Institut für Formale Methoden der Informatik
Abteilung Formale Konzepte
Universität Stuttgart
Universitätsstrasse 38
D-70569 Stuttgart

Institut für Maschinelle Sprachverarbeitung
Abteilung Theoretische Computerlinguistik
Universität Stuttgart
Pfaffenwaldring 5b
D-70569 Stuttgart

Master's thesis Nr. 3413

Information Extraction from Social Media for Route Planning

Mirna Megally

Course of Study: Infotech

Examiner: Prof. Dr. Stefan Funke

Supervisor: Prof. Dr. Hinrich Schütze
Dipl.-Inf. Wiltrud Kessler

Commenced: May 16, 2012

Completed: November 15, 2012

CR-Classification: I.2.7

Acknowledgment

I would like to express my gratitude to all those who gave me the possibility to complete this thesis.

Thank you God for always giving me strength, guiding my steps and blessing me with your grace.

I am really thankful to my supervisor Prof. Dr. Hinrich Schütze, at IMS, for his continuous guidance and help. Additionally, I am deeply thankful to my supervisor Prof. Dr. Stefan Funke, at FMI, for his invaluable suggestions and encouragement.

I would like to gratefully acknowledge the supervision of Dipl. Inf. Wiltrud Kessler, who has been abundantly helpful and has assisted me in numerous ways. The discussions I had with her were invaluable. I would like to thank Dipl. Eng. Andre Blessing for putting me on the right track at the beginning of my thesis. I would like also to thank Olga Podushko for helping me with some annotation tasks.

I would like to especially thank my friends, most importantly dearest Christine, Michael, Mina, Youssef, Hallawa, Amr and all residents of WG 69 for being always by my side during the two years I stayed in Stuttgart and for their continued love and support.

I am deeply grateful for my mother, father and my sister Mirette whose patient love enabled me to complete this work.

My special thanks to my fiance Kareem for his endless love and encouragement in all the time of research and writing this thesis.

On a different note, many people have been a part of my undergraduate education and made me reach this point, I am highly grateful to all of them.

Abstract

Micro-blogging is an emerging form of communication and became very popular in recent years. Micro-blogging services allow users to publish updates as short text messages that are broadcast to the followers of users in real-time. Twitter is currently the most popular micro-blogging service. It is a rich and real-time information source and a good way to discover interesting content or to follow recent developments. Additionally, the updates published on Twitter public timeline can be retrieved through their API. A significant amount of traffic information exists on Twitter platform. Twitter users tweet when they are in traffic about accidents, road closures or road construction. With this in mind, this paper presents a system that extracts traffic information from Twitter to be used in route planning. Route planning is of increasing importance as societies try to reduce their energy consumption. Furthermore, route planning is concerned with two types of constraints: stable, such as distance between two points and temporary such as weather conditions, traffic jams or road construction. Our system attempt to extract these temporary constraints from Twitter. We train Naive bayes, Maxent and SVM classifiers to filter non relevant traffic. We then apply NER on traffic tweets to extract locations, highways and directions. These extracted locations are then geocoded and used in route planning to avoid routes with traffic jams.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Social media, such as Twitter[1] and Facebook[2], are the latest trend in Web 2.0 world. Twitter is a microblogging service. It celebrated its sixth birthday in March 2012 and announced the number of active users to be 140 million users with a daily rate of 340 million tweets[3]. Twitter is a popular network for broadcasting news, staying in touch with friends, and exchanging opinions through 140-characters update status messages called *Tweets*. In recent events such as the Egyptian Revolution, the death of Michael Jackson and terrorist attacks in Mumbai, Twitter was faster in providing news than mainstream media. Therefore, for any possible topic, Twitter became a medium for informal discussion, exchange of ideas or following recent developments. Many interfaces can be used to tweet such as web interface, mobile interface or IM clients.

Furthermore, it is a great medium for development and data mining tasks that can be achieved through the Twitter API[4]. Tweets that are published on Twitter public timeline can be retrieved using the API.

Since researchers are attracted to Twitter, various researches take place in this area. For example, Sitaram and Huberman [1] used Twitter to predict the box-office revenues for movies and implemented a system with an accuracy of 98%. Diakopoulos and Shamma [6] analyzed tweets that were broadcast during the US presidential debate in 2008. Twitter

[1]http://www.twitter.com
[2]http://www.facebook.com
[3]http://mashable.com/2012/03/21/twitter-has-140-million-users/
[4]http://dev.twitter.com

users tended to favor Obama over McCain and Obama was the winner. Twitter can then be used to capture the opinions of the audience and forecast the presidential results.

According to our findings and as discussed in Wanichayapong et al.[42], a significant amount of traffic information exists on Twitter platform. Users on the road are interested in traffic events eg. road closures, accidents and traffic congestion. With this in mind, we address in this thesis the idea of building a system that extracts information from Twitter to be used in route planning. Route planning is of increasing importance as societies try to reduce their energy consumption. At the same time, businesses need fast and reliable transport as economic activity is becoming ever more integrated and interdependent. Many constraints in route planning are stable over time: the distance in kilometers between two cities, the steepest gradient on a railway line or the maximum load for a bridge. However, there are also many types of constraints that are temporary: weather conditions, traffic jams or road construction. We discuss the possibility to extract information about these temporary constraints from user tweets and integrate them into a route planning system.

1.2 Objective of this thesis

In this thesis, we develop a system that extracts temporary traffic information from Tweets to be integrated with route planning. The system works in an online manner and processes each tweet once it arrives. We create and evaluate a supervised machine learning classifier to classify tweets into either tweets related to traffic or tweets not related to traffic. From traffic related tweets, the location and the nature of the incident (road closed or open) are extracted. For this purpose, any tweet not related to traffic is discarded from any further processing. We train the Stanford NER system based on Conditional Random Fields with unstructured traffic tweets. All the locations of interest need to be recognized and plotted on a map. This can be achieved through the use of a Geocoding API that receives requests containing the address to geocode and returns responses containing the geographical coordinates of each location. The coordinates are used to locate the places where traffic incident occurs and avoid them when providing a route.

The research of Java et al. [15] shows that Twitter is most popular in US, Europe and Asia. San Francisco, New York and Tokyo are the major cities where Twitter is highly used. For this reason, we choose the area of San Francisco to analyze its traffic related tweets. We train and test our system on a dataset created from traffic tweets in San

Francisco bay area to achieve high accuracy and precision and recall better or equal to the precision and recall achieved by the classifiers of the state of the art.

1.3 Structure of this thesis

The thesis is structured as follows. Chapter 2 highlights previous research about Twitter and the state of the art in text filtering, text classification and Named Entity Recognition for Twitter and User Generated content. Chapter 3 presents the system architecture with its different modules including: tweets retrieval module, tweet classification module, NER module, geocoding module and visualization and routing module. Chapter 4 explains Twitter and the different APIs offered by Twitter to access tweets such as REST API, Search API and Streaming API. For text classification, we used three different approaches that are highlighted in Chapter 5: Naive Bayes, Maximum Entropy and Support Vector Machines. Text classification is used in two ways: Each incoming tweet is classified into either *traffic* or *junk*, furthermore, each *traffic* tweet is categorized into either *positive*, i.e. mentioning positive traffic information such as clear lanes or end of road construction; or *negative* tweet mentioning an incident occurring at a particular place. Chapter 6 deals with Named-entity recognition and Information extraction. Chapter 7 explains our data structure used to represent the maps. It also deals with the use of Dijkstra algorithm in route planning as well as our weight function. It concludes by describing the Geocoding API, which is used to extract the geographical coordinates of the locations of interest to plot it on a map. Chapter 8 is a detailed evaluation chapter showing the experiments and evaluation results of every module. Finally, Chapter 9 summarizes the project and gives an outlook for future work.

Chapter 2

State of the Art

In this chapter, we give an overview of Twitter related work in the field of information extraction and filtering. In the past years, Social Media and Micro-blogging services became very popular research topics. Twitter became especially appealing for researchers because unlike most social networks, twitter messages are publicly available and can be retrieved in real-time using the Twitter API.

2.1 The Use of Twitter

Java et al. [15] present a study about the topological and geographical structure of Twitter's social network and analyze the intentions of the users and why they tweet. Their analysis proved that the tweets are mostly about daily chatter, sharing opinions, broadcasting news and events. It was found that people are more active on micro-blogging services for two reasons. First, it's a faster mode of communication. Second, the frequency of posting updates is higher. A user is considered active if he posts at least once during a week. Furthermore, people can interact with Twitter through web interface, mobile interface or IM clients. Their study shows that Twitter is most popular in US, Europe and Asia. San Francisco, New York and Tokyo are the major cities where Twitter is highly used. A manual approach was adopted to determine the user intention from every tweet and their analysis shows that the largest percentage of Twitter users use it as daily chatter, 21% of the users use it for conversations and 13% of the users use it for sharing information and reporting news.

Morris et al. [25] ran a user study to investigate the use of social networks and status messages to ask questions and the reasons behind this phenomenon. Their research proposes novel ideas for a new type of search engine integrated with social networks. 624 people participated in the survey. The survey included questions about what types of questions they post as status messages and they are asked to provide examples, whether they answer questions posted by other people and the reasons behind the use of status messages to ask questions. The findings reflect that the type of questions asked are recommendation and opinion questions. The popular topics are shopping, technology and entertainment. The survey proved that people usually answer questions because they have experience about the topic in question, they are in close relationship to people asking the questions, they have free time or they might earn social capital. People sometimes prefer to ask on social networks rather than search engine depending on the type of information they need as well as the level of trust and personalization.

Jansen et al. [14] address three research question: whether Microblogging can be used as a form of electronic word-of-mouth for sharing customer opinions concerning brands, the characteristics of brand microblogging and the patterns of microblogging communications between companies and customers. Their study shows that microblogging service can impact customer relationship management and promote brand image and brand awareness. It is apparent that microblogging services such as Twitter could become key applications in the attention economy. Their data sets consisted of more than 150,000 tweets containing comments about brands, sentiments and opinions. Clustering techniques were used to classify the microblogs into positive and negative as well as by brand name. Furthermore, there was a comparison between automated and manual techniques in classifying sentiments. The research findings show that 19% of microblogs contain mention of a brand. Of the branding microblogs, nearly 20% contained some expression of brand sentiments. Of these, more than 50% were positive and 33% were critical of the company or product.

Zhao et al.[44] is a similar study, an exploratory research was held on how and why people use Twitter and its impact in informal communication at work. They suggested that microblogging may play an important role for collaborative work and organizational innovation. They have organized the benefits of informal communication into relational benefits and personal benefits. The relational consequences consist of the two persons' relationship and their future common activities eg. Personal perception, Common ground and Connectedness. The personal consequences is shown in the effect that informal communication may have for one's personal interests and goals. They proved that Twitter among employees in organizations play an important role in organization's success and provide a variety of benefits supporting for collaborative work.

2.2 Twitter User Classification

Java et al. [15] classified the Twitter users into three categories: Information Source, Friends, Information seeker.

Naaman et al.[27] assigned the expression "Social Awareness Streams" (SAS) to the new types of social media and microblogging service. They study the properties and the behavior of the users in such networks. For their case study, they choose Twitter platform to answer the research questions. According to their research, SAS are characterized by three main factors that differentiate them over other communication platforms: the publicity of the posted messages by different users, the limited length of the messages and the connected network where the messages spread. The aim of this research is to categorize tweets based on the content and to study how the content of the message differs depending on the user information and the tools or devices used to post messages. Tweets are streamed from the public timeline and stored in a database with all the metadata including user ID, user profile, timestamp of the tweet and the tweet content. Some criteria such as number of followers and number of friends have been imposed to select a number of users for the analysis. 911 users have been selected out of 125,593 unique users. A set of 3379 messages tweeted by the selected users have been manually annotated by the paper authors to fall under one or more of nine categories: IS (Information Sharing), SP (Self Promotion), OC (Opinions/Complaints), RT (Statements and Random Thoughts), ME (It's all about me), QF (Questions to followers), PM (Presence maintenance), AM (Anecdote - me), AO (Anecdote - other). The results show that more than 40% of the Tweets are categorized as ME followed by RT, OC and IS with approximately 20% each. In other words, this means that 20% of the Tweets have a news character, while 80% can be characterized as user-to-user communication.

