PERFORMING NATURAL LANGUAGE PROCESSING TO FACILITATE DATA ANALYSIS

A DISSERTATION SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF MASTER OF SCIENCE
IN THE FACULTY OF SCIENCE AND ENGINEERING

2020

Student id: 10473203

Department of Computer Science

Contents

Word Count: 1626

List of Tables

List of Figures

Abstract

PERFORMING NATURAL LANGUAGE PROCESSING TO FACILITATE DATA ANALYSIS
Yefeng Li Candidate
A dissertation submitted to The University of Manchester
for the degree of Master of Science, 2020

The aim of the thesis is to investigate the performance of Gismos and to design and construct a super multi-functional Gismo.

The novel aspects of the new Gismo are described. The abstract should perhaps be about half a page long.

The results of testing, which show the abject failure of the Gismo, are presented.

In the conclusions proposals for rectifying the deficiences are outlined.

Declaration

Copyright

Acknowledgements

I would like to thank...

Chapter 1

Introduction

1.1 Project Description and Motivation

Big Data now drives almost every aspect of society. A successful large-scale analysis often requires the collection of heterogeneous data from various sources [1]. For example, if we want to study the impact of coronaviruses on British society, then we need data from various sources, such as information from hospitals, information from carehouse for elder, information from ins, facebook, twitter, information from financial institutions, ect.

However, these data are stored in different formats, such as CSV, JSON, XML, and also some data stored as text, which is natural language. If these data are not integrated together and have a unified format, then the value of these data is difficult to maximize. On the other hand, information from a single data source is not completely reliable, like the data from the sensors. Due to the complex working environment of sensors, the generated data is very noisy. Because some sensors cannot work normally, there are a lot of abnormal data and missing data [2].

If we combine these potentially erroneous data with information from certain more reliable data sources, the quality of the data will be higher. That is why we need data integration, especially integrating natural language information, which can not only improve the reliability of data but also bring us a more comprehensive view of the situation. Moreover, the unified format of data makes it easier for us to conduct further data analysis.

But there are some challenges of this kind of data integration. In addition to common data heterogeneity issues, firstly, data from different sources of natural language has a certain degree of information redundancy. And then, most of the data is expressed in the free text, which is not structured, which means that during data integration, extracting relevant information from natural language data and transforming it into structured data is essential. In order to address these challenges, information extraction (IE) is crucial, it takes unrestricted text as input and 'summarizes' information according to the specified topic or domain of interest, in other words, it seeks valuable information and encodes these information into a structured form, ideal for the storage of database.

1.2 Project Aim and Objectives

The aim of this project is to benefit traffic data service providers and road users, and implement text classification, information extraction and data integration for the data comes form following information sources to generate comprehensive and valuable data for further data analysis.

- data comes from social media like Twitter as people often post when they are late, encounter traffic jams, or abnormal traffic conditions, which are using natural language.
- data comes from news website like BBC, because these news websites have some pages about traffic conditions, some reporters or citizens will provide traffic news to these news organizations. After verifying the situation, the news websites will release these news soon, which are also using natural language.
- data comes from road-side sensors like inductive loop sensors, Bluetooth sensors, etc.

The ultimate idealized goal of this project is to develop an application that can collect and analyze traffic-related data, answer road users' questions about journey time, and can discover road accidents, warning road users, and transportation agencies to shorten transit time and reduce casualties. Because of the time limitation, the core components are developed first, and the remaining parts can be implemented in the future.

In this report, these core components(data collection, sentence classification, information extraction, data integration) will be introduced, while development methods and evaluation methods will also be presented. the objectives of this project:

- review the literature in data preparation for analysis, with a focus on data integration;
- review the literature in techniques for NLP and identify the most appropriate techniques for incident detection in social media;
- design, implement and test a neural network sentence classification solution.
- design, implement and test a NLP solution for incident detection and information extraction;
- design, implement and test a data integration process that includes the proposed NLP solution;

1.3 Project Scope

To achieve the aim and objectives discussed in previous Section, the scope of this project is identified as follows:

- All natural language data will be obtained from the BBC News website and Twitter.
- Choose the appropriate method to obtain data from the above information sources
- Choose a suitable natural language processing method or library for data preprocessing
- Select or develop appropriate text classification algorithms, and establish evaluation indicators to evaluate algorithm performance
- Research on named entity recognition algorithms and choose appropriate algorithms for implementation
- Research on information extraction algorithms and select suitable solutions for implementation
- Research data integration, investigate existing tools, select or implement data integration algorithms

1.4 Dissertation Structure

The remainder of this dissertation is organized as follows. Chapter 2 shows the background and literature review of this project to illustrate the comprehensive understanding of related research and algorithm about text classification, information extraction, and data integration,.ect. Chapter 3 presents the research methodology and the specific method of project realization. Chapter 4 describes the two use cases of this project to show how the various modules of the project work together. The first use case is to combine sensor data with natural language information using the BBC News website as the data source. And the second use case is to combine sensor data with natural language information using the BBC News website as the data source. Chapter 5 gives the evaluation result of the algorithms used in this project and the performance of two language models in text classification. Future work and conclusion are given in Chapter 6.

