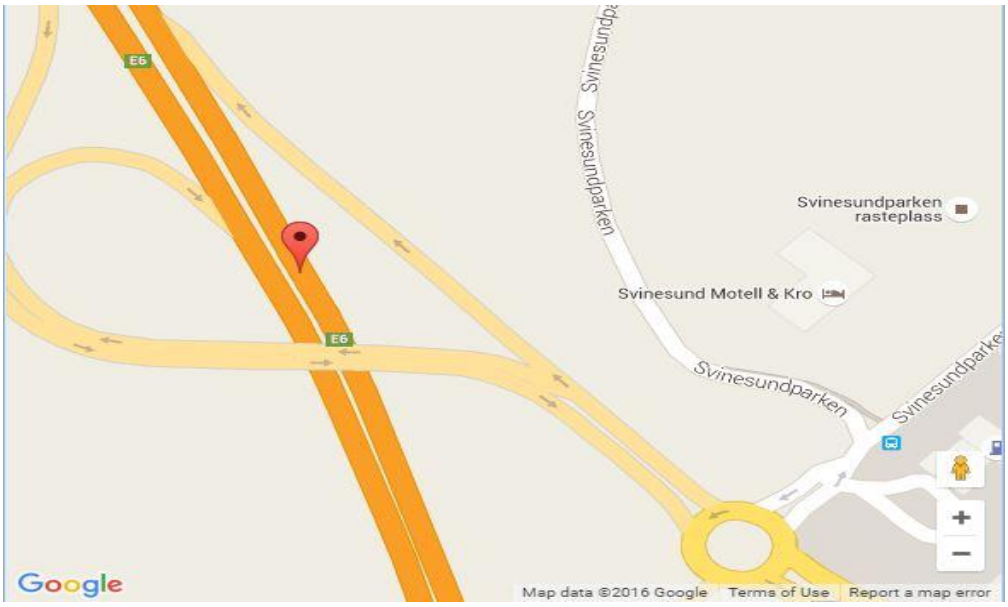


Figure 5.1: near case news report result for event on E6 road near Svinesundparken

News Report: 'On Friday 12 June 2015, a truck hit the median barrier on the E18 road approaching Oslo. The accident took place near Fiskevollbukta.'

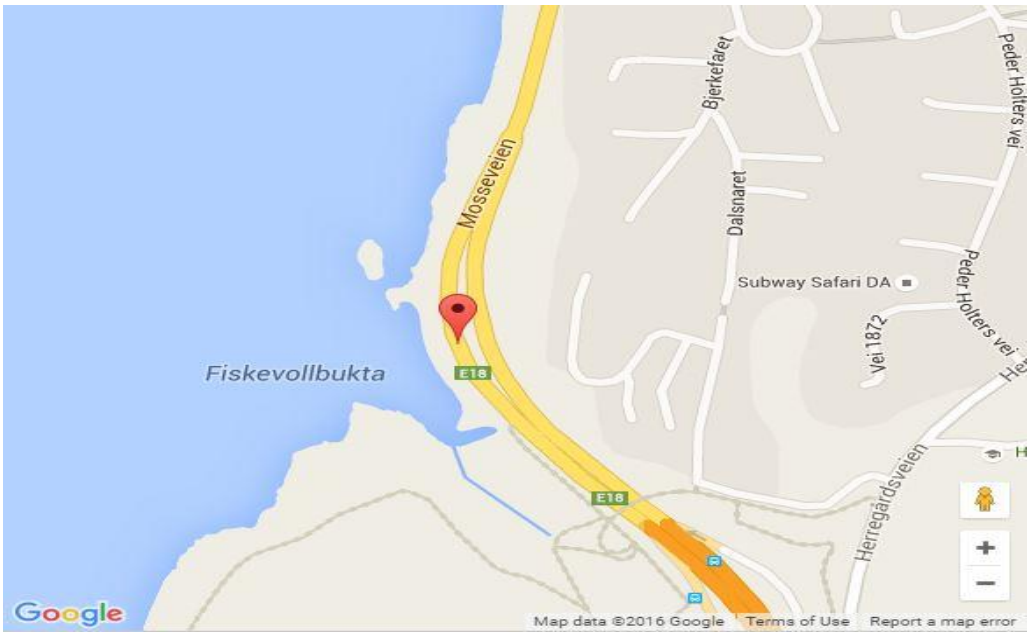


Figure 5.2: near case news report result for event on E18 road near Fiskevollbukta

Similarly all the online accident news reports from AIBN which are publish are extracted manually and processed. The program is able to extract and locate all the news report that has near case it. The following table 5.1 shows the near case results with distance difference in the automatically located events and AIBN located events.