Krishnamurthy et al. [16] presents a deep analysis of Twitter. They study the characterization of Twitter users, the characterization of status updates and some other Twitter properties. Data are collected using two techniques provided by the Twitter API. The first data set was obtained by selecting a sample of users based on some criteria and collect information about the users they were *following.* The size of the first set was 67527 users. The second data set is collected from Twitter public timeline which shows the status of the most recent active users and resulted in 35978 users. With the help of the collected data, they analyze and categorize the users into three categories: *broadcasters, acquaintances* and *miscreants. Broadcasters* are users having number of followers more than the number of people they follow. It was proved that a high percentage of this category are media stations. *Acquaintances* are people having almost equal number of *followers* and *following.* The third category, *miscreants* are people having a higher number of *following* compared to *followers.* The users falling into this category might be spammers, or users who are interested in stalking the people on their following list. These three categories can also be used to give an impression about the number of tweets posted by each user with broadcasters having the highest rate and miscreants the lowest. Further, an analysis of the status updates proved that about 60% are posted from the web, 5% come from different third-party applications using Twitter API and the rest from mobile devices.

Huberman et al. [12] studies the interaction between users in social networks particularly in Twitter. In order to run this research, a list of 309,740 users who posted around 255 posts, have on average 85 followers and follows about 80 users were selected as a data set. They define active users as the users who post at least twice. They were 211,204 users. They also found that 25.4% of the tweets are directed to particular users who are *friends* with the *twitterer.* The number of *friends* is proportional to the level of user activity rather than the number of followers. However, only 0.2% of the users have greater number of friends than followees (people followed by a user). They conclude that the social network resulting from the follower-following relationship is very dense and in reality, the interaction level on Twitter is due to the more sparse network relying on the friends relationships.

2.3 Text Classification in Twitter

Wanichayapong et al. [42] found that there exists a significant number of traffic information such as traffic congestion and incidents in Twitter. Traffic information was extracted from Twitter using syntactic analysis and then classified in two categories: point and link. Point is related only to one point on a road while link information associates with a road start and end point. This method can classify point with an accuracy of 76.85% and link with an accuracy 93.23%. The study was held in Thailand. 1,191,760 tweets were collected for training and testing. People tweet in real-time to share information about traffic events e.g. road closures, traffic congestion, and accidents. The aim of the research was to extract events related to traffic and retweet them to make a large audience capture this information. The purpose of syntactic analysis is to determine the structure of the input text. This structure consists of a hierarchy of phrases. A tweet is considered informative only if it can provides the "what" and "where". The twitter streaming API was used to extract tweets related to certain queries such as : traffic congestion, accident. The tweets are then tokenized using Lexto (dictionary based tokenizer with lexical analysis for Thai language). Their study lead to filtering results of 91.75% accuracy, 91.39% Precision and 87.53% recall.

Sankaranarayana et al. [33] propose a system called *TwitterStand* that leverages Twitter to capture any tweet mentioning breaking news. This news processing system is a distributed system, that uses filtering techniques to get rid of noise, classifies the tweets into clusters depending on the content in real time. Tweets capture the moment and not being able to process the information in an online manner can make the system deviate from the main purpose. Geographic location criteria is used in clustering the tweets since this information can be obtained from the tweets. Geotagging information in tweets is extracted from either the meatdata attached to the tweet when crawled from Twitter API or by extracting information from Tweet content. A large number of users use their mobile devices to tweet and have location sensors enabled. A score is assigned to each news topic depending on the number of users who interact with it. This information can be crawled from Twitter using the API. The first step in building the system is to decide whenever tweet comes in whether it is news or not. The system adopts a naive bayes classifier. A preprocessing step is to required to improve the quality of the input to be able to apply the different NLP tasks. A manual identification of 2000 users or *Seeders* frequently posting about news has been performed because previous research has proved that 10% of the users account for more than 90% of the tweets. Their research proves the feasibility of building a news system based solely on Twitter data.

Phelan et al. [31] leverage Twitter and RSS feeds to build a news recommendation system called Buzzer in a user-personalized manner. Existing recommendation systems require a background containing information about the user to correctly function. Information retrieval from tweets can be used in ranking RSS feeds according to user preferences. The system constructs a term vector of both the RSS list and the user's Tweets, and uses the TF-IDF weighting scheme to locate the RSS feeds which correspond the most with the Tweets. Two ranking strategies are implemented and evaluated: Public rank and Friends rank. Public rank relies on tweets pulled from the public timeline while Friends Rank relies on tweets mined from user's Twitter friends. 67% of the users preferred the recommendation based on Public Rank scheme , 22% preferred Friends rank and 11% did not prefer any strategy.

Another recommendation system called zerozero88 is proposed by Chen et al. [3]. Three designed algorithms are implemented: 'Candidate URL', 'topic relevance' and 'social voting process'. They decided to use social information streams such as Twitter for their recommendation systems. This is due to the fact that any content on Twitter is recent, there exist a communication and interaction between the users and the users are active members who are involved in generating this content. Candidate URL algorithm finds the most promising set of URLs than can be used in the recommendation system. Since tweets come at a furious rate, a certain criteria is needed to filter out tweets and hence URLs non relevant. First, a social neighborhood criteria is adopted which only considers URLs posted by a user's followees and followees of followees. Second, most popular URLs can be retrieved from Tweetmeme[1] through their API. Topic relevance algorithm relies on the content a user has posted or interacted with before. A bag of words collected from user tweets is indexed and put into a vector following tf-idf approach. High term frequency implies that the user mentioned this word frequently in his tweets. These terms are then matched against URLs and ranked accordingly. 'Social voting process' is based on the 'one person, one vote' approach implemented by Hill and Terveen [11]. If a certain URL is

[1]http://www.tweetmeme.com

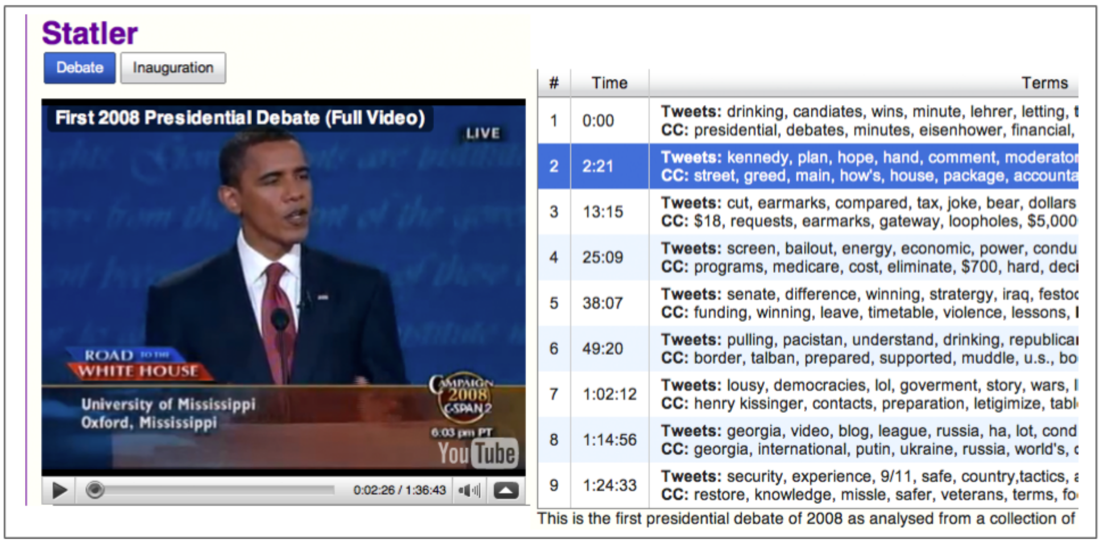
mentioned in the user's Twitter followees, or followees-of-followees, the URL gains a vote. Later, the URLs with the most votes are recommended. Furthermore, a weight is assigned to the votes based on the Tweet author. If that person posts a link every hour, the weight is decreased. On the other hand, if the person posts a link twice a months, the weight is increased.

The recommendation system can be configured in 12 different ways. The results show that the 'Social voting process' improves the performance resulting in approximately 70% useful recommendations. A configuration which only recommends random links from the popular Twitter URLs results in only 30% useful recommendations.

Shamma et al. [36] focuses on the use of Twitter in the annotation of uncollected sources. Their primary analysis was for the use of Twitter during the 2008 Presidential debates. They found that Twitter coupled with live broadcast media can lead to very significant annotation results. Users can freely annotate the event with just a hash tag in their tweet. Questions asked in the paper were: What can Twitter activity say about a media event? Can these data be used to find important moments in the source media? Can topics or higher semantics be discovered? Their study shows that the tweets grow in volume especially at the end and after the event. Furthermore, they discovered that the structure of Twitter traffic can provide insights into segmentation and entity detection. The annotations collected from the tweet can turn the event into real media object.

Shamma et al. [35] studies the capability of using Twitter timeline to disclose information about broadcast events and discover its structure. The research also proposes methods to publish the discovered information to people using an application called Statler[2]. Two videos are used to achieve the purposes: the 2008 USA Presidential debate as shown in Figure 2.1 and the Inauguration of Barack Obama. Around 56000 tweets were pulled from Twitter interface for the analysis. As can be seen in Figure2.1, the table of contents shows at each time interval keywords extracted from the tweets during the event broadcast. These annotations reveal the topic of each segment. Furthermore, Statler also calculates two metrics: importance and chattiness. Importance declares whether a hot topic is mentioned and it is a function of the @ mentions symbol while the chattiness metric measures whether people are paying less attention to the event and chatting with each others. Tf-idf model

[2]http://bit.ly/statler



#	Time	Terms
1	0:00	**Tweets:** drinking, candiates, wins, minute, lehrer, letting, t **CC:** presidential, debates, minutes, eisenhower, financial,
2	2:21	**Tweets:** kennedy, plan, hope, hand, comment, moderator **CC:** street, greed, main, how's, house, package, accounta
3	13:15	**Tweets:** cut, earmarks, compared, tax, joke, bear, dollars **CC:** $18, requests, earmarks, gateway, loopholes, $5,000
4	25:09	**Tweets:** screen, bailout, energy, economic, power, condu **CC:** programs, medicare, cost, eliminate, $700, hard, deci
5	38:07	**Tweets:** senate, difference, winning, stratergy, iraq, festoc **CC:** funding, winning, leave, timetable, violence, lessons, l
6	49:20	**Tweets:** pulling, pacistan, understand, drinking, republica **CC:** border, talban, prepared, supported, muddle, u.s., bo
7	1:02:12	**Tweets:** lousy, democracies, lol, goverment, story, wars, l **CC:** henry kissinger, contacts, preparation, letigimize, tabl
8	1:14:56	**Tweets:** georgia, video, blog, league, russia, ha, lot, cond **CC:** georgia, international, putin, ukraine, russia, world's, c
9	1:24:33	**Tweets:** security, experience, 9/11, safe, country,tactics, a **CC:** restore, knowledge, missle, safer, veterans, terms, fo

This is the first presidential debate of 2008 as analysed from a collection of

Figure 2.1: Screenshot of Statler: Video of Obama and McCain debate with annotated information extracted from Twitter

is adopted to measure the importance of the individual terms based on their frequency normalized by the total number of documents in which the term appears. These results can be used to provide real-time feedback during the event broadcast as well as to tag the video segments for post-event consumption.