Chapter 2

Background and Theory

2.1 Background Overview

Decision-makers in a number of areas, such as transport, urban planning and education, will benefit immensely from the large amount of data available from various data sources, such as roadside traffic sensors, mobile phone tracking [2] and social media. But the data obtained through these approaches are vulnerable to data quality (DQ) issues, such as imprecision, incompleteness and heterogeneity [3] [1]. In recent years, data management techniques for data profiling, cleaning and integration have been uesd to improve the quality of raw data, during preparation for futher analysis or for machine learning. As more and more unstructured data (e.g., text written in natural language) becomes a component to be integrated with other types of data before analysis can be performed, challenges associated with Natural Language Processing (NLP) need to be addressed in this scenario.

Because this project is to use NLP technology to generate data from natural language data source and implement data integration with traffic sensor data, this chapter will investigate the related background, techniques, and methods about data collection, text classification, language model, neural network, named entity recognition, information extraction, data integration.Furthermore, a number of important and representative studies will be reviewed.

2.2 Web Scraping

In order to obtain reliable natural language information, we need to grab traffic-related news information from the news website. Therefore, how to automatically obtain valuable information on the page, especially natural language information, is the first problem to be solved in this project. If a website provides a specific API for query such as Twitter, then obtaining from the API is an excellent method, but for some news websites, using web scraping is a more common method of obtaining information. Web scraping involves web page fetching and extracting. Web scrapers usually grab something out of a website, and use it for some other purpose. For example, web scrapers can find and copy the name and phone number of users or companies and their URLs, and make them to a list.

2.3 Text Classification

After obtaining a piece of news, we need to extract information. If we can't accurately find which sentence needs to be extracted, we can only do information extraction to the entire news. This strategy will significantly reduce the extraction effect. In order to avoid this situation, we need to classify sentences before information extraction to determine which sentences can be used to do further extraction and which sentences are useless to us. And the text classification can also be used to classify twitter data, because even if the keywords are used for searching, the results obtained may not be all valuable. We need to filter the data returned by the Twitter API through the text classification algorithm. There are six kinds of key methods, which are commonly used for text classification-decision trees, pattern(rule)-based classifiers, SVM classifiers, neural network classifiers, bayesian(generative) classifiers, and some other classifiers including nearest neighbor classifiers and genetic algorithm-based classifiers. [4]. This project mainly explores the application of neural networks in text classification, so the background and theory of neural networks will be introduced as follow.

2.3.1 Word Embedding

Glove

Word2Vec

2.3.2 Sentence Vector calculation

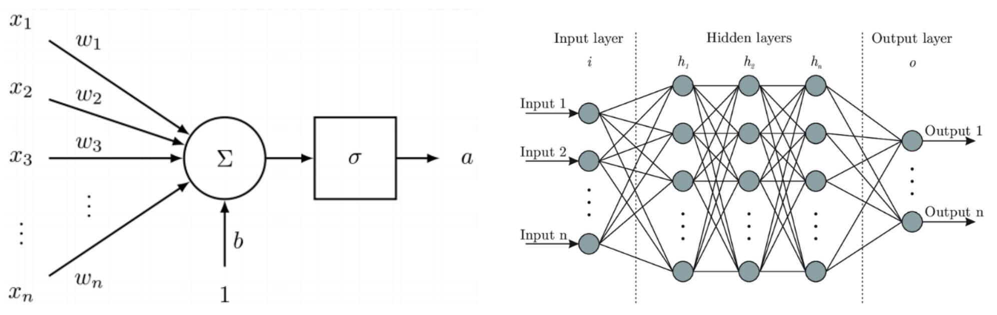
2.3.3 Neural Network

The Theory of Neural Network

Artificial neural networks, commonly simply called neural networks (NNs), are computational models that are partially inspired by the biological neural networks that compose animal brains [5]. An NNA consists of a series of linked units or nodes known as artificial neurons that model the neurons loosely in a biological brain. Much like the synapses in a biological brain, each neuron will send a signal to other neurons.Artificial neuron can process the received signal and send the signal to other neurons connected to it. The signal sent between neurons is a real number, and the output of every neuron is calculated by the sum of inputs and some non-linear function. The connections between each neuron are called edges. Typically, the Neurons and edges have weight and this weight will be adjusted many times in the process of learning. The strength of the signal at a connection are effected by weights. In other words, the greater the weight, the higher the signal strength, the smaller the weight, the weaker the signal strength. Sometimes, neurons have a threshold, only when the aggregate signal in this neuron exceeds the threshold, that signal will be sent out. As shown in Figure 2.1(a), for each neuron, the $x_1...x_n$ are (input) signals, the $w_1...w_n$ are weights. After all the signals are multiplied by the weights, they will be summed with another parameter b. Then the result of the sum will pass the activation functionσ, which is the threshold mentioned earlier, to calculate the output signal a.