Extracted information for 'near' case Accident Reports				
Location	**Location**	**Automatic Location (Latitude/Longitude)**	**AIBN Location (Approximate) (Latitude/Longitude)**	**Difference in Distance (km)**
Fiskevollbukta	E18	59.8427909 10.7753124	Not Given	---------
Bugøyfjord	E6	69.8681851 29.3457551	69.862469 29.347343	0.64
Svinesundparken	E6	59.1293206 11.2690093	59.129801 11.267899	0.008
Vinjeøra	E39	63.20512 8.9868861	63.201424 8.970995	0.9
Fardal	Rv55	61.191797 7.0232358	Not Given	---------
Dombås	E6	62.0756982 9.1278419	62.072332 9.123917	0.43
Søgne	E39	58.0934852 7.7853272	58.097842 7.808075	1.42
Åsen	E6	63.6099791 11.0498965	63.609687 11.048584	0.07
Veme	Rv7	60.2039427 10.0908416	60.203195 10.102644	0.66
Neverdal	RV3	62.0756982 9.1278419	62.697892 10.136719	86.57
Lenefjorden	E39	58.1151539 7.1939869	58.135502 7.182426	2.36

Table 5:1: difference in distance between automatic located and AIBN located events for near case.

The result shows that the minimum difference in distance is 0.008km and with maximum distance difference being 86.57km. The mean distance difference is less than 1km. The large difference in distance depends both on the preciseness of the news report and the geocoding result. For instance it was reported that accident occurred near Neverdal on Rv3 but the Google Geodocind locates the place Neverdal as at a very distance Rv3 road.

2. 'between/intersection' case results:

In this case the news report that shows the accident events as between or intersection of certain roads or streets is taken for testing. Some results are shown below.

News Report: 'On Tuesday 11 November 2014, a bus in regular service collided with a tram in the junction between Cort Adelers gata and Munkedamsveien in Oslo.'

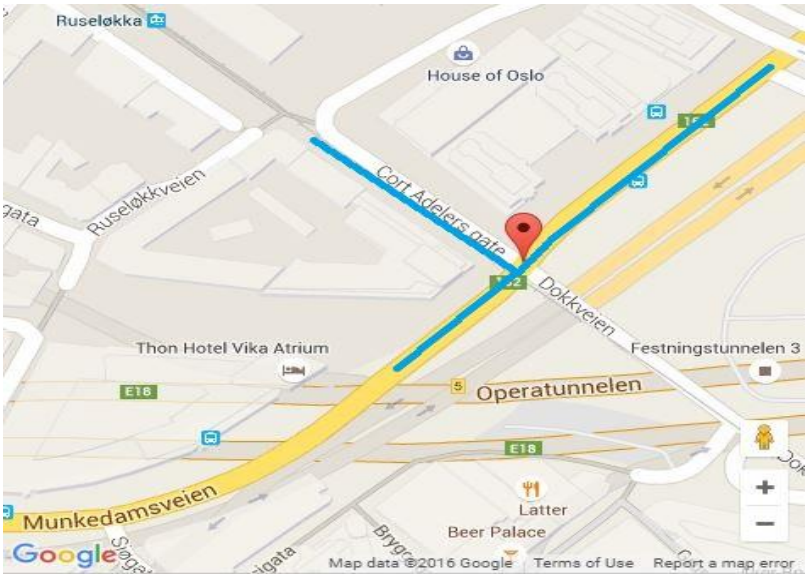


Figure 5.3: between case news report result for event between Cort Adelers gata and

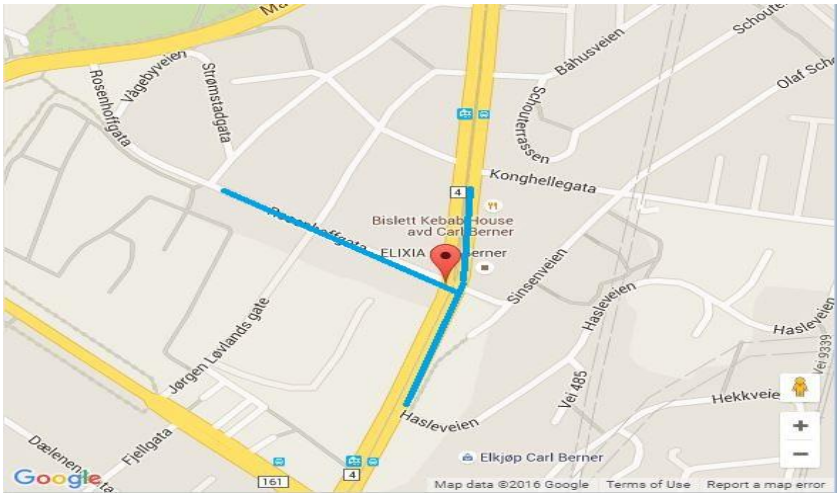


Figure 5.4: between case news report result for event the intersection of Trondheimsveien