Weng et al. [43] proposes a system called TwitterRank (inspired from Google's PageRank algorithm) to find the influential users on Twitter. Several benefits can result from identifying these users: search results can be ranked based on their authors, they can be used in marketing campaigns widely available in Twitter. Two phenemona are studied in this paper: reciprocity and homophily. Reciprocity refers to the idea of following and being followed by the same person. Their findings proved that out of the 1000 users picked for the data set that 72.4% follow more than 80% of their followers and 80.5% have 80% of their friends follow them back. This might be due to the curiosity of the people to follow back who follows in a casual manner. Homophily is the act of following friends because of the interest in the topics they usually post about. Topics are extracted from the tweets data set using Latent Dirichlet Allocation (LDA) model, an unsupervised machine learning algorithm following bag of words approach to recognize topics from large collection of documents.

Cheong and Lee [4] adopt artificial intelligence data mining methods to categorize the tweets. They also study the relation between the user characteristics based on his profile, the topics mentioned in his tweets and the device used to tweet. In May 2009, they randomly selected four top trending topics from Twitter which were Grey's Anatomy, H1N1, Nizar and TwitHit. They also categorized the Twitter users into following groups: 'Personal', 'Group' (e.g. a fanclub), 'Aggregator' (e.g. news agencies), 'Satire' (e.g. sarcasm and humour) and 'Marketing'.

Sriram et al. [40] studies short text classification in Twitter to improve information filtering. They suggested 8 features which consist of one nominal and other seven binary features which are presence of shortening of words and slangs, time-event phrases, opinioned words, emphasis on words, currency and percentage signs, "@username" at the beginning of the tweet, and "@username" within the tweet. The authorship feature falls under social-based filtering. Their study shows that their approach combined with bag-of-words model performs better than bag-of-words model solely.

2.4 Named Entity Recognition in Twitter

Doehrmann [7] studies Named Entity Recognition in colloquial settings such as Twitter. NLP becomes a sophisticated task when it comes to text written in non-standard English. For this reason, the paper suggests an approach for text normalization as a preprocessor to increase the precision of NER systems. Text normalization will correct any spelling mistakes, replace acronyms for phrases and fix grammatical sentence structures to a certain extent. The study shows that this improves the task of NER on Tweets as they depend on sentence structure and context to be able to extract entities. Another suggested approach is to improve the effectiveness of NER tool through the use of cross-referencing classifiers. Differently trained classifiers will be able to find entities missed by other and then a voting algorithm is implemented to be able to improve combined precision and recall and consequently F-measure of the system.

Wagner et al. [41] proposes an approach that leverages social awareness streams such as Twitter to collect information about particular topic. This is achieved by developing a network-theoretic tri-partite model called 'Tweetonomies'. The model consists of users, messages and resources referring to the content of the messages. This system helps researchers to analyze and compare different streams. The topic of extracting unstructured data from structured data has been studied and all the research depends always on the content of the web pages and the links existing between them. For example, consider the topic 'support vector machines'. In order to aggregate a stream which deals with this topic, all messages which contain the hashtag '# svm' could be added. The stream can as well be aggregated by searching for the key words 'support vector machines' in the messages. Also, it's possible to browse through a user directory which deals with support vector machines. As one can see, for the same topic, there are different ways to aggregate streams. To study the different properties of the streams, several measures have been defined, these measures are used later with the tri-partite built model to extract semantics from Twitter:

- Social diversity measures the number of distinct users in a stream
- Conversational Diversity measures the number of active users in a stream that are mentioned with '@' or by slashtags
- Conversational Coverage counts the number of messages incorporated into conversations.
- Lexical Diversity counts the number of unique keywords
- Topical Diversity represents the average number of topics per message
- Informational Diversity measures how many messages have an informational character
- Spatial Diversity counts the number of different locations attached to tweets.
- Temporal Diversity measures the period of time elapsed to cover a particular conversation and the number of messages participating in it.

The results show that hashtag streams reveal information that can be extracted as knowledge about external events. The results indicate that different aggregations lead to different semantic models.

Locke et al. [21] describe an approach to create a Named Entity Recognizer for use on Tweets. The approach relies on transferring knowledge and learning techniques from

Named Entity Recognition in formal sources like newswire to unstructured data of microblogging. They are classifying their Named Entities as People, Locations and Organizations. Locke [21] mentioned that in transfer learning, a classifier is trained on data coming from one distribution, called the source domain, with the intent of using that classifier on another domain, called the target domain. It is a tricky task especially when source and target domains are different. Their source domain was the CoNLL-2003 named entity dataset which is a multilingual corpus for Named Entity Recognition compiled for the Conference on Natural Language Learning shared task in 2003. Their experiments were only held on the English part. The target domain consisted of 600 tweets collected from Twitter Public Timeline with some filtering constraints imposed. They were mainly concerned with three events in tweets: the first event is the economic recession, the second is the Australian Bushfires and the last is the gas explosion in Bozeman. As a comparison to the transfer learning approach, a corpus of 1684 annotated tweets was used as a training dataset. Support Vector Machines algorithm is used to classify entities. When training the SVM with the CoNLL training corpus and tested on Twitter test set, the performance was extremely poor. It scored an F_1 measure of 31.05 % in contrast to 88.19 % when tested on CoNLL dataset. The system proved to perform better when the SVM was trained on Twitter corpus instead. The F_1 measure increased to 59.50 % . This paper proved that it is difficult to develop a classifier that can transfer learning from one domain to another.

Ritter et al. [32] runs an experimental study for recognizing named entities in Tweets. They evaluated the performance of existing NLP tools trained on news articles on Twitter. The POS tagging accuracy dropped from 0.97 to 0.80 on tweets. According to this result, they suggested a novel approach where LabeledLDA is applied. LabeledLDA uses Freebase (open-domain database) as a baseline to evaluate performance. They were mainly concerned by POS tagging, noun-phrase chunking and capitalization. The performance degraded due to the large number of out of vocabulary words present in tweets due to the informal slang and spelling mistakes. For this reason, manual annotation of tweets was necessary to be used as training dataset. The training dataset consisted of 2400 manually annotated tweets with named entities. Their approach increased the F_1 measure by 25% in NER.

Finin et al.[8] and Murnane et al. [26] describe the approach of annotating large volumes of twitter data using MTurk and CrowdFlower. These annotations can be further used for information extraction from less structured text like Twitter and Facebook. MTurk as described by Amazon.com is a service that gives businesses access to a diverse, on-demand, scalable workforce and gives workers a selection of thousands of tasks to complete whenever convenient. Amazon Mechanical Turk is based on the idea that there are still many things that human beings can do much more effectively than computers. It is a service at low cost. It uses standard HTML and javascript to create the interface, in contrast with CrowdFlower which is a similar service but uses CrowdFlower Markup Language (CML). The workers are presented with an interface where each tweet is presented as a sequence of tokens and a type has to be assigned to each token according to their enforced annotation guidelines. The token can be tagged to be Person, Location, Organization or none of these. Each task consisted of five tweets in reward of five cents. They were able to annotate 4400 tweets with $ 100. CrowdFlower task involved 30 tweets, where each task had three tweets and was paid five cents. Since they are paid services, there was an incentive for workers to produce results of good quality. These labeled tweets are then considered to be gold data for training existing Named Entity Recognizers.

Liu et al. [20] propose an approach to improve Named Entity Normalization for Tweets. Named Entity Normalization (NEN) is the process of converting each encountered word to its original unambiguous form. They found that the results of NEN system are affected by NER as well as the lack of information in tweets which are only 140 characters in length. Therefore, they present a new method of creating a graphical model between multiple tweets. Considering multiple tweets at a time allows the model to recognize entities related to the same mention and thus normalizing them.

Liu et al. [19] propose a novel NER system that combines K nearest neighbor (KNN) to capture word level classification leveraging their similarity to a set of labeled tweets followed by Conditional Random Fields (CRF) model to find the fine-grained entities in tweets. This two-level combination is further augmented by thirty gazetteers which contain knowledge about county names, locations and temporal expressions to boost the performance. It is a semi-supervised learning approach that exploits both labeled and unlabeled data. The gold standard data consisted of 12,245 manually annotated tweets. The focus was to distinguish four types of entities: persons, organizations, products and locations.

Their method shows that they outperform the baselines on all entity types with an increase of F_1 score from 75.4 % to 80.2%.

Chapter 3

System Architecture

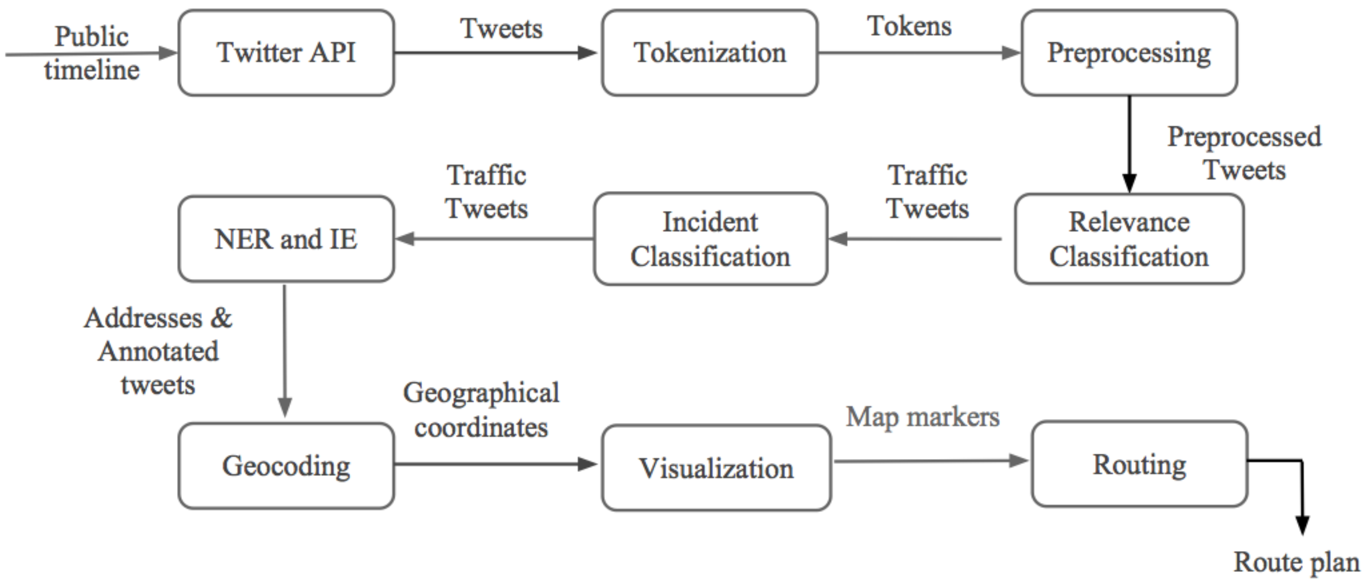


Figure 3.1: System Architecture

This chapter gives an overview about the system architecture as shown in Figure 3.1. Each system module is described briefly, a more detailed description for some modules is given in the following chapters.

Modularity in design is the approach used in the system architecture. This approach divides the system into several components or modules in an isolated separate manner with limited dependencies between them. This concept has several characteristics:

- Each functional module has its own interface
- Scalability of components
- Reusability of separate modules
- Unit testing facility
- Ease of change of particular modules to allow technology transparency

An overview about each module is given below in the order shown in Figure 3.1:

- **Twitter API**
  Twitter allows access to a huge corpus of data through different APIs (REST API, Search API and Streaming API). Each of these APIs is explained thoroughly in Chapter 4. In this module, the Twitter Streaming API is used to retrieve tweets from public timeline which are the tweets publicly available in real-time. Several request parameters have to be set when sending the HTTP request such as `track`, `follow` and `locations`. These parameters are used to customize the request and filter out some non relevant tweets. `track` is a list consisting of keywords; tweets containing any of these keywords are returned in the response, `follow` contains twitter user ids where their tweets are returned by the HTTP response and `locations` is a filtering criteria imposed to return tweets within certain geographical bounding box. More details about these parameter values can be found in appendix A.4.