The Structure of Neural Network

Neurons are normally aggregated into layers. Each layer may perform its own special transformation on the input. Signals are transmitted from the first layer(input layer) to the last layer(output layer), usually through different layers and multiple traversals. [6] The layers between the input layer and the output layer are called the hidden layers, and there are generally zero to more hidden layers. Multiple link patterns are possible between two layers. They can be connected completely, with each neuron in one layer connecting to each neuron in the next layer. They may also be pooling, where a group of neurons in one layer connect to a single neuron of the next layer, which can reduce the number of neurons in that layer.



(a) The structure of neuron (b) The structure of neural network

Figure 2.1: The structure of neuron and neural network

As shown in Figure 2.1(b), for whole neural network, the first layer is the input layer, the last layer is the output layer, Between the input layer and the output layer is the hidden layer. Depending on the task, the number of neurons in each layer may be different. The number of hidden layers in a neural network can range from zero to more. Empirically, for a wide range of tasks, greater depth does seem to result in better generalisation [7]. This suggests that the use of deep architectures does indeed demonstrate a useful priority over the space of functions that the model learns. But in fact, deeper layers may cause over-fitting problems, and at the same time increase the difficulty of training, making it difficult for the model to converge. Therefore, choosing an appropriate number of hidden layers is crucial for building a good neural network

The Training of Neural Network

Now that we know the theory of neural networks, only the structure is not enough. We need to let the neural network learn from the samples to complete various tasks. This process is called the training of neural network. When a neural network is being trained, all of its thresholds and weights are initially set to random values. The training data is conveyed to the bottom layer(input layer), and it goes through the following layers, multiplying and adding together in complex ways, until it finally reaches the output layer. The thresholds and weights are continuously adjusted during training process, until training data containing the same labels consistently produce similar outputs [8].

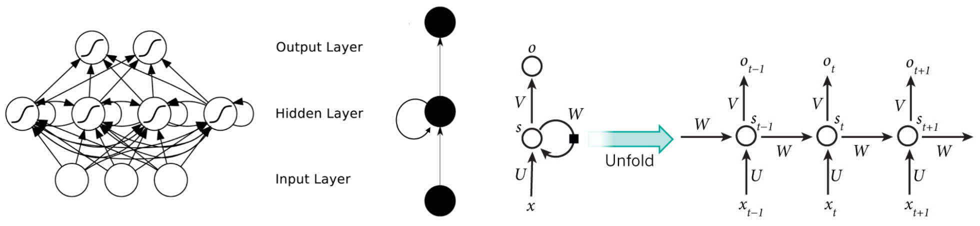
2.3.4 Recurrent Neural Network

The Theory of Recurrent Neural Network

Recurrent neural network (RNN) is a kind of artificial neural networks; the difference between it and ordinary neural network is that the connections between nodes in RNN are a directed graph along a temporal sequence. This enables it to show temporal dynamic behavior.RNNs derived from feedforward neural networks can use their internal state (memory) to process input sequences of variable lengths. [9]. This special structure makes RNN very advantageous when dealing with unsegmented, connected handwriting recognition [10] or speech recognition. [11] [12]

The Structure of Recurrent Neural Network

The recurrent neural network contain input units (input units), the input sets are marked as $x0, x1, ..., xt, xt+1$, and the output sets of output units (output units) are marked as $y0, y1, ..., yt, yt+1$. The recurrent neural network still contain hidden units (hidden units), and we label their output sets as $s0, s1, ..., st, st+1$, these hidden units complete most of the main work. You will find in the figure2.2: there is a one-way flow of information from the input unit to the hidden unit, and at the same time another one-way flow of information from the hidden unit to the output unit. In some cases, the recurrent neural network will break the limitation of the interval and guide information from the output unit to the hidden unit. These are called ”back projections”, and the input of the hidden layer also includes the state of the previous hidden layer. The routines can be self-connected or interconnected.



(a) The structure of recurrent neural network (b) The unfold structure of recurrent neural network

Figure 2.2: The unfold structure of recurrent neural network

2.3.5 Long Short-term Memory

The Theory of Long Short-term Memory

Long Short-Term Memory (LSTM) is an artificial neural recurrent network (RNN) architecture that is used in deep learning. [13] The difference between LSTM and standard neural networks is that the former has feedback connections. This difference makes lstm can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as speech recognition, unsegmented, connected handwriting recognition and anomaly detection in network traffic or IDSs (intrusion detection systems). [14]

The Structure of Long Short-term Memory

As shown in Figure 2.3, in the network structure diagram, each line transmits a vector, which is output from one node and then inputs to another node. The pink circle represents point-by-point operations, such as vector addition; the yellow rectangle represents a neural network layer (that is, a lot of neural nodes); the merged line represents the combination of the vectors carried on the two lines (for example, one vector is h_{t-1}, another is x_t, then the combined output is $[h_{t-1}, x_t]$).

The separated line means that the vector passed on the line is copied and passed to two places. The most critical part of LSTM is the state of the cell (the entire green box is a cell) and the horizontal line on the structure diagram. The transmission of the cell state is like a conveyor belt. The vector passes through the entire cell with only a few linear operations. This structure can easily realize that information can pass through the entire cell without changing it.