News Report: 'Just before 20.00 on 15 December 2013, an articulated bus was driving down the public transport lane into the intersection of Trondheimsveien and Rosenhoffgata where it collided with a van that turned left directly in front of the bus after driving in the parallel lane to the right of the bus.'. Result is shown in Fig 5.4

The following table 5.2 shows the intersection/between case results with distance difference in the automatically located events and AIBN located events.

Extracted information for 'Intersection/between' case Accident Reports				
Location A	**Location B**	**Automatic Location (Latitude/Longitude)**	**AIBN Location (Latitude/Longitude)**	**Difference in Distance (km)**
Trondheimsveien	Rosenhoffgata	59.9285937 10.7777677	59.928606 10.777677	0.001
Cort Adelers gate	Munkedamsveien	59.9118406 10.7265001	59.911804 10.726474	0.005

Table 5:2: difference in distance between automatic located and AIBN located events for intersection case

The result shows that the minimum difference in distance is 0.001km and with maximum distance difference being 0.005km. Thus for this case the distance difference is very less.

3. inside tunnel case results:

In this case the news report that shows the accident events that took place at a certain distance inside a tunnel are tested. Some results are presented below.

News Report: 'On 5 August 2013, an empty Polish-registered heavy goods vehicle caught fire about 8.5 km into the 11.4 km long Gudvangatunnelen. The vehicle caught fire, the smoke from the fire was intense, and a total of 67 persons were forced to evacuate the tunnel.'

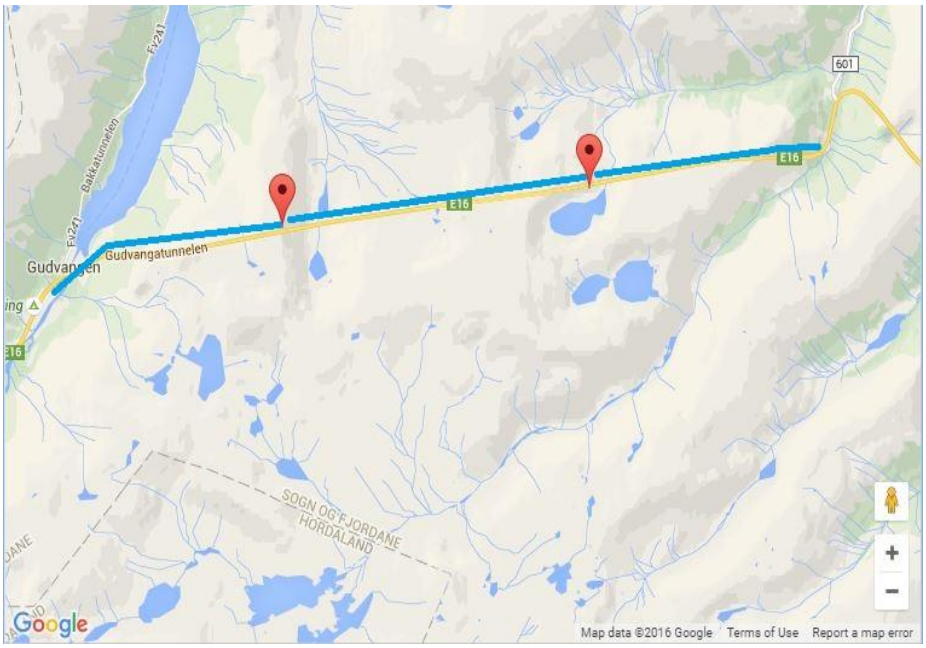


Figure 5.5: certain distance inside a tunnel case news report result.

News Report: ‘On Thursday, 23 June 2011, at 1436 hours, a lorry truck registered in Poland started burning in the 7.3 km long Oslofjord tunnel as a result of an engine breakdown. ‘

For the above example where event location is not specifies, the location can be pointed at the entry, exit or at the middle of the tunnel as shown in Fig 5.6