- **Tokenization**
  An essential preprocessing step for classification and Named Entity Recognition is Tokenization. This refers to breaking each incoming tweet into tokens. Sentence boundaries are identified first then sentence segmentation. A white space tokenizer, which segments words in a sentence based on whitespace between them, is used. Additionally, some punctuation marks (More details in appendix A.3) are removed from the token list during this phase. Each incoming tweet is wrapped into an object carrying the following information:
  - Original tweet
  - Timestamp of the tweet obtained through Twitter API
  - List of tokens

- **Preprocessing**
  Once tweets are retrieved from Twitter with the help of the API, a preprocessing phase is applied for each tweet to prepare the data for the various consequent natural language processing tasks such as text classification, Named Entity Recognition and information extraction. Stop-words such as 'the', 'a', 'as', 'such', . . . , are detected and removed in this phase. We do not use a standard list of stop-words, but a designed one according to the needs of application. For example, in our system, words like 'in', 'at', 'way', that would usually be removed as stopwords, are not removed because they play an important role in recognizing locations. The original list of stopwords was retrieved from the web[1]. Furthermore, since tweets are limited in message length, shortened URLs are often used in tweets. The URLs are filtered out since they do not contribute any information for the NLP tasks. The complete list of stopwords can be found in appendix A.1.

- **Relevance Classification**
  A decision should be taken concerning each incoming tweet. Each tweet has to be classified into one of two classes *traffic* and *junk*. A *Traffic* tweet is one mentioning any information related to traffic in San Francisco while *Junk* refers to any tweet not relevant to our research. Supervised Machine learning techniques are used in the text classification module. Three different text classification techniques are used: Naive Bayes, Maximum Entropy and Support Vector Machines (SVMs). Each of these techniques is presented in details in Chapter 5 as well as a basic majority voting system with the combination of these three algorithms. More details can be found in Section 5.6.3.

- **Incident Classification**
  A further classification is performed for *traffic* tweets. Tweets related to traffic do not always mention that there is a traffic jam or incident blocking a road (*Negative* tweets). Rather, it sometimes mention that the traffic at a particular place has been cleared or that road construction has been finished (*Positive tweets*). There is a need to differentiate between these two classes. The classifiers presented in Chapter 5 are used to achieve this type of classification. This is a crucial step since the classification

[1]http://code.google.com/p/iestwitter/ source/browse/trunk/src/ontology/tech/StopWords.java retrieved in June, 2012

result is the main parameter used to assign weights to edges of the graph for routing. A higher weight is assigned to an edge having traffic problem and a weight equal to the distance is assigned to edges where traffic is eliminated. More details can be found in Section 5.6.4.

- **Named Entity Recognition (NER)**
  NER is a subtask of information retrieval (IR). Entities are discrete objects in the world and the task of NER is to find and classify these entities to enable Information Extraction. NER has been well studied on structured data. However, well trained model on structured data proved to have low performance on Twitter and other forms of unstructured data. In our research, we use a two-phase NER system. The first phase consists of training Stanford NER Conditional Random Fields classifier to recognize locations in tweets. The second phase is supported by an especially built dictionary containing frequently occurring words in tweets related to traffic. Our classes of named entities are LOCATION, HIGHWAY, DIRECTION, PREPOSITION, VOCABULARY and WORD. More details can be found in section 6.3 and the dictionary can be found in appendix A.2.

- **Information Extraction (IE)**
  During this step, structured information is extracted from the unstructured Twitter data and a template is filled for each tweet. The template contains the following fields:

  – Start point and end point: Of traffic or blocked road
  – Ref-point(s): A junction of roads or a particular landmark mentioned in a tweet.
  – City: City or district mentioned in a tweet where an incident has occurred
  – Highway(s) and direction(s): The highways affected by the incident as well as the direction whether Northbound, Southbound,...
  – Traffic Condition: Positive or negative as classified during the text classification step

- **Geocoding**
  The mapping of the extracted information into geographical locations is achieved through Geocoding. Geocoding is the process of converting textual addresses into

coordinates in the form of longitude and latitude. Google offers a Geocoding service[2] that can return geographical coordinates via HTTP. All addresses extracted during the IE step are sent to the Geocoding API to obtain their coordinates. Details about request parameters and response format is presented in section 7.3.

- **Visualization**
  All locations where incidents have been recorded are plotted on a map by placing markers at the specified coordinates returned by the geocoding API. For this task, we use JXMapViewer[3] , a java Swing Component, similar to JPanel that visualizes a map. JMapKit loads tiles from OpenStreetMap server and presents it. More details can be found in Chapter 7.

- **Routing**
  Real time constraints such as traffic information are then used for route planning. Dijkstra, Shortest Path Algorithm for weighted graphs, is used to obtain the path between start and destination points. The link between two points is an edge in the graph representation. Each edge is assigned a weight based on distance. However, another weight function is added based on the traffic level as extracted from tweets. This weight combination is the basis to provide the shortest path between two points. However, in our application, the term shortest does not necessarily mean shortest distance but more precisely means shortest time depending on the traffic level. More detailed explanation of the graph presentation and routing algorithm is given in Chapter 7.

[2]https://developers.google.com/maps/documentation/geocoding/
[3]http://wiki.openstreetmap.org/wiki/JXMapViewer

Chapter 4

Twitter

Microblogging is a form of communication where users can share short messages with others through IM, mobile phones or the web. Twitter is the most widely known microblogging service. It emerged in March 2007. It allows each user to share his status updates with the world in messages of 140 characters called *Tweets.* Tweets can be used for many purposes including reporting local news, sharing an answer to the question *What are you doing* with the world, rebroadcasting information and asking questions (Mackice et al. [22]) Twitter does not restrict the user, in contrast, it gives the user freedom to tailor it to his own needs resulting in diverse experiences. TweetRush, a third party application, estimated peaks of about 1.9 million tweets per day in January 2009. Twitter is still growing and as mentioned by Twitter for developers, that Twitter produces more than 340 million tweets a day and has about 140 million active users by Twitter sixth birthday in March 2012[1].

A user can *follow* others without being their friends. A *Friend* in Twitter is when two users are following each other. Twitter can be viewed as a graph where users are nodes and their relations are edges of the graph. In contrast with other social media like Facebook, Twitter relations do not have to be bidirectional. Java et al. [15] categorizes Twitter users into three main categories:

- **Information Source** These usually have a large number of *Followers.* They do not need to be frequent posters, however, their information is valuable and they serve as a hub of information.

- **Friends** There are many sub-categories of friendships on Twitter. For example a

[1]http://mashable.com/2012/03/21/twitter-has-140-million-users/

user may have friends, family and co-workers on their friend or follower lists.

- **Information Seeker** is a user who posts status casually but follows other users regularly.

Some other keywords of Twitter language include *retweet* where a user reposts a previous tweet by another user.

4.1 Twitter API

Twitter allows access to this huge corpus of data through different APIs that facilitate developers building new and creative applications. Each API is described briefly, more information can be found at Twitter Developer Website[2].

Search API
The Search API allows querying Twitter content. This can be achieved by specifying a set of keywords and the API returns back tweets containing these keywords. It can also be used to track tweets of a particular user. However, it is advised by Twitter developers to use the Streaming API instead if the application is hitting the limit rates.

REST API
REpresentational State Transfer (REST) is a style of software architecture that was introduced in 2000 and since then replaced its predecessors SOAP and WSDL due to its simplicity in building distributed systems such as the World Wide Web. REST is characterized by the generality of interfaces as well as scalability of its components. The Twitter REST API conforms to the REST standards where the change of the request format results in an automatic change in the response format such as XML, JSON, RSS and ATOM. The REST API enables developers to access some of the core primitives of Twitter including timelines, status updates, and user information. It also allows the user to create, post tweets, retweet and reply back to Twitter through RESTful calls. The REST API follows a *polling technology*, an internet based style, where the request for the transmission of information is initiated by the receiver as can be seen in Figure 4.1.

[2]https://dev.twitter.com/docs/

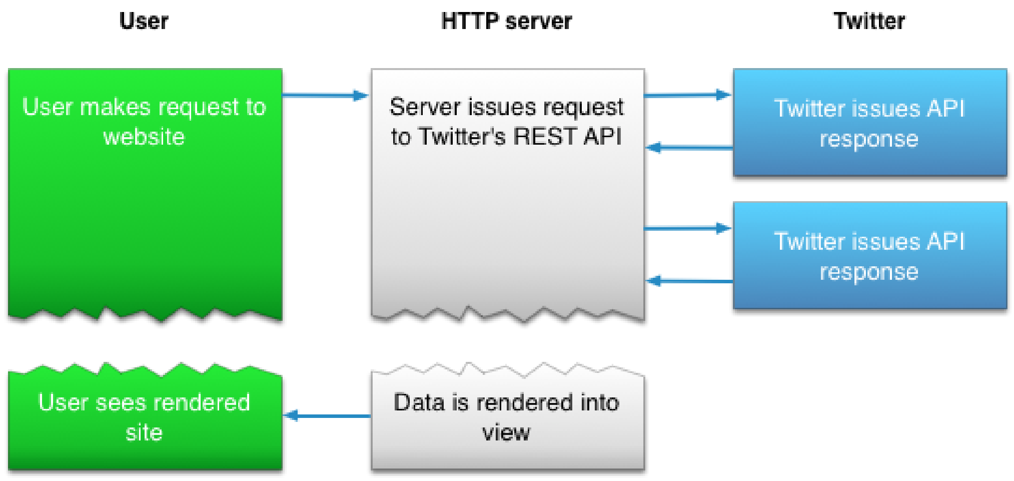

User
HTTP server
Twitter
User makes request to website
Server issues request to Twitter's REST API
Twitter issues API response
Twitter issues API response
User sees rendered site
Data is rendered into view

Figure 4.1: REST API[3]

Streaming API

The Streaming API is the real-time sample of the Twitter *Firehose* - Twitter speak of all incoming public statuses. This API is for those developers with data intensive needs. It is most suitable for data mining products and analytics research. The Streaming API allows for large quantities of keywords to be specified and tracked, retrieving geo-tagged tweets from a certain region, or have the public statuses of a user set returned. This requires the establishment of a long-lived HTTP connection and to maintain that connection. This API follows a *push technology* concept where only the request for initial communication is initiated by the subscriber. Once a connection is established between the subscriber (the application) and the publisher (Twitter), Twitter keeps pushing incoming tweets depending on the request initiated by the application. In contrast to the REST API, maintaining the streaming connections and handling HTTP requests are run with different processes. The streaming process gets Tweets as a response, performs any filtering depending on the request customization and stores the result to a certain data store. Then, the HTTP handling process queries and retrieves the responses from the data store as indicated in Figure 4.2.