If there is only the horizontal line above, there is no way to add or delete information. Lstm is implemented through a structure called gates. The gate can selectively let information through, mainly through a sigmoid neural layer and a point-by-point multiplication operation. Each element of the output of the sigmoid layer (which is a vector) is a real number between 0 and 1, which represents the weight (or proportion) that allows the corresponding information to pass. For example, 0 means ”don’t let any information pass”, and 1 means ”let all information pass”. Each LSTM has three such gate structures to implement protection and control information [15].

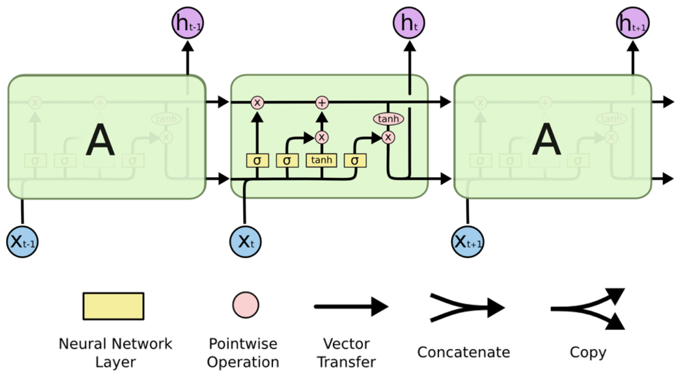
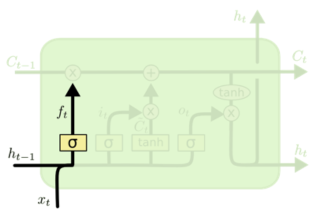


Figure 2.3: The structure of long short-term memory

Forget Gate Layer

As shown in figure 2.4, the first is that LSTM decides to let that information continue to pass through this cell, which is achieved through a sigmoid neural layer called ”forget gate layer”. Its input is h_{t-1} and x_t, and the output is a vector with values between 0 and 1 (the length of the vector is the same as the state of the cell C_{t-1}), which means that the C_{t-1} is the proportion of each part of the information passed. 0 means ”don’t let any information pass”, 1 means ”let all information pass”. Going back to the language model mentioned above, the LSTM model has to predict the next word based on all the above information. In this case, the state of each cell should contain the gender information (reserved information) of the current subject so that the LSTM model can use pronouns correctly. But when the LSTM model starts to describe a new subject, the model should forget the gender of the subject (forget the information) [15].

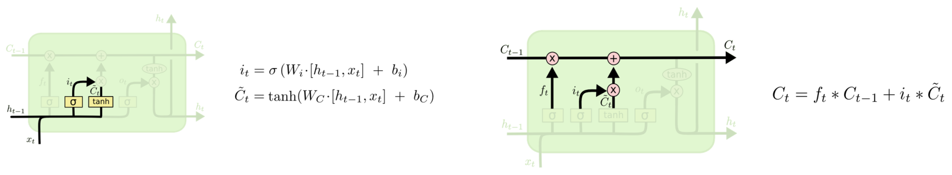


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \;+\; b_f\right)$$

Figure 2.4: The structure of forget gate layer

Input Gate Layer

The next step is to decide how much new information to add to the cell state. As shown in figure 2.5, this requires two steps: First, a sigmoid layer called ”input gate layer” determines which information needs to be updated; a tanh layer generates a vector, $\tilde{C}_t$,which is the alternative content for updating. In the next step, the two parts will be combined to update the state of the cell.


$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$

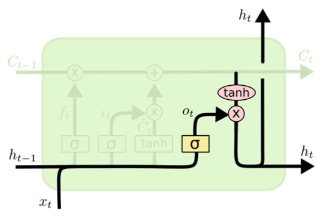
(a) Input gate layer status 1 (b) Input gate layer status 2

Figure 2.5: The structure of input gate layer

In the previous language model example, the LSTM model wants to add new subject gender information to the cell state to replace the old state information. With the above structure, the model can update the cell state, that is, update C_{t-1} to C_t. As shown in figure 2.5, first, the model multiplies the old state C_{t-1} with f_t, this step can let model forget some irrelevant information. The result of previous step will be added with $i_t * \tilde{C}_t$. The final result is the new content we want to add [15].

Output Gate Layer

The lstm model needs to decide what value to output. This output depends on the state C_t of the cell, but not only depends on C_t, it needs to go through a filtering process. As shown in Figure 2.6, first of all, a sigmoid layer will be used to calculate the decision, but need to go through a filtering process. In this filtering process, the LSTM model uses a sigmoid layer (calculated) to determine which part of the information in C_t will be output. After that, a tanh layer (return the values between -1 and 1) will be used to calculate a new value base of C_t. And then, the output of the tanh layer will be multiplied with the weight calculated by the sigmoid layer, so that the final output result is obtained [15].