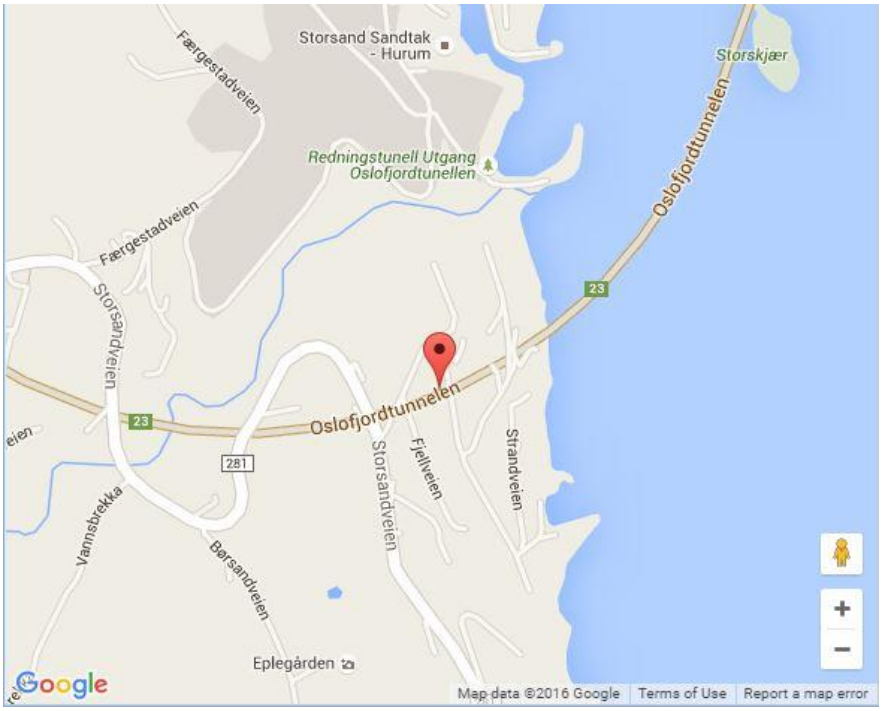


Figure 5.6: certain distance inside a tunnel case news report result for event on Oslofjord

Extracted information for ‘inside tunnel’ case Accident Reports						
Location	**Distance Inside Tunnel**	**Automatic Location (Latitude/Longitude)**		**AIBN Location (Latitude/Longitude)**	**Difference in Distance**	
		Tunnel End	Tunnel Start		End	Start
Gudvangatunnelen	8.5km	60.888216 6.9881384	60.883339 2 6.9007203	60.888599 6.988878	0.05	4.81
Oslofjordtunnelen, Tunnel	--------	59.6539232 10.5999154		59.653580 10.599718	0.04	

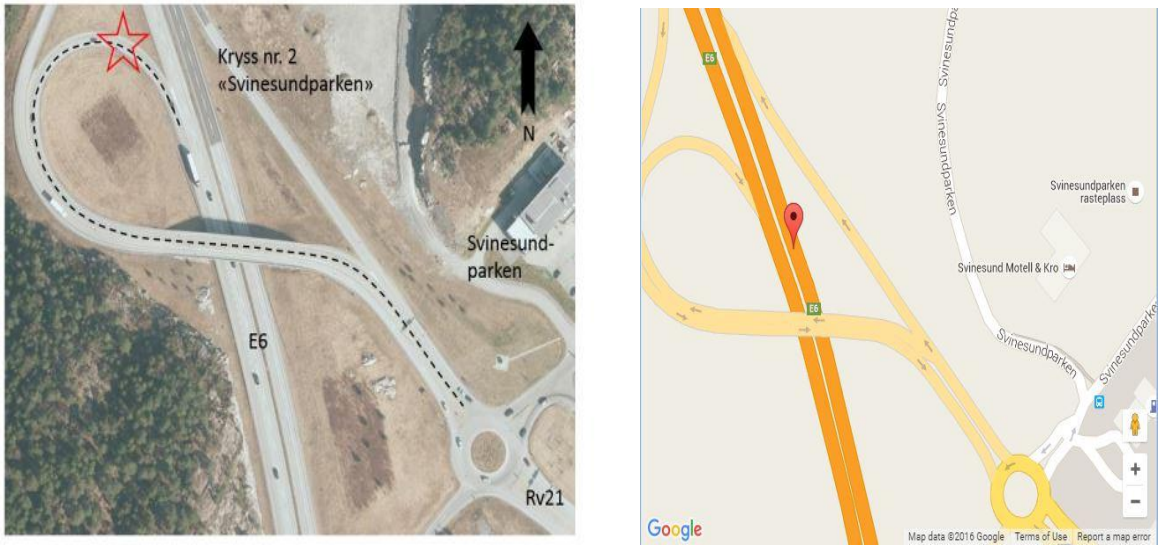
Table 5:3: difference in distance between automatic located and AIBN located events for insidetunnel case

Thus for this case the accident location can be one or two depending on the accident news description. For the news report that describes the accident distance within the tunnel, two locations are pointed in map. And by comparing both the location distances with the actual accident location manually pointed by AIBN, the location which has minimum distance difference can be taken as the most probable location of accident event.