In our research, we decided to use the Streaming API since it is most suited for our application as it has a limit rate of 20000 requests per hour. Furthermore, the streaming

[3]https://dev.twitter.com/docs/ retrieved on 9 Aug, 2012

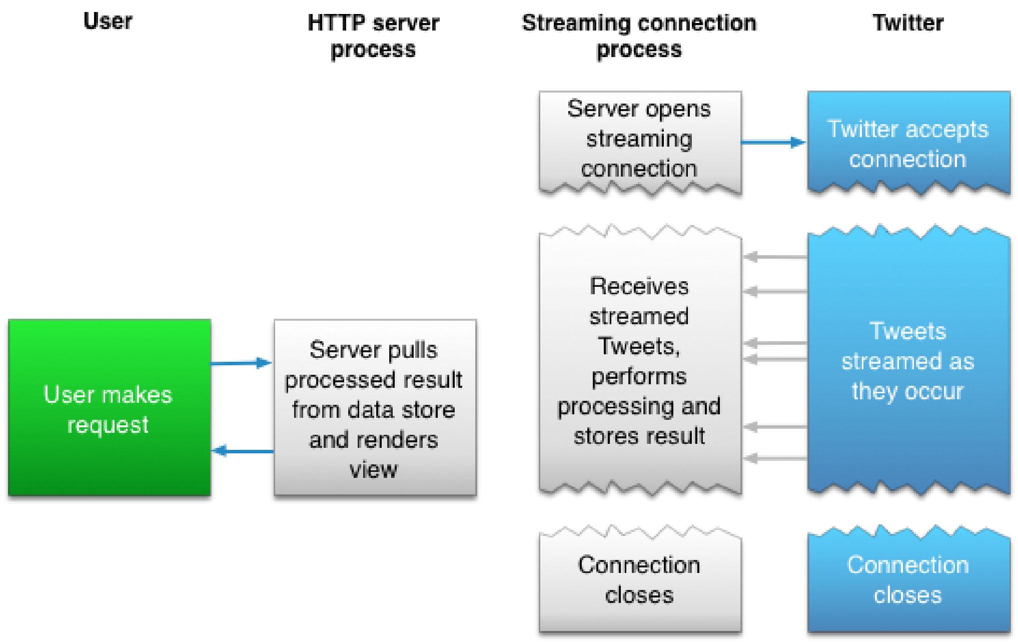

User
HTTP server process
Streaming connection process
Twitter
Server opens streaming connection
Twitter accepts connection
Receives streamed Tweets, performs processing and stores result
Tweets streamed as they occur
User makes request
Server pulls processed result from data store and renders view
Connection closes
Connection closes

Figure 4.2: Streaming API[4]

API offers several streaming endpoints, each for a specific use:

- **Public streams** a streaming channel of incoming information. It can be customized to follow certain users or topics.
- **User streams** a single user stream which returns all the data corresponding to a particular user.
- **Site streams** a multi version of user streams. It is most suited for applications having multiple servers which need to connect to Twitter.

From the brief description about each stream mentioned above, we select public stream endpoint for our research since it is most suited for data mining because we need to retrieve data from several users and we have only one server.

Connecting to Streaming Endpoint

Connecting to the streaming endpoint requires authentication. All requests sent to Twitter API over HTTP must be authorized. For this purpose, OAuth - an open standard for

[4]https://dev.twitter.com/docs/ retrieved on 10 Aug, 2012

authorization is used. Several parameters must be passed with the HTTP request to the Twitter API to allow the authentication of the application. Each application is granted access to the following parameters by Twitter API once the application that will make use of Twitter API is created. These parameters need to be passed with every HTTP request that requires authorization:

- oauth consumer key: the key identifies the application to Twitter
- oauth consumer secret: the secret value identifies the application to Twitter
- oauth access token: the access token identifies the account to which the application is connected
- oauth access token secret: this secret identifies the account the application is acting on behalf

Once the connection is established, it will be held open indefinitely by Twitter API and the HTTP Client will return the response data incrementally.

Streaming API request parameters
These are the most important parameters/values for our application:

delimited This parameter is set to a particular `length` which denotes the number of bytes to read off the stream from the response. By setting this parameter, the HTTP client knows how many bytes to read before the end of the response message.

stall warnings Setting this parameter to `true` allows the delivery of warning messages whenever there is a risk to lose the connection with the client.

follow A list of user ids can be specified which indicates the users whose Tweets should be pushed to the client. This parameter allows the customization of the request sent to Twitter depending on our needs. A research has been conducted to search for users who frequently tweet about traffic in San Francisco and their IDs have been added to the list that can be found in appendix A.4.

track A set of keywords or phrases to be sent with the search query. The returned tweets should contain one or more of the keywords that have been listed. This parameter can be used to filter the incoming stream by including keywords that frequently occur in traffic tweets like "accident", "lane", "incident", "highway" etc. The complete list is provided in appendix A.4.

locations A comma-separated list of longitude and latitude to filter geo-tagged tweets depending on the specified bounding box. Each bounding box should be specified by longitude and latitude of its Northeast and Southwest corners. Our request is customized to the geo-positions of San Francisco area which are -122.75,36.8 and -121.75,37.8

Processing Streaming data
The streaming API responses are JSON encoded. JSON, JavaScript Object Notation, is a lightweight data interchange format where every object is represented as a key-value pair. It is derived from the JavaScript language. It is characterized by its simplicity and is usually used to transmit data over networks as an alternative to XML. Two data structures are the basis of JSON format :

- Object: an unordered collection of key-value pairs. The name is a string and the value can be one of different data types: string, number, boolean, array or another object.
- Array: an ordered list of values.

Twitter uses a service called Snowflake to assign unique IDs for every tweet returned. This ID is a 64bit unsigned integer which contains a timestamp of the tweet and a sequence number. Since not all programming languages can process large integers. Twitter decided to return the ID in a string format as well as encoded in the response. Using the JSON format, the field `id_str` should be used instead of `id` because JSON cannot process integers greater than 53 bits.

Example Twitter Request
http://api.twitter.com/1/statuses/public_timeline.json

Example Twitter Response

An example of Twitter response can be viewed in figure 4.3.

This chapter presented the different APIs offered by Twitter to retrieve the publicly available tweets in real-time. A description of the parameters that need to be set for each request sent to Twitter can be seen in section 4.1. The Twitter Streaming API has been adopted in our research because it is mos suitable for data mining tasks and the limit rate is high.

```
 {
   "place":null,
   "geo":null,
   "retweeted":false,
   "text":"Very unlucky played well. Scholes going off made a big diff .
   "in_reply_to_status_id_str":null,
   "in_reply_to_status_id":null,
   "in_reply_to_user_id_str":null,
   "truncated":false,
   "in_reply_to_user_id":null,
   "contributors":null,
   "retweet_count":0,
   "favorited":false,
   "created_at":"Sat Jan 28 14:46:21 +0000 2012",
   "coordinates":null,
   "id_str":"163271701172465665",
   "user":{
      "show_all_inline_media":false,
      "profile_text_color":"333333",
      "statuses_count":2224,
      "screen_name":"patfoxyboy",
      "listed_count":18,
      "geo_enabled":true,
      "friends_count":843,
      "description":"Deise lad  .Married Father of two . Manchester United fanatic",
      "default_profile":true,
      "notifications":null,
      "profile_background_tile":false,
      "created_at":"Sun Jul 03 13:10:37 +0000 2011",
      "protected":false,
      "default_profile_image":false,
      "contributors_enabled":false,
      "followers_count":577,
      "name":"Patrick Fox",
      "id_str":"328480065",
      "favourites_count":21,
      "id":328480065,
      "lang":"en"
   }
}
```

Figure 4.3: Example Twitter Response

Chapter 5

Text Classification and Filtering

The task of text classification is to determine for an object to which class it belongs out of a given set of classes. This process can be done manually or using a machine learning approach. For a supervised machine learning approach, we need a set of good example documents referred to as training documents. The training dataset consists of a collection of labeled documents where labeling refers to the process of annotating each document with its class. Using a learning method, we then wish to learn a classifier to map documents to classes. This is called *supervised learning* because the process is supervised by a human who assigns the labels (Manning et al. [24]).

This chapter describes the two text classification modules as mentioned in chapter 3. We adopt three well known algorithms in text classification which are: Naive Bayes, Maximum Entropy and Support Vector Machines. An introduction about each of these algorithms is presented in a separate section (section 5.1, section 5.2 and section 5.3 respectively). Section 5.4 deals with the application of these techniques in our research as well as our feature selection process for the two tasks, relevance classification and incident classification.

Each of these algorithms follow under the content-based filtering paradigm. We focus on using the content for classification. As a result, document representations in content-based filtering systems can exploit only information that can be derived from document contents and do not depend on the author or other meta-information such as timestamps.

5.1 Naive Bayes Text Classification

Naive Bayes is a supervised probabilistic learning method. It is based on the bag-of-words model. The bag-of-words representation neglects every information about the position of a word in a document and instead focuses only on the word frequency in the document. Each document will be represented as a set of words and their frequency count.

The conditional probability $P(c|d)$ for a document d and a class c according to Bayes rule is:

$$P(c|d) = P(c) \cdot \frac{P(d|c)}{P(d)} \tag{5.1}$$

where $P(c)$ is the probability of the class c, $P(d|c)$ is the probability of document d given the class c and $P(d)$ is the probability of document d.
For each document d and class c, the probability is computed using equation 5.1. Each document d is presented as a set of words $w_1, w_2, w_3, \ldots, w_k$. We estimate the probability $P(d|c)$ of a document d given a class c as the multiplication of the probabilities of each individual word w_i given the class c. This is due to two assumptions:

- Bag of Words assumption.
- Conditional independence: the word probabilities in the document $P(w_i|c)$ are independent given the class c.

$P(d|c)$ can then be written as follows:

$$\begin{aligned} P(d|c) =& P(w_1, w_2, w_3, \ldots, w_k|c) \\ =& P(w_1|c) \cdot P(w_2|c) \cdot P(w_3|c) \cdot \ldots \cdot P(w_k|c) \\ =& \prod_{w_{i=1,2,\ldots,k}} P(w_i|c) \end{aligned} \tag{5.2}$$

However, the probability $P(w_i|c)$ might be equal to zero. To eliminate zeros, we use add-one or Laplace smoothing, which simply adds one to each word w_i count. Add-one smoothing can be interpreted as a uniform prior (each term occurs once for each class) that is then updated as evidence from the training data coming in.

Then the probability of a class c needs to be computed. This is equal to the relative frequency of the class c in the corpus of training documents. It denotes how often does this class occur.

$$P(c) = \frac{N_c}{N} \tag{5.3}$$

where N_c is the number of documents in class c and N is the total number of documents.

During the test phase, we calculate the maximum a posteriori (MAP) class c_{map} for each document d we need to classify. c_{map} is the class whose probability given the document is the maximum.

$$\begin{aligned} c_{MAP} =& \arg\max_c P(c|d) \\ =& P(c) \cdot \frac{P(d|c)}{P(d)} \\ =& P(c) \cdot \frac{\prod_{w_{1,\dots,k}} P(w|c)}{P(d)} \end{aligned} \tag{5.4}$$

The class that maximizes the conditional probability c_{map} is chosen.
More details about the algorithm can be found in Manning et al. [24].

5.2 Maximum Entropy

Classifiers can fall under one of two categories: Generative and Discriminative. Generative models generate the observed data from hidden states like Naive Bayes. In a generative model, the focus is on joint probability and we want to maximize this joint likelihood. In contrast, discriminative models target the classification decision that we want to make. Discriminative models take the data as given and put a probability over hidden structure. Currently there is much use of discriminative models because they tend to have high accuracy performance and they are linguistically interesting because of the ease of adding lots of linguistic features. The discriminative model focuses on the conditional probability and seeks to maximize it.

The Maximum Entropy (Maxent) classifier is a discriminative classifier. Each document d is represented as a set of features $f_1, f_2, \dots, f_k$. The model calculates a linear function of these features. This linear function represents a score for each class and can be written as:

$$vote(c) = \sum_i \lambda_i f_i(c, d) \tag{5.5}$$

where λ_i is a weight assigned to each feature f_i. $vote(c)$ is calculated for each class c and the class that maximizes it is then chosen.

5.2.1 Building a Maxent Model

In this section we describe the steps required to build a Maxent model

Converting the linear combination to a probabilistic model

The problem with linear combination in Equation 5.5 is it comes out with either positive or negative value and that does not work for probability. Taking the exponential of the value assures that we always get positive weights despite the sign of this value. Normalization over all possible classes is then used to convert the value to probability estimate.

$$P(c|d, \lambda) = \frac{\exp \sum_i \lambda_i f_i(c, d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c', d)} \tag{5.6}$$

Training feature weights

Given this model form, we will choose parameters λ_i that maximize the conditional likelihood. The selection of features is basic requirement for the use of discriminative models. Features f are elementary pieces of evidence that link aspects of what we observe in a document d with a category c that we want to predict. A feature is a function with a bounded real value. The model assigns to each feature f_i a weight λ_i.