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 2.6: The structure of output gate layer

2.4 Information Extraction

Since the content of natural language information on the Internet is mostly repetitive, and the information we focus on is limited, how to extract the content we are interested in from a large amount of text information and convert it into a structured form is our main research objective. Jianshu Weng et al. [16] pointed out that twitter, as a data source of social media, has some common problems with its data, for some real accidents, there are usually a lot of related tweets on Twitter that are meaningless nonsense.

2.4.1 The Theory of Information Extraction

To solve this problem, we need to implement information extraction (IE) on natural language data. Information extraction (IE) is the task of extracting structured information automatically from unstructured and/or semi-structured machine-readable documents and from other electronically recorded sources. In most cases, information extraction involves processing natural language text with natural language processing techniques. The ultimate objective of information extraction is to facilitate the computation of previously unstructured data. A specific purpose of information extraction is to allow logical reasoning to draw inferences on the basis of the input data 's logical content. Structured data is semantically well-defined data in a chosen target domain, interpreted with respect to category and context [17].

Charu Virmani et al. [18] analyzed and summarized the current mainstream information extraction algorithms and analysis methods, including Automatic Summarization, Chunking, Parts-of-Speech Tagging, Named Entity Recognition, etc. Claire Cardie [19] introduced the architecture of the information extraction system,the first stage is tokenization and tagging, and then sentence analysis, the third stage is extraction, the last stage is merging and template generation. This entire architecture is clear and has a strong guiding role in the development of this project

2.4.2 Natural Language Data Preprocessing for Information Extraction

Natural language data from the network cannot be directly used for information extraction. These natural language data need to be preprocessed first. The main natural language data preprocessing steps include Tokenisation, Part-of-speech Tagging, Named Entity Recognition, etc. These steps will be introduced in the next section.

Tokenisation

Breaking the raw text into small chunks is the main content of tokenization. Tokenization splits the raw text into words, sentences defined as tokens. These tokens can help to understand the context or to create an NLP model. The tokenization is helpful in interpreting the meaning of the natural language data by analyzing the sequence of the words. For example, the text ”It is a chair” can be tokenized into ”It”, ”is”, ”a”, ”chair”. As the most basic step in natural language processing, there are different methods and libraries available to perform tokenization. NLTK [20], Gensim [21], Keras [22] are the libraries which can be used to accomplish the mission. Tokenisation can be achieved either to separate words or phrases.

Part-of-speech Tagging

Tagging is a sort of classification that can be described as automatic description assignment to the tokens. The descriptor is called tag, which can represent one of the part-of-speech, semantic information and so on. In simple words, POS tagging is a task of labelling each word in a sentence with its appropriate part of speech. Part-of-speech tagging is more complicated than just providing a list of words and their parts of speech, since certain words can represent more than one part of speech at different times, and there are some parts of speech are complex or unspoken. This is not unusual — there are a large percentage of word-forms are ambiguous in natural languages (as opposed to many artificial languages). For instance, even ”dogs”, which is generally considered as just a plural noun, can also be a verb: The sailor dogs the hatch. For that sentence, a correct tagging should reflect that ”dogs” is here used as a verb, not as the more common plural noun [23].

2.5 Named Entity Recognition

Named-entity recognition (NER) (also known as entity identification, entity chunking and entity extraction), refers to the recognition of entities with specific meanings in the text, including names of people, places, organizations, proper nouns, etc. Named entity recognition is an important basic tool in application fields such as information extraction, question answering systems, syntax analysis, machine translation, Semantic Web-oriented metadata annotation, and it occupies an important position in the process of natural language processing technology becoming practical. Generally speaking, the task of named entity recognition is to identify three categories (entity category, time category and number category) and seven categories (person name, organization name, place name, time, date, currency and percentage) in the text to be processed. [24]

2.5.1 Challenges in Named Entity Recognition

Named entity recognition consists of the following two sub-problems: (1)recognition of named entity boundaries; (2) recognition of named entity categories(classes). These problems are usually (but not necessarily) addressed concurrently. As with most language handling issues, there are ambiguities in the language that adds to the challenge of the task. Heavily lexicalized and domain-dependent nature is key challenges to named entity recognition. A large part of a language are names which are constantly evolving in different domains. A robust named entity recognition system for any domain needs labeled corpora and lexicons (e.g., names of monuments). It is expensive to create and update such resources for a number of topics, and it also requires linguistics and domain expertise. There are two popular frameworks of named entity recognition, rule-based and statistical NER. They will be introduced in the following sections [25].

2.5.2 Rule-Based Named Entity Recognition

Early approaches to the recognition of named entities were mainly based on rules. The majority of systems based on rule used three main components : a group of named extraction rules, gaseous (a domain specific lexicon) for the various kinds of named entity classes,an extraction engine that applied the rules and the text lexicons. [25]

Rule-based named entity recognition systems are relatively precise but usually have low coverage and work well on narrow domains. Usually their performance depends on the comprehensiveness of the rules and lexicons. Frameworks such as [26] are still restricted to the seed rules and the lexicon domain. In addition, it is costly to manually integrate deeper information in a rule-based structure beyond surface terms and lexicons. Statistical frameworks, by contrast, are more flexible in incorporating richer linguistic knowledge (e.g., syntax), resulting in more robust systems.