5.3 Result Comparison

In this thesis most of the news report locations are successfully geocoded bases on their spatial relationship. To test the preciseness of the geocoded results, a comparison is done with the manually located accident events by the AIBN report found online.The results are listed in table 5.1, table 5.2, and table 5.3 for all three cases.

- **'near' case :** Results for 'near Svinesundparken on E6 road'

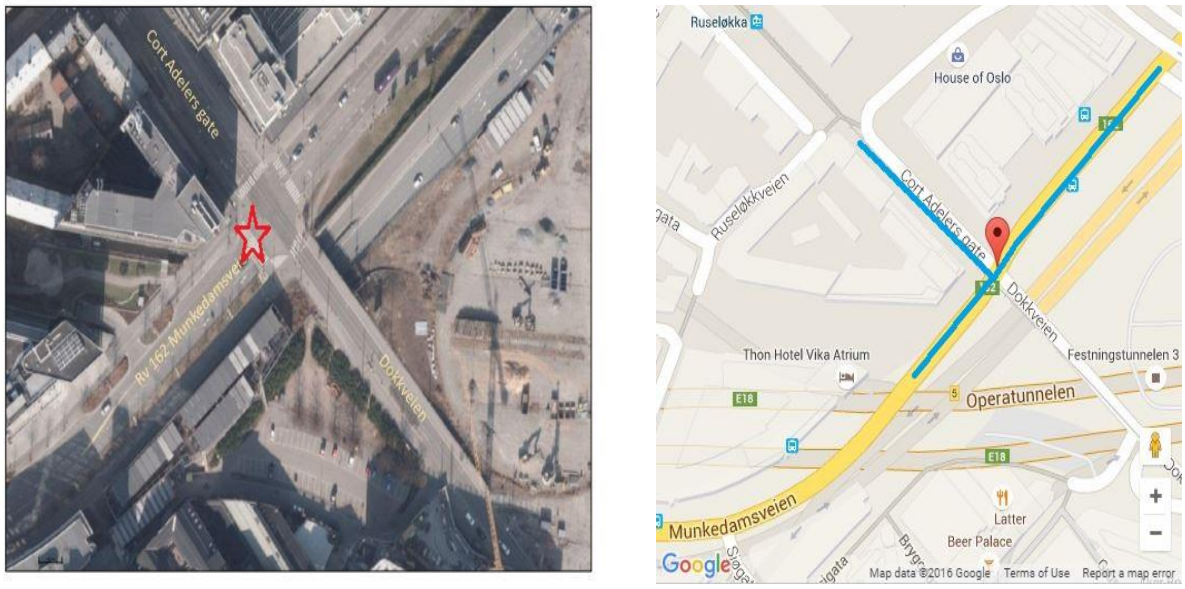


A. **B.**

Figure 5.7: Event Location result comparison. (A) AIBN manually located near result, (B) automatically located near result

Most of the automatically located results are close to the actual accident events locations. From table 5.1, the minimum difference in distance was found to be about 8 meters and the maximum difference in distance was about 87km for some exceptions.

- **'between' case :** Results for 'between Cort Adelers gata and Munkedamsveien'



A. **B.**

Figure 5.8: Event Location result comparison. (A) AIBN manually located between result, (B) automatically located between result

This case showed the most accurate results among all the three cases. Only difference of some metres is found between the manually located and automatically located accident events.