The conditional (log) likelihood of a maxent model is a function of the data (C, D) where C is the set of classes and D is the collection of the documents and the parameters λ which are the weights of the features, it can be written as:

$$\begin{aligned} \log P(C|D, \lambda) &= \log \prod_{(c,d)\in(C,D)} P(c|d, \lambda) \\ &= \sum_{(c,d)\in(C,D)} \log P(c|d, \lambda) \end{aligned} \tag{5.7}$$

In practice, this log likelihood can be calculated if the number of classes is small because we are dividing over the likelihood of all classes. The log likelihood can be separated into two components :

$$\log P(C|D,\lambda) = \sum_{(c,d)\in(C,D)} \log \exp \sum_i \lambda_i f_i(c,d) - \sum_{(c,d)\in(C,D)} \log \sum_{c'} \exp \sum_i \lambda_i f_i(c',d) \quad (5.8)$$

These two components are N(λ) and M(λ).

$$logP(C|D,\lambda) = N(\lambda) - M(\lambda) \quad (5.9)$$

To maximize the log likelihood, we need to assign for each feature an optimum weight. To determine these optimum weights, we need to calculate the derivative of the log likelihood function with respect to each feature weight λ_i.

The derivative of N(λ) is the empirical count (f_i, c) :

$$\frac{\delta N(\lambda)}{\delta \lambda_i} = \sum_{(c,d)\in(C,D)} f_i(c,d) \quad (5.10)$$

And the derivative of M(λ) is the predicted count (f_i, λ) :

$$\frac{\delta M(\lambda)}{\delta \lambda_i} = \sum_{(c,d)\in(C,D)} \sum_{c'} P(c' \mid d,\lambda) f_i(c',d) \quad (5.11)$$

Then, the optimum values are the values for which each feature's predicted count in Equation 5.10 equals its empirical count in Equation 5.11. This optimum value is when their differences is equal to zero as shown in 5.12:

$$actualcount(f_i, c) - predictedcount(f_i, \lambda) = 0 \quad (5.12)$$

A more detailed explanation can be found in Manning et al. [23].

5.3 Vector Space Model

The vector space model is the representation of a collection of documents as vectors. This model is essential for several document classification techniques and document clustering.

Each vector corresponds to a single document and each entry in the vector corresponds to a term in the document. Each term in the document is assigned a weight based on its frequency of occurrence in the document and hence its importance. If a term does not occur within a document, it is assigned the value zero. For example if the words in the dictionary are 'Hello', 'World', 'first', 'example'. For a document A containing "Hello World", the vector representation is (1,1,0,0).

However, several approaches can be used to assign weights to terms one of them is TF-IDF (Term Frequency- Inverse Document Frequency). *Term Frequency* in its simplest form is equal to the number of occurrences of a term t in a document d. The term frequency is referred to by $tf_{t,d}$ for term and document respectively. This model is also known as *bag of words model* because it does not take into consideration the order of the terms. Two documents having exactly the same terms but in different order are treated equally according to this model. The only property that matters is the frequency of the terms. However, not all of the terms should be treated equally. For example, the high number of occurrences of stopwords such as 'the' and 'and', does not reveal any importance.

To solve the problem of equal treatment of all terms, another weight scheme can be used. This weight is referred to as *Inverse Document Frequency.* This approach reduces the effect of terms occurring frequently in the collection. The factor by which the weight of a term is reduced is *document frequency* df_t. The document frequency is defined to be the number of documents in the collection that contain a term t. The inverse document frequency idf_t of a term t is defined as follows:

$$idf_t = \log \frac{N}{df_t} \tag{5.13}$$

where N refers to the total number of documents in the document set and df_t is the document frequency of a term t. The idf_t is inversely proportional to the number of occurrences of a term. The idf_t of a rare term is high and the idf_t of a frequently occurring term is low.

$tf_{t,d}$ and idf_t are combined in a weighting scheme tf-idf to assign a weight to each term in the document. The $tf - idf_{t,d}$ assigns to term t in a document d the following weight:

$$\text{tf-idf}_{t,d} = tf_{t,d} \cdot idf_t \tag{5.14}$$

$tf - idf_{t,d}$ has the following properties:

- It is highest when a term t occurs frequently in a small fraction of documents
- It is lower when the term occurs few times in a single document or many times in several documents.
- It is lowest when the term occurs in all the documents.

More details can be found in Manning et al. [24].

5.4 Support Vector Machines (SVMs)

A Support Vector Machine is another example of a discriminative model. SVMs have been applied successfully in text classification. SVM is a large-margin classifier: it is a vector space based machine learning method where the goal is to find a decision boundary between two classes that is maximally far from any point in the training data (possibly discounting some points as outliers or noise).

There might exist several possible linear separators between the two classes in the training data set. However, it is always better to choose a decision boundary which is as far as possible from all data points of the two classes. SVM adopts this approach in choosing a decision boundary in binary classification. It defines a criterion to pick a decision surface that is maximally far away of any data point. This distance is referred to as the margin of the classifier. The construction of the margin mainly relies on the selection of data points known as support vectors that specifically determine the position of the separator. The maximized margin can be shown in Figure 5.1. The large margin ensures the decrease of low certainty classification decisions. This is referred to as the classification safety margin. During the SVM construction, an SVM classifier insists on the choice of a large margin around the decision boundary. A decision hyperplane can be defined by an intercept term b and a decision hyperplane normal vector $\vec{w}$ which is perpendicular to the hyperplane.

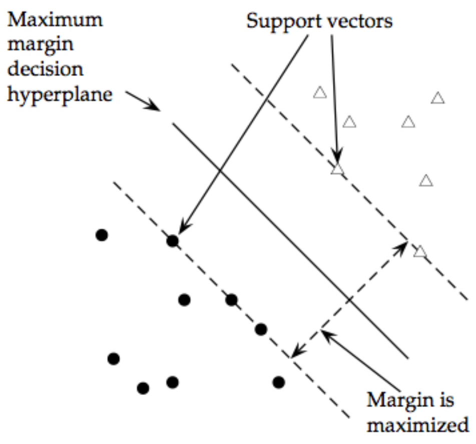


Figure 5.1: Separating hyperplane in SVM.

This vector is called the weight vector. All points $\vec{x}$ on the hyperplane satisfy $\vec{w}^T\vec{x} = -b$. The linear classifier is then:

$$f(\vec{x}) = sign(\vec{w}^T\vec{x} + b) \tag{5.15}$$

The algorithm starts with a training data set D = $\{(\vec{x_i}, y_i)\}$ where each point $\vec{x_i}$ has a class label y_i. For a binary SVM the class labels are always +1 and -1. The data defines the best separating hyperplane and the data is fed into a quadratic optimization procedure to find this plane.

During the test phase, whenever a new point $\vec{x}$ needs to be classified, the classification function $f(\vec{x})$ computes the projection of the point into the hyperplane normal. The sign of the function determines the class to assign the point to. More details can be found in Manning et al. [24]

5.5 Voting System

A voting system combines the output of the three classifiers (Naive Bayes, Maxent and SVMs) to improve the classification performance. With combination techniques the objective is to emphasize the strengths of individual classifiers while diluting their weaknesses.

In the simple voting scheme used in this work, each classifier has a single vote. Once a tweet comes in, it passes through phases of feature selection then text classification performed by each classifier. After all classifiers have voted, the votes are collected and the class label having the maximum number of votes is selected. This model could be further extended by including a probability based voting scheme or a weight based weighting scheme. This was not done in this work as the results obtained by the simple voting scheme were satisfactory.

5.6 Our Implementation

Twitter data comes at a very high throughput rate and the attractiveness about them is that they capture the moment. Therefore, there is a need to implement an online system that can filter tweets one at a time and not to halt to process them in batches. However, this is a challenging task.

This section describes our application of the text classification algorithms explained in previous sections. Two classifications are performed in our research as visualized in Figure 5.2:

- Traffic or junk: distinguish *traffic* tweets that are important for the task and will be used in further processing, from other tweets referred to as *junk* in this work.
- Positive or negative: each *traffic* tweet is classified into *positive* if it mentions that lanes are clear or road is open; else it is *negative* if it mentions that a lane or road is blocked.

First, we describe feature selection then details about each of these two classifications is given in the following sections.

5.6.1 Social-Based Filtering

According to Oard [29], the relevancy of a document can be determined by its author. Applying this argument to our research, the authorship feature can play an important role in filtering a considerable number of tweets related to traffic. We have manually identified a number of twitter accounts concerned with publishing tweets about San Francisco traffic, the accounts are listed in appendix A.4. If the tweet is posted by any of these accounts, the

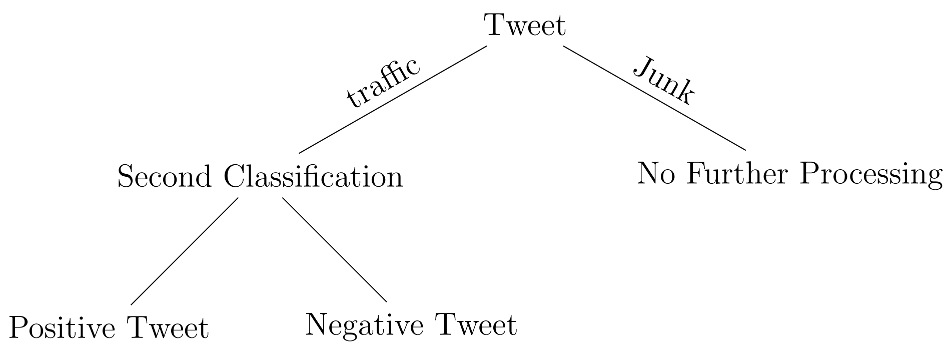


Figure 5.2: Text classification in our implementation

tweet should be selected as potentially relevant. However, Social-based filtering cannot be used solely for the sake of filtering and must be accompanied with content-based filtering which is a more mature concept. For this reason, we use the previously described text classification techniques as described in the following sections.

5.6.2 Feature Selection

The purpose of feature selection is to determine which features are the most relevant to the current classification task. In text classification, features are typically words from a document. Choosing an appropriate feature selection method can be vital because of the large number of features usually present in text documents.

Sebastiani et al. [34] mention that a typical text document will contain hundreds of possible features, and it can be extremely difficult to produce an accurate classification without some sort of feature selection. However, since tweets are only 140 characters long, features chosen carefully can greatly help filtering out noise tweets.

Following Nicolosi et al. [28], we apply pruning prior to feature selection to reduce the number of possible features. This is particularly important because the number of possible features is typically very large in a text document, and it is likely that most of these features are irrelevant. Generally, very rare and very frequent words are commonly eliminated. For example, stop words in English language like ”a”, ”an”, ”the” . . . are very common and their frequency does not contribute to the importance of a document. Features selected in each of the two classification tasks are described in the two following sections.

5.6.3 Relevance Classification: Traffic or Junk

Inspired from Sankaranarayanan et al. [33], we classify incoming tweets as either *traffic* or *junk*, where traffic tweets have a chance of being related to traffic and hence kept for further processing, while the junk tweets have a chance being non-related to traffic and hence discarded. Our goal is not to completely get rid of junk tweets which might not even be possible but at least to discard tweets that clearly cannot be traffic. First, stopwords and URLs are removed from tweets and then features are selected.

In this type of classification, our features are the words extracted from *traffic* tweets that were further refined to include distinctive features like places, roads, streets, avenues and boulevards in San Francisco. All these words have been manually collected throughout 6 months and stored in our built dictionary described in details in section 6.3.2. The dictionary vocabulary is listed in appendix A.2.

Of course these features are not complete but they give a good indicator of whether a tweet is related to traffic in San Francisco or not.