2.5.3 Statistical Named Entity Recognition

Named entity recognition work has concentrated on data-driven and statistical methods due to an growing adoption of NLP statistical methods as well as expanding available data resources. By using statistical methods, the human effort for the tedious construction of rule sets and gazeteers has been reduced. Shortly after their development, the performance of statistical and hybrid systems like [27] [28] is better than the state-of-the-art rule-based systems. There are two components usually used in statistical named entity recognition: (1) labeled training data: text corpora where named entities are annotated (2) a statistical model: a probabilistic representation of the training data. A statistic model consists of parameters that map a language event to a probability. Named entity recognition can be modelled as a classification task for each individual token as a supervised learning problem. Such an approach, however, fails to consider the interdependence between various tokens. In contrast, named entity recognition is usually seen as a structured learning problem for a sequence of variables. This is the view of the sequence labelling where the learner predicts the labels for the whole sequence of tokens (usually as entence). The modelling of the dependency is allowed to exist between tokens under this method [25].

2.6 Accident Detection in Social Media

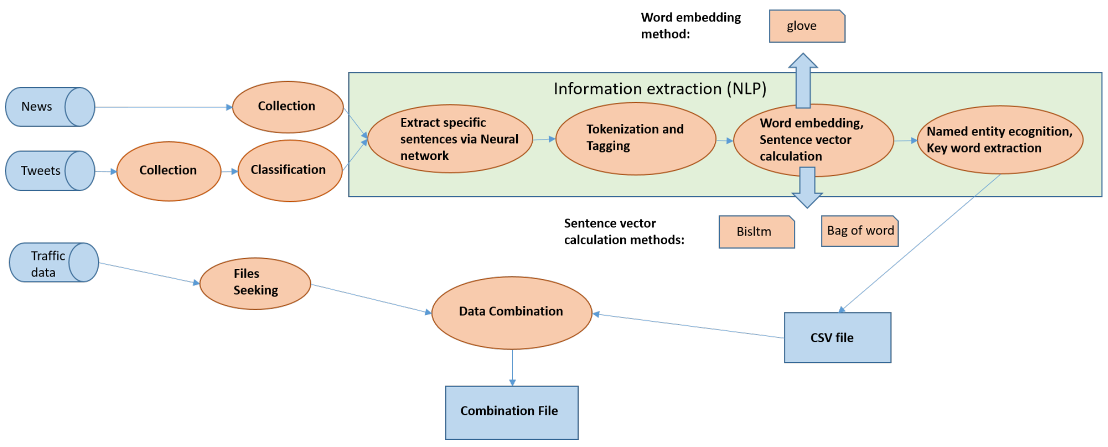
So far, natural language processing technology has been very widely used on social media, and there has been a lot of research focused on accident detection on social media and news websites. Hamed Abdelhaq et al. [29] proposed a system that not only detects accidents in real-time Twitter data streams, but also tracks the evolution of accidents. Fabian Abel et al. [30] presented an framework, Twitcident, which can search, filter, analysis information about events or crises in real world. When incident occurs, this framework can track and filter data from Twitter. Michael Hund et al.developed a methodology for real-time event detection by recognising keywords whose frequency is dramatically higher than expected. Axel Schul et al. [31] uses semantic web technologies combined with machine learning to identify microblogs related to accidents with an accuracy rate of up to 89%. Eleonora D'Andrea [16] presented an analysis of the Twitter stream in real-time traffic event monitoring network. Their programme used SVM to identify tweets as traffic events or not, and achieved an accuracy of 95.75 per cent.

Chapter 3

Design and Implementation

3.1 Chapter Overview

This chapter will introduce the modules developed in this project including data collection, natural language data processing, text classification, named entity recognition, information extraction and other modules, and introduce the exploration of data integration in this project.



Word embedding method:
glove
Information extraction (NLP)
News
Collection
Tweets
Collection
Classification
Extract specific sentences via Neural network
Tokenization and Tagging
Word embedding, Sentence vector calculation
Named entity ecognition, Key word extraction
Sentence vector calculation methods:
Bisltm
Bag of word
Traffic data
Files Seeking
Data Combination
CSV file
Combination File

Figure 3.1: The structure of the project

Taking the BBC News website as the data source as an example, the process of the entire project is shown in Figure 3.1. Natural language data is obtained from the BBC website, and then sentence classification is performed through the neural network to find the sentence to be extracted, and then pick the selected sentence to undergo a series of preprocessing and finally information extraction. Simultaneously, according to the geographic location information extracted from the information, the google map API is used to obtain the latitude and longitude coordinates, and a CSV file is generated. In order to combine the data in the CSV file with the related sensor data, an algorithm is needed to find the corresponding excel files containing sensor data based on the latitude and longitude information in the CSV file, and then use the data combination algorithm to combine the CSV data with the sensor data, generate the final CSV file.