- **'inside tunnel' case:**

Figure 5.9: AIBN manually located accident location

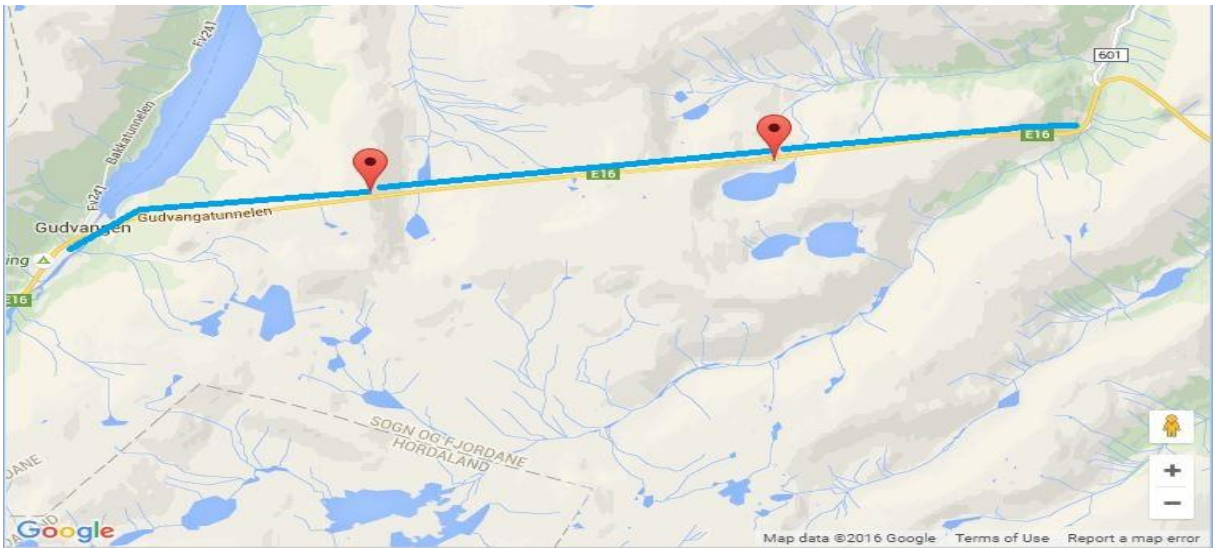


Figure 5.10: Automatically located result.

In this case the program produces two possible locations of accident. When compared with the AIBN accident result it is found that and any one of this can be the actual location of the accident event. In the above case one of the locations is the actual location of accident and the other is the approximated place.

Chapter 6 Conclusions and Discussion

In this thesis information extraction is done on the accident news report with the help of Stanford CoreNLP and various manually designed information extraction filters. The main aim of the thesis was to extract those news reports that that has an accident event described in it and locate the accident event locations in map as precisely as possible.

This study was driven by practical reasons. It is unfeasible to manually locate the accident event location. So, some sort of model is required to extract information and automatically locate the accident locations precisely based on the location description found in the accident news report.

Stanford CoreNLP works very efficiently in bringing out a dependency tree in one sentence and brings out the relation between words within the sentence but it does not provide the same functionality for more than one sentence. Geocoding services available do not provide the functionality to extract the full geometry of the street or road. Although Google Maps API Geocoding service provide road intersection point but it is only limited to major roads.

In this thesis, we presented some techniques to extract information in the text based on the pattern of the text and also with the help of Stanford CoreNLP toolkit. We also designed a database for the purpose of this thesis by extracting relevant information from the Norway road osm shape file. The database is built to carry out the geocoding queries locally. While the place name geocoding is done with Google Maps API. From the database a full geometry of most of the roads and streets in Norway can be accessed in order to precisely locate the accident event places.

Results from the geocoding are compared with the manually located accident events. The results in Chapter 5 show varying difference in distance for different cases between the automatically geocoded location and manually pointed locations. The difference depends on the description of the event location available in the news report and the ability of the geocoding service to bring out the ambiguity in place names. Also geocoding an event on a certain street preciseness depends on the availability of street information in the database. For the ‘near’ case, the difference in distance can vary from 8 meters to 87km. For the between case the geocoded location is almost precise but not exactly similar to the manually pointed event location.

Thus to summarize various factors affects the geocoding service such as the description of the accident events in the news report, ability of geocoder to define ambiguity in place names, ability of the named entity recognizer to tag the place, street names and road names, availability of location, street, road information in the geodatabase, ability of natural

language processor to process the language in use. Both Information extraction and geocoding are non-trieval tasks which require lots of details in formulating the problem.

6.1 Future work:

This thesis only deals with extracting accident information and its location. A possible future work could be to extract the date and time of accident and the cause of the accident. Accident data can be collected to see the accident patterns and changes that took place over a period of time. At present an efficient NLP for Norwegian language is not available. We hope that Stanford CoreNLP or any other NLP in the near future will provide the NLP functionality. This could help in extracting and analysing the information in real-time.

Bibliography

[1] WHO, "Global status report on road safety 2015," 2015.

[2] AIBN, "the AIBN ten years "on the road"- Anniversary Report," 2015.

[3] R. J. Fleischer, "Extraction of key sections from texts using automatic indexing techniques," ed: Google Patents, 1999.

[4] M. Ando, S. Sekine, and S. Ishizaki, "Automatic Extraction of Hyponyms from Japanese Newspapers. Using Lexico-syntactic Patterns," in *LREC*, 2004.

[5] V. Gupta, "Automatic Extraction of Headlines from Punjabi Newspapers," presented at the Proceedings of the First International Conference on Applied Algorithms - Volume 8321, Kolkata, India, 2014.

[6] Y. W. Wanjari, V. D. Mohod, D. B. Gaikwad, and S. N. Deshmukh, "Automatic news extraction system for Indian online news papers," in *Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), 2014 3rd International Conference on*, 2014, pp. 1-6.