5.6.4 Incident Classification: Positive or negative

Text classification is used for a second task in our research. A distinction should be made between traffic-related tweets mentioning that an incident or a traffic problem is happening and traffic-related tweets mentioning that traffic has been cleared and the roads are back to their normal states. This classification helps in the routing process since there is no need to penalize edges in locations where traffic is clear and lanes are open. The features for this classification task are all the words in the tweets after the tokenization except stopwords and URLs.

Chapter 6

Named-Entity Recognition (NER)

Named Entity Recognition (NER) is a subtask of information retrieval. It is concerned with finding and classifying named entities in text. Entities are meant to describe discrete things in the world. Standard categories of named entities include ”person”, ”location”, ”geo-political organization”, ”facility”, ”organization” and ”time”. NER has been well studied and can be divided into three categories: Rule-based (e.g. [17]), machine-learning based (e.g. [9], [37]) and hybrid methods (e.g. [13]). Our research adopts a hybrid method which is presented in details in section 6.3. We make use of Stanford NER system[1] which is based on Conditional Random Fields. An introduction about conditional random fields is presented in section 6.1 followed by details about stanford NER in section 6.2, our customized NER system is described in section 6.3 and Information Extraction (IE) is described in section 6.4.

6.1 Conditional Random Fields

Assigning labels to a sequence of words is one of the tasks of Natural Language Processing (NLP). It is a preprocessing task of NLP for higher tasks. Each element in the sequence of data is annotated during this task with a label carrying descriptive information depending on the application. NER is an example of this task where each word is labeled with a class indicating its named entity type.

A model used for labeling sequence of data is the conditional model. It is a discriminative model that satisfies two criteria: support of inference and representing data without

[1]http://nlp.stanford.edu/ner/index.shtml

making any non-guaranteed assumptions. The probabilities of a possible label sequence according to the observation are specified and no further effort is spent on modeling the observations that remain unchanged during the training phase. A conditional model assigns the label y_* which maximizes the probability $\mathrm{P}(y_*|x_*)$ to the observation sequence x_*.

In contrast to generative model like Hidden Markov Models (HMMs), the transition between states takes into account the current, past and future observations; the label sequence can depend on non-independent features of the observation. Maximum Entropy Markov Models (MEMMs) and Conditional Random Fields (CRFs) are an example of such conditional discriminative model.

Conditional Random Fields [18] is a framework for labeling and segmenting sequential data based on the conditional model. The CRF classifier takes into consideration the context when assigning label to a particular sequence. Furthermore, a CRF classifier has all the advantages of other preceding conditional classifiers like Maximum Entropy Markov models (MEMMs) and additionally eliminates the *label bias problem*. Details about the *label bias problem* can be found in [18]. Conditionally trained CRFs can easily include large number of arbitrary non independent features [39]. When applying CRFs to the Named Entity Recognition problem, an observation sequence is the token sequence of a sentence or document of text and state sequence is its corresponding label sequence. CRFs use a single exponential model combining the probabilities of the whole sequence of labels based on the observation sequence trained by maximum likelihood. It moves from a current state to the next state having the highest vote based on the observation sequence. More details about CRFs can be found in Lafferty et al. [18].

6.2 Stanford NER

In this research, we used Stanford NER which is an implementation of a Named Entity Recognition system based on the CRF model. Stanford NER can assign labels to a sequence of words falling under one of three categories: PERSON, LOCATION and ORGANIZATION in English language. The provided models are trained using data sets such as CoNLL, MUC-6, MUC-7 and ACE which are in formal English language. The Stanford NER performs a feature selection step before NER and uses IO encoding. Feature selection, IO and IOB encoding are explained next.

Feature Selection

Feature extraction is a preliminary step in sequence labeling. The obvious starting point for feature extraction used in sequence labeling is assigning features to the words. Substrings in words can be used as features such as if a word contains 'san' as a substring at the beginning of the string, it is most likely categorized as place. Features can also be assigned for the previous and next words labels, for example by assuming that any word after 'at' or 'to' is most likely a location. The word features are only looking for observed data. Sequence labeling is only achieved when also looking to the label context. Therefore, we also use context features.
The features used in Stanford NER model are: context features taking into consideration the current, previous and next labels, word shapes based on capitalization and punctuation marks, prefixes and suffixes of words, label sequences and a combination of these.

IO and IOB Encoding

There are two commonly known encoding types for sequence labeling: IO encoding (inside/outside) and IOB (inside, outside and beginning of an entity). The Stanford NER system follows an IO encoding style because it has a faster performance over the IOB encoding style which is crucial for real-time applications. In practice, the systems trained according to IOB encoding usually lead to the same results achieved by IO Encoding NER systems[2]. Table 6.1 shows the differences between these encodings where PER refers to class PERSON, O refers to class OTHER. Additionally, B-PER for "Fred", for example, in IOB encoding denotes that "Fred" is the *beginning* of a token categorized as PERSON. Similarly, I-PER for "Huang" denotes that "Huang" lies *inside* a token categorized as PERSON. As can be seen from Table 6.1 IO encoding can not distinguish between "Sue" and "Mengqui Huang" as being different persons.

6.3 Our Customized NER system

We follow the approach of Liu et al. [19] in applying two steps of NER. In the first step, we use Stanford NER system, and in the second step, we use a dictionary. Our NER system recognizes the following entities from tweets: DIRECTION, HIGHWAY, LOCA-

[2]Stanford NLP online course https://class.coursera.org/nlp/lecture/preview/61

	IO Encoding	IOB Encoding
Fred	PER	B-PER
showed	O	O
Sue	PER	B-PER
Mengqui	PER	B-PER
Huang	PER	I-PER
’s	O	O
new	O	O
painting	O	O

Table 6.1: Sequence label Encoding

TION, PREPOSITION, VOCABULARY. Definitions of these categories can be found in section 6.3.2. Stanford NER is concerned with recognizing the LOCATION class and the dictionary-based step is concerned with recognizing the DIRECTION, HIGHWAY, PREPOSITION, VOCABULARY classes. Details of the two steps are described next.

6.3.1 First Step: Stanford NER

As explained in section 6.2, Stanford NER is able to recognize named entities such as: LOCATION, PERSON, ORGANIZATION. However, based on our research, we are only interested in recognizing the LOCATION class which can be an alley, zone, avenue, boulevard or a landmark. Furthermore, the provided models are trained using formal English language data and do not perform well on unstructured data extracted from Twitter because of the informal language used in tweets. For this reason, we train the model using tweets. A training data source is needed to train the NER model. The training data must contain each word on a separate line and the correct annotation tab separated from it. Another requirement for training the model is a properties file mentioning the features, the location of the training data and a description of the training data. The model is trained once and saved for further use. Each time a tweet arrives, the model is loaded and used to recognize locations in the newly arrived tweet. More details about the training data and system evaluation is in section 8.4.3.

6.3.2 Second Step: Dictionary-based NER

The dictionary is responsible for recognizing the DIRECTION, HIGHWAY, PREPOSITION, VOCABULARY classes. The specially built traffic dictionary contains 855 pairs where each pair contains a token type and a value. All the words are manually extracted from tweets related to traffic in San Francisco bay area. The definitions of token types and the way of extracting them are described as follow:

- Direction: Direction can be one of the following: eastbound, westbound, northbound, southbound, east, west, north, south or combination of these.These directions are extracted from tweets.
- Highway: highways in or passing by San Francisco extracted from California Gazetteer[3] and included in the dictionary.
- Preposition: is a set of words preceding a location and thus marking this location as start or end point depending on the preposition.
- Vocabulary: is a set of words describing Traffic Event extracted from a data set of 2000 tweets,e.g. traffic problem, accidents, road works, incidents. All the words are extracted from tweets.

Due to the ambiguous nature of unstructured data, Tweets may contain entities of equal meaning but in different formats. For example, highway "US-101" can be mentioned as "U.S. 101", "Hwy 101" or "US 101". Furthermore, directions and highways might be written as one word,e.g. S101, N101. Patterns of strings are defined with regular expressions and then are used to match against the input strings to recognize the different forms of writing highways. There are different rules on how to write regular expressions than can be mastered in details in [10]. Examples of regular expressions to recognize highways in our research are:

- [NESW] d d* : this regular expression matches any string that starts with a direction N, E, S or W standing for north, east, south and west respectively followed by one or more digits where '*' signifies the occurrence of 0 or more digits
- [H][W][Y] d d*: is used to match strings starting with the word "hwy" and then followed by a number composed of one or more digits
- [I][-] d d*: is used to match highways starting with the letter "I" and then followed by a number composed of one or more digits, e.g. I-680

[3]California gazetteer http://california.hometownlocator.com/ca/san-francisco/

- [C][A][-] d d*: is used to match strings starting with the word "CA" and then followed by a number composed of one or more digits, e.g. CA-12

The results obtained from Stanford NER system trained with our own data set as well as the results of the second phase of NER and presented in Chapter 8.

6.4 Information Extraction (IE)

Information extraction is the task of extracting structured data from unstructured text. We adopt a template-based information extraction (IE) approach. This approach requires predefined template schemas and labeled data, to learn to extract their slot fillers [2]. Each template has several slots and the information we extract are used to fill these slots. The template extraction algorithm requires full knowledge of the templates and labeled corpora, such as in rule-based systems [5]. Furthermore, template design for information extraction depends on the nature of the task and therefore it affects the success of capturing information from text. The template has to be clear, deterministic, descriptive and monotonic [30].

After annotating *traffic* tweets and recognizing named entities in them, an IE module is responsible for filling the fields of a template for every tweet. By the analysis of *traffic* tweets, it was found that when a traffic jam or an accident is reported at a particular place, people usually mention the location by specifying a start and end point of the traffic jam. Twitter users can also mention a specific landmark or roads intersection which are blocked, this is referred to as "ref-point" in the template. Furthermore, if the traffic information is published by a traffic channel, the tweet usually include the city or county where the incident happens. Additionally, when a highway is mentioned in a tweet, it is accompanied by a direction that is also extracted to fill a slot in the template. The fields of the template are as follows:

- Start point: of the traffic incident.
- End point: of the traffic incident.
- Ref-point(s): One or more specific reference points or crossroads where the incident has occurred. It can also be a landmark.
- City: The city or county that the tweet talks about.

- Highway(s): One or more highways mentioned in the tweet where the incident occurs.
- Direction(s): Direction of blocked or cleared lanes of a highway.
- Request(s): Formulated geocoding request out of the fields mentioned above.

These slots are filled using rules described shortly along with the annotations or token types attached to each token. The slots are optional i.e., not all of the information have to be extracted from a single tweet. There might be zero or more geocoding requests sent to the Geocoding API for a single tweet. No geocoding requests are sent if the information in the template is not sufficient to detect the incident location. On the other hand, if there are several affected location by the traffic information, a separate request is sent to each location to determine its coordinates. We have implemented the following rules based on an analysis of 1000 tweets to fill the previously described template slots:

- Start point: Any token labeled as LOCATION or HIGHWAY and occurs after a "from" or "between" preposition is considered a starting point.
- End point: Any token labeled as LOCATION or HIGHWAY and occurs after "to" or "onto" or the "between ... and" preposition is considered a starting point.
- Ref-point: Any token labeled as LOCATION or HIGHWAY and occurs after these prepositions: after, past, near, at, before, beyond, across
- City: A token labeled as LOCATION that occurs as the first token in the tweet or after "in" preposition.
- Highway and Direction: Any token labeled as HIGHWAY that occurs after "on" preposition or is not preceded by any prepositions is used to fill this slot along with the direction if it's mentioned.
- Request: All the slots contribute to constructing the requests sent to the Geocoding API as follows:
  - If a Ref-point slot and Highway slots are not empty then their values are combined in a string with an "&" in between denoting that we need to find the intersection point between them.
  - The values of From and To slots are sent as separate requests.

  – If the City slot is not empty, its value is appended to each request.

An example of a tweet and the template slots extracted according to the described rules is:

> # SUNOL: Accident NB 680 past Koopman Rd has all lanes blocked as a medivac helicopter is landing on the freeway. Expect delays.