3.2 Tools Used

3.2.1 Programming Language

Python

Python is an high-level, interpreted, general-purpose programming language.This language is created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. The reason for choosing python as the development language is that this project involves machine learning, natural language processing, data processing and other fields. Among many programming languages, Python is commonly used in machine learning projects and artificial intelligence projects with the help of libraries like TensorFlow, Keras, Pytorch and Scikit-learn. As a scripting language with modular architecture, simple syntax and rich text processing tools, Python is often used for natural language processing.[167]

3.2.2 Database

3.2.3 Softwares

3.2.4 APIs

3.3 Data Collection

Chapter 4

Use Case Description

Whenever you wish to refer to books or articles relevant to your report you should use a citation such as [32]. You can also force entries to appear in the bibliography without a citation appearing in the document, by using `\nocite`.

Each document cited must have an entry in a .bib file. For this document we have only one, called refs.bib. These files are listed in the `\bibliography` command at the end of report.tex. Note that the .bib files can (and often do) contain many more entries than are actually cited in a partcular document; the only ones that appear in the bibliography are those that have been referenced using `\cite` or `\nocite`.

In order to generate the appropriate reference entries, you will need to run `bibtex` after `latex` has been run, using the command `bibtex report`. This will generate a file report.bbl, which contains the bibliography entries. Once that file is there, you do not need to run `bibtex` again unless you add new citations, but you will probably have to run `latex` twice after running `bibtex` the first time.

The TEX FAQ ([35]) gives tips on how to cite URLs.

The file refs.bib provides an example of what can be done with BibTEX. You can find much more information in any book on LATEX, for example

Chapter 5

Evaluation

Whenever you wish to refer to books or articles relevant to your report you should use a citation such as [32]. You can also force entries to appear in the bibliography without a citation appearing in the document, by using `\nocite`.

Each document cited must have an entry in a .bib file. For this document we have only one, called refs.bib. These files are listed in the `\bibliography` command at the end of report.tex. Note that the .bib files can (and often do) contain many more entries than are actually cited in a partcular document; the only ones that appear in the bibliography are those that have been referenced using `\cite` or `\nocite`.

In order to generate the appropriate reference entries, you will need to run `bibtex` after `latex` has been run, using the command `bibtex report`. This will generate a file report.bbl, which contains the bibliography entries. Once that file is there, you do not need to run `bibtex` again unless you add new citations, but you will probably have to run `latex` twice after running `bibtex` the first time.

The TEX FAQ ([35]) gives tips on how to cite URLs.

The file refs.bib provides an example of what can be done with BibTEX. You can find much more information in any book on LATEX, for example

Chapter 6

Conclusion

Whenever you wish to refer to books or articles relevant to your report you should use a citation such as [32]. You can also force entries to appear in the bibliography without a citation appearing in the document, by using `\nocite`.

Each document cited must have an entry in a .bib file. For this document we have only one, called refs.bib. These files are listed in the `\bibliography` command at the end of report.tex. Note that the .bib files can (and often do) contain many more entries than are actually cited in a partcular document; the only ones that appear in the bibliography are those that have been referenced using `\cite` or `\nocite`.

In order to generate the appropriate reference entries, you will need to run `bibtex` after `latex` has been run, using the command `bibtex report`. This will generate a file report.bbl, which contains the bibliography entries. Once that file is there, you do not need to run `bibtex` again unless you add new citations, but you will probably have to run `latex` twice after running `bibtex` the first time.

The TEX FAQ ([35]) gives tips on how to cite URLs.

The file refs.bib provides an example of what can be done with BibTEX. You can find much more information in any book on LATEX, for example

Bibliography

[1] H. V. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, and C. Shahabi, “Big data and its technical challenges,” *Communications of the ACM*, vol. 57, no. 7, pp. 86–94, 2014.

[2] J. Lopes, J. Bento, E. Huang, C. Antoniou, and M. Ben-Akiva, “Traffic and mobility data collection for real-time applications,” in *13th International IEEE Conference on Intelligent Transportation Systems*, pp. 216–223, IEEE, 2010.

[3] J. Hutchins, A. Ihler, and P. Smyth, “Probabilistic analysis of a large-scale urban traffic sensor data set,” in *International Workshop on Knowledge Discovery from Sensor Data*, pp. 94–114, Springer, 2008.

[4] C. C. Aggarwal and C. Zhai, “A survey of text classification algorithms,” in *Mining text data*, pp. 163–222, Springer, 2012.

[5] Y.-Y. Chen, Y.-H. Lin, C.-C. Kung, M.-H. Chung, I. Yen, *et al.*, “Design and implementation of cloud analytics-assisted smart power meters considering advanced artificial intelligence as edge analytics in demand-side management for smart homes,” *Sensors*, vol. 19, no. 9, p. 2047, 2019.

[6] Wikipedia, “Artificial neural network.” `https://en.wikipedia.org/wiki/Artificial_neural_network`.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[8] L. Hardesty, “Explained: Neural networks.” `http://news.mit.edu/2017/explained-neural-networks-deep-learning-0414`.