[7] C. D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky, "The Stanford CoreNLP Natural Language Processing Toolkit," *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55-60, 2014.

[8] gisgeography, "what is geographic-information-systems," 2016.

[9] ESRI, "What is GIS," 2016.

[10] M. N. DeMers, *GIS for Dummies*: John Wiley & Sons, 2009.

[11] F. Ormeling and M.-J. Kraak, *Cartography: Visualization of Spatial Data*: Guilford Press, 2010.

[12] P. Folger, "Geospatial Information and Geographic Information Systems (GIS): Current Issues and Future Challenges " 2009.

[13] R. R. Larson, "Geographic information retrieval and spatial browsing," *Geographic information systems and libraries: patrons, maps, and spatial information [papers presented at the 1995 Clinic on Library Applications of Data Processing, April 10-12, 1995]*, 1996.

[14] M. Himmelstein, "Local Search: The Internet Is the Yellow Pages," *Computer*, vol. 38, pp. 26-34, 2005.

[15] M.-F. Moens, "Information Extraction: Algorithms and Prospects in a Retrieval Context," vol. 21, 2006.

[16] R. Grishman, "Information Extraction," *IEEE Intelligent Systems*, vol. vol.30, pp. 8-15, 2015.

[17] D. Okanohara, Y. Miyao, Y. Tsuruoka, and J. i. Tsujii, "Improving the scalability of semi-markov conditional random fields for named entity recognition," in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, 2006, pp. 465-472.

[18] R. Bunescu, R. Ge, R. J. Kate, E. M. Marcotte, R. J. Mooney, A. K. Ramani, *et al.*, "Comparative experiments on learning information extractors for proteins and their interactions," *Artificial intelligence in medicine*, vol. 33, pp. 139-155, 2005.

[19] J. i. Kazama, T. Makino, Y. Ohta, and J. i. Tsujii, "Tuning support vector machines for biomedical named entity recognition," presented at the Proceedings of the ACL-02 workshop on Natural language processing in the biomedical domain - Volume 3, Phildadelphia, Pennsylvania, 2002.

[20] G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. Strassel, and R. M. Weischedel, "The Automatic Content Extraction (ACE) Program-Tasks, Data, and Evaluation," in *LREC*, 2004, p. 1.

[21] G. Zhou and J. Su, "Named entity recognition using an HMM-based chunk tagger," presented at the Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Philadelphia, Pennsylvania, 2002.
[22] V. Ng and C. Cardie, "Improving machine learning approaches to coreference resolution," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 2002, pp. 104-111.
[23] W. J. Hutchins, "The Georgetown-IBM experiment demonstrated in January 1954," in *Machine Translation: From Real Users to Research*, ed: Springer, 2004, pp. 102-114.
[24] G. G. Chowdhury, "Natural language processing," *Annual Review of Information Science and Technology*, vol. 37, pp. 51-89, 2003.
[25] M. A. G. Mark Stevenson "Dependency Pattern Models for Information Extraction," *Research on Language and Computation* March 2009.
[26] K. Sudo, S. Sekine, and R. Grishman, "Automatic pattern acquisition for Japanese information extraction," in *Proceedings of the first international conference on Human language technology research*, 2001, pp. 1-7.
[27] R. C. Bunescu and R. J. Mooney, "A shortest path dependency kernel for relation extraction," in *Proceedings of the conference on human language technology and empirical methods in natural language processing*, 2005, pp. 724-731.
[28] S. Bird, "NLTK: the natural language toolkit," in *Proceedings of the COLING/ACL on Interactive presentation sessions*, 2006, pp. 69-72.
[29] T. A. S. Foundation, "Apache OpenNLP " *The Apache Software Foundation*, 2010.
[30] K. S. Jones and J. R. Galliers, *Evaluating natural language processing systems: An analysis and review* vol. 1083: Springer Science & Business Media, 1995.
[31] T. A. S. Foundation, "Apache OpenNLP," *The Apache Software Foundation*, 2010.
[32] Stanford, "Stanford NLP Named Entity Recognition Results."
[33] M.-c. D. M. a. C. D. Manning, "Stanford typed dependencies manual," 2008.
[34] Google, "geocoding," 2016.
[35] G. D. Swift JN, and Wilson JP, "Geocoding Best Practices: Review of Eight Commonly Used Geocoding Systems," 2008.
[36] Earthnewspapers.
[37] AIBN. *Road-Traffic Published-reports*. Available: http://www.aibn.no/Road-Traffic/Published-reports
[38] G. News, 2016.
[39] geofabrik, 2016.
[40] ARCGIS, "Shapefiles."
[41] E. S. R. I. (Esri), "ESRI Shapefile Technical Description," July 1998.
[42] dev.mysql.com, 2016.
[43] O. KOVYRIN, "Geo distance search with MySQL," *Scribd.com*, 2014.
[44] M. T. Scripts, "Calculate distance, bearing and more between Latitude/Longitude points," 2016.

Appendix A

A.1 Haversine Formula

```
function haversineFormula (latA,lonA,latB, lonB)
   Radius = 6372.8; // Radius of Earth in Kms
   dLat = Radians(latB - latA);
   dLon = Radians(lonB - lonA);

   latA = Radians (latA);
   latB = Radians (latB);

   a = sin(dLat / 2) * sin(dLat / 2)+ sin(dLon / 2) * sin(dLon / 2) *
            cos (latA) * Math.cos(latB);

   c = 2 * asin(sqrt(a));
   Distance = Radius * c;

return Distance;
```

Listing A.1: Pseudo code for Haversine to calculate distance between two coordinates[44]

A.2 'near' case pseudo code

```
If (spatialRelation==’near’) {
  LocationA=getLocationRetationLocation index [0]
        //LocationRetationLocation is a string array of
        //type  <Location><spatialRelation><Location>
  LatitudeLongitudePairA=PlaceSearchGoogleAPI.Search(LocationA)
        // LatitudeLongitudePairA contains the Geocoding result from Google Geocoding
        //service
  LocationB=getLocationRetationLocation index [2]
  LatitudeLongitudePairsB =select st_astext(Route) from geodatabase where ref=' LocationB '
        for (All LatitudeLongitudePairsB) {
        shortestDistance =shortest distance between LatitudeLongitudePairA
                                    and LatitudeLongitudePairsB
        getLatitudeLongitudePairB(shortestDistance)
        }
}
reverseGeocode(getLatitudeLongitudePairB)
```

A.3 'between/intersection' case

```
If (spatialRelation== between OR intersection’) {
    LocationA=getLocationRetationLocation index [0]
        //LocationRetationLocation is a string array of
        //type  <Location><spatialRelation><Location>
  LocationB=getLocationRetationLocation index [2]
```

```
  LatitudeLongitudePairsA =select st_astext(Route) from geodatabase where ref=' LocationA'
  LatitudeLongitudePairsB =select st_astext(Route) from geodatabase where ref=' LocationB '
        for (All LatitudeLongitudePairsA && LatitudeLongitudePairsB) {
        IntersectingLatitudeLongitudePairs=Intersection (LatitudeLongitudePairsA&
                                                        LatitudeLongitudePairsA)
        getFirstLatitudeLongitudePairs (IntersectingLatitudeLongitudePairs)
        }
}
reverseGeocode(getFirstLatitudeLongitudePairs)
```

A.4 Shortest Distance between Latitude and Longitude Pairs

```
shortestDistance(latitude, longitude, LatitudeLongitudePairsA) {
        for(All coordinates pairs in LatitudeLongitudePairsA ) {
        Distance= haversineFormula(latitude, longitude, LatitudeA, LongitudeA);
        Minimum (Distance)
        }
Return Minimum (Distance)
}
```

A.5 Intersection between Coordinate Pairs of streets (lineStrings)

```
lineStringPairsIntersection( Location1LineStrings, Location1LineStrings) {
        for (All Location1LineStrings & Location2LineStrings) {
        IntersectingLatitudelongitudePairs=Location1LineString
                                                        Intersection(Location2LineString)
        }
        Return IntersectingLatitudelongitudePairs
}
```

A.6 Google Geocoding

```
PlaceSearchGoogleAPI(Location) {
Give Google Geocoding API Key
GeocoderRequest geocoderRequest
GeocodeResponse geocoderResponse
        If(GeocodeResponse geocoderResponse status is ‘OK’) {
                getLatitudeLongitude(Location)
        }
Return LatitudeLongitude(Location)
}
```

A.4 Stanford POS tags description

CC - Coordinating conjunction
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
NN - Noun, singular or mass
NNS - Noun, plural
NNP - Proper noun, singular
NNPS - Proper noun, plural
PDT - Predeterminer
POS - Possessive ending
PRP - Personal pronoun
PRP$ - Possessive pronoun (prolog version PRP-S)
RB - Adverb
RBR - Adverb, comparative
RBS - Adverb, superlative
RP - Particle
SYM - Symbol
TO - to
UH - Interjection
VB - Verb, base form
VBD - Verb, past tense
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
WP - Wh-pronoun
WP$ - Possessive wh-pronoun (prolog version WP-S)
WRB - Wh-adverb