[9] S. Dupond, “A thorough review on the current advance of neural network structures,” *Annual Reviews in Control*, vol. 14, pp. 200–230, 2019.

[10] A. Fernandez, R. B. H. Bunke, and J. Schmiduber, "A novel connectionist system for improved unconstrained handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, 2009.

[11] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," 2014.

[12] X. Li and X. Wu, "Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4520–4524, IEEE, 2015.

[13] S. Hochreiter, "Ja1 4 rgen schmidhuber (1997)."long short-term memory"," *Neural Computation*, vol. 9, no. 8.

[14] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 5, pp. 855–868, 2008.

[15] C. Olah, "Understanding lstm networks." `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[16] E. D'Andrea, P. Ducange, B. Lazzerini, and F. Marcelloni, "Real-time detection of traffic from twitter stream analysis," *IEEE transactions on intelligent transportation systems*, vol. 16, no. 4, pp. 2269–2283, 2015.

[17] Wikipedia, "Information extraction." `https://en.wikipedia.org/wiki/Information_extraction`.

[18] C. Virmani, A. Pillai, and D. Juneja, "Extracting information from social network using nlp," *International Journal of Computational Intelligence Research*, vol. 13, no. 4, pp. 621–630, 2017.

[19] C. Cardie, "Empirical methods in information extraction," *AI magazine*, vol. 18, no. 4, pp. 65–65, 1997.

[20] U. of Pennsylvania, "Natural language toolkit." `https://www.nltk.org/`.

[21] R. Řehůřek, "gensim." `https://radimrehurek.com/gensim/`.

[22] F. Chollet, "Keras." https://keras.io/.

[23] Wikipedia, "Part-of-speech tagging." https://en.wikipedia.org/wiki/Part-of-speech_tagging#:~:text=In%20corpus%20linguistics%2C%20part%2Dof,its%20definition%20and%20its%20context.

[24] Wikipedia, "Understanding lstm networks." https://en.wikipedia.org/wiki/Named-entity_recognition.

[25] I. Zitouni, *Natural language processing of semitic languages*. Springer, 2014.

[26] E. Riloff and W. Phillips, "An introduction to the sundance and autoslog systems," tech. rep., Technical Report UUCS-04-015, School of Computing, University of Utah, 2004.

[27] A. Mikheev, M. Moens, and C. Grover, "Named entity recognition without gazetteers," in *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, 1999.

[28] S. Miller, M. Crystal, H. Fox, L. Ramshaw, R. Schwartz, R. Stone, R. Weischedel, *et al.*, "Bbn: Description of the sift system as used for muc-7," in *Seventh Message Understanding Conference (MUC-7): Proceedings of a Conference Held in Fairfax, Virginia, April 29-May 1, 1998*, 1998.

[29] H. Abdelhaq, C. Sengstock, and M. Gertz, "Eventweet: Online localized event detection from twitter," *Proceedings of the VLDB Endowment*, vol. 6, no. 12, pp. 1326–1329, 2013.

[30] F. Abel, C. Hauff, G.-J. Houben, R. Stronkman, and K. Tao, "Twitcident: fighting fire with information from social web streams," in *Proceedings of the 21st International Conference on World Wide Web*, pp. 305–308, 2012.

[31] A. Schulz, P. Ristoski, and H. Paulheim, "I see a car crash: Real-time detection of small scale incidents in microblogs," in *Extended semantic web conference*, pp. 22–33, Springer, 2013.

[32] L. Lamport, *LaTeX – A document preparation system*. Addison-Wesley, 2nd ed., 1994.

[33] R. D. Boyle and R. C. Thomas, *Computer Vision - A First Course*. Blackwell Scientific Publications, 1988.

[34] W. Dyke, “The nutter’s guide to LaTeX,” *Read Only*, May 1994.

[35] LaTeX. Team, “URLs in BibTeX bibliographies.” `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=citeURL`, 2012.

Appendix A

Example of operation

An appendix is just like any other chapter, except that it comes after the appendix command in the master file.

One use of an appendix is to include an example of input to the system and the corresponding output.

One way to do this is to include, unformatted, an existing input file. You can do this using `\verbatiminput`. In this appendix we include a copy of the C file hello.c and its output file hello.out. If you use this facility you should make sure that the file which you input does not contain `TAB` characters, since LaTeX treats each `TAB` as a single space; you can use the Unix command `expand` (see manual page) to expand tabs into the appropriate number of spaces.

A.1 Example input and output

A.1.1 Input

(Actually, this isn't input, it's the source code, but it will do as an example)

```
/* Hello world program */

#include <stdio.h>

int main(void)
{
    printf("Hello World!\n") ;
    return 0 ;
}
```

A.1.2 Output

```
Hello World!
```

A.1.3 Another way to include code

You can also use the capabilities of the `listings` package to include sections of code, it does some keyword highlighting.

```
/* Hello world program */

#include <stdio.h>

int main(void)
{
    printf("Hello World!\n") ;
    return 0 ;
}
```