The template filled out of this tweet is as follows:

- Ref-point : KOOPMAN ROAD
- City: SUNOL
- Highway: 680
- Direction: NB
- Request: KOOPMAN ROAD, SUNOL & Interstate 680

The evaluation of the information extraction task is described in details in section 8.4.6.

Chapter 7

Routing

This chapter provides a thorough explanation of the use of graphs for representing roads and points in our research as described in section 8.1, the Dijkstra shortest path algorithm used for routing in section 8.2 and geocoding tools in section 8.3. Our map interface allows the user to click any source and destination points for route planning. Two routes are provided: shortest path if no traffic jam occurs and shortest path avoiding the congestion area. Setting the edge weights for routing is described in section 8.4.

Graphs in computer science are abstract representation that can be used to model any relationship. A Graph $G = (V, E)$ consists of a set of vertices V and a set of edges E connecting the vertices. Graphs are fundamental structures since they can describe any network. Social networks are graphs where nodes represent people and edges represent relationships between them. Road networks are graphs where nodes represent cities and edges represent roads between them.
In our research, a graph G is used to model cities as nodes and routes between them as edges. Graph G has the following characteristics:

- Directed - a graph is said to be *undirected* if an edge $(x, y) \in E$ implies that also an edge $(y, x) \in E$ exists. If that condition does not hold, we say that the graph is *directed.* Since streets within cities have directions, they must be modeled with a directed graph.

- Weighted - Edges and/or vertices might be assigned weights in the graph. In road networks, edges might be assigned values which represent distance, speed limit or other constraints according to the application. In our graph model, weights on edges are function of two values: distance between the two nodes and a weight assigned depending on the traffic level as extracted from Twitter.

- Simple - A graph is considered *simple* if it fulfills two conditions:
  - *no self-edge*: if a node has a an edge to itself i.e $(x, x) \in E$
  - *no multi-edge*: if an edge (x, y) occurs multiple times in the graph.

  Since graph model of road networks avoids these structures, we consider our graph to be *simple*.

- Sparse - Graphs are considered sparse when there exists a small portion of any combination of two vertices having edges between them. A sparse graph is a graph with fewer edges than $|V|^2$. However, there is no strict definition to differentiate sparse and dense graphs. Our graph has 1613303 nodes and 3946702 edges, therefore it is considered a sparse graph.

- Cyclic - A graph that contains cycles. A cycle might exist in a map because a path can begin at a node x and goes through several edges and returns back to the same point.

- Explicit - Some graphs are constructed while we traverse them like backtrack search. These graphs are considered implicit. Otherwise, the graph is called explicit as it is constructed once before the start of any traversal and saved in a data structure. Our graph data structure is explicit

- Labeled - In a labeled graph, each node is assigned a unique identifier to distinguish it from other nodes. The ID can be a name or a numerical value. These identifiers do not exist in unlabeled graphs. Street names can be used as identifier but in our graph model, each node has a unique integer ID.

7.1 Data structure for graphs

Performance is greatly affected by the type of data structure chosen to represent graphs. Data structures of constant time access O(1) should be used whenever possible especially in large graphs. In this section, we describe the data structure used for our graph representation. Assume the graph $G = (V,E)$ contains $|V|$ nodes and $|E|$ edges. Since the number of nodes and edges is previously known and will remain fixed within the run time of the application, arrays are the fundamental data structure used in the graph model for the following reasons:

- Constant time access - Any element in the array can be directly accessed in a constant time given its index. Each array index maps to a memory location. This reduces access time especially in real time applications

- Space efficiency - Arrays do not waste any extra space other than the space allocated to save the data since they do not have any links between the cells which are of fixed size.

- Memory locality - Array records are contiguously allocated in memory which is very suitable for sequential iteration.

Nodes are represented in an array of size V where each node has a unique identifier, longitude and latitude as shown in Table 7.1. The node identifier is used to index the node in the *nodes array.*
Each edge is represented in the *edges array* with the IDs of start and end nodes as can be seen in Table 7.1. A weight is assigned to the edge which is the distance between the two nodes. The array of edges is sorted by the start node ID. Furthermore, an *offset array* is created which carries for each node the start and end indices of its outgoing edges in the *edges array.* If a node does not have any outgoing edges, then it is assigned the same offset assigned to its previous node. The complete data structure of the graph in Figure 7.1 is presented in Table 7.1.

7.2 Dijkstra - Shortest Path Algorithm

We use Dijkstra - one of the shortest paths algorithm - to find routes between departure and destination points. A *path* is defined as a sequence of edges connecting two vertices. A *shortest path* is a path between two vertices which satisfies minimum weight constraint. The cost of a path is equal to the sum of weights of the edges that constitute this path. The edge weight in our graph model is a function of the distance between two nodes and

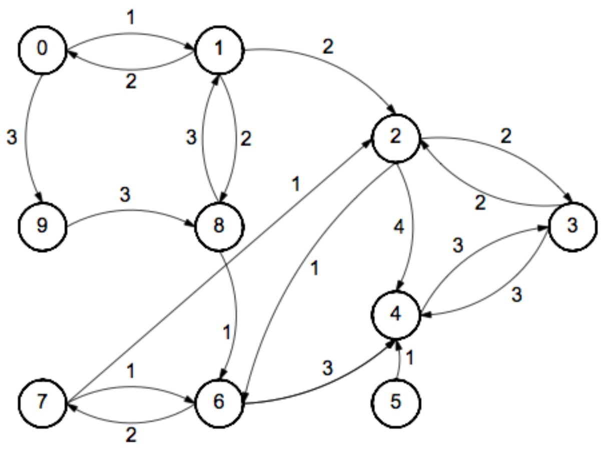


Figure 7.1: Example of a graph

an additional cost set according to the traffic intensity extracted from Twitter.
Dijkstra algorithm finds the path with lowest cost (i.e. the shortest path) between the source vertex s and every other vertex including the destination vertex t. Dijkstra consists of a series of iterations where each iteration finds the shortest path from start node s to a node x where x might be an intermediate node on the path from s to the destination point t. Each found shortest path from s to x must satisfy the minimum weight constraint $dist(s, v_i) + w(v_i, x)$ for all unfinished nodes $1 \leq i \leq n$ where $dist(s, v_i)$ is the length of the shortest path between s and v_i, and $w(v_i, x)$ is the distance (edge length) between v_i and x. The distance from each node to itself is 0. In each round, the algorithm discovers a new vertex for which we know the shortest path from s. Dijkstra pseudo-code [38] is provided below where G denotes the graph, s denotes the start node and t denotes the target node.

```
*ShortestPath-Dijkstra(G,s,t)*
    $known = \{s\}$
    for $i = 1$ to $n$, $dist[i] = \infty$
    for each edge $(s, v)$, $dist[v] = w(s, v)$
    $last = s$
    while $(last \neq t)$
        select $v_{next}$, the unknown vertex minimizing $dist[v]$
        for each edge $(v_{next}, x)$, $dist[x] = \min\ [dist[x], dist[v_{next}] + w(v_{next}, x)]$
        $last = v_{next}$
        $known = known \cup \{v_{next}\}$
```

The data structure described in section 8.1 allows the retrieval of the edges of a particular node in constant time access O(1). The node ID is used to index the *offset array* to find the start and end offsets of its edges from the *edges array*. We use a *parent array* in the implementation of Dijkstra to record the parent/previous of each node during the process of finding the path from s to t. This array is then traversed in a backward manner to find the whole path with all the intermediate edges. The cost of the path can be found as well in *weight array* to record the costs. In each iteration, we use a heap data structure to get the next node with minimum distance to the start node s.

The running time complexity of Dijkstra algorithm is a function of the set of vertices V and the set of edges E. The simplest implementation of the algorithm has the time complexity of $O(|E| + |V|^2)$ since the vertices are stored in a linked list and extracting the minimum will require a linear search over the list to pick the closest node not inserted in the list of *known* vertices yet. However, this run time complexity can be easily improved in the case of a sparse graph. As mentioned earlier, a sparse graph is a graph with fewer edges than $|V|^2$. With our graph representation, the set of known edges are stored in a heap and therefore extracting the node with minimum distance to the source node s is efficient. The run time in this case is $O(|E| + |V| \log |V|)$. More details about the algorithm can be found in Skiena [38].

7.3 Geocoding

Geocoding is the process used to convert textual addresses into geographical coordinates consisting of longitude and latitude. The Geocoding process is the link between the information extracted from Tweets and markers on a map.
Google offers a Geocoding service via HTTP. Any HTTP request issued to the Geocoding API must be in the following form

$$http://maps.googleapis.com/maps/api/geocode/output?parameters$$

Geocoding Request Parameters

- `address` (Required), the address in text form that needs to be geocoded. The addresses to be geocoded are formulated from the fields in the template after the information extraction phase and used to fill the *request* field as described in section 6.4.

- `sensor` (Required), a boolean value to indicate whether the request comes from a device with GPS enabled. This parameter is always set to false in all the requests our application is issuing.

- `bounds` (Optional) A filtering criteria used to limit the results to be within a particular bounding box specified by two geographical coordinates pairs; North East longitude and latitude and South West longitude and latitude. The bounding box coordinates of San Francisco are (-122.75, 36.8) and (-121.75, 37.8) respectively.

- `language` (Optional) the language of the returned HTTP responses. In our customized requests, it is set to "en" for english.

- `region` (Optional) the region code represented in top level domain form which in our case is set to "US". It limits the response as the `bounds` parameter.

Geocoding Response The output can be provided either in JSON or XML format which has to be specified in the HTTP request form. The JSON output has two root elements:

- `status` contains all metadata associated with the request

- `results` contains the full address including street name, zip code, city and country, and the geographical coordinates of the requested address. More than one result can be retured by the service if the requested address is ambiguous.

7.4 Setting Edge Weights

After getting the geographical coordinates of a traffic incident from the Geocoding API, we need to add a penalty over all the edges passing by the node where a traffic incident occurs. To accomplish this task, a mapping is needed between the longitude-latitude pair and the ID of the affected node. For this reason, we overlay the map with a grid of 5137x4735 cells, where each grid cell covers an area of 200m to 300m and contains all the IDs of the nodes within this range. This grid is indexed with the longitude-latitude pair. An additional cost is added on all the edges starting from the node IDs in this cell as well as all the nodes in the surrounding cells.

Index = node ID	Longitude	Latitude
0	36.1	-122.1
1	36.2	-122.2
2	36.3	-122.3
3	36.4	-122.4
4	36.5	-122.5
5	36.6	-122.6
6	36.7	-122.7
7	36.8	-122.8
8	36.9	-122.9
9	36.11	-122.11

(a) Nodes Array

index = node ID	start offset
0	0
1	2
2	5
3	8
4	10
5	11
6	12
7	14
8	16
9	18

(b) Offset Array

Index	Start node ID	End Node ID	Weight (distance)
0	0	1	1
1	0	9	3
2	1	0	2
3	1	2	2
4	1	8	2
5	2	3	2
6	2	4	4
7	2	6	1
8	3	2	2
9	3	4	3
10	4	3	3
11	5	4	1
12	6	4	3
13	6	7	2
14	7	2	2
15	7	6	1
16	8	1	3
17	8	6	1
18	9	8	3

(c) Edges Array

Table 7.1: Data Structure for the Graph in Figure 7.1

Chapter 8

Experiments and Results

8.1 Evaluation Metrics

The evaluation metrics used in our experiments are Precision, Recall, Accuracy, F-Measure which are the widely common metrics used in natural language processing tasks. To compute these measures, it is essential to create a 2-by-2 contingency table. The table contains two axes, truth axis and the system to be evaluated axis. The table contains the four states to be found in any data sets subject to analysis. The four states are :

- True positive: means that the piece of data under analysis is true and the system said it is true.

- False positive: means that the system mistakenly treated it as positive although it is not.

- False negative: means that the system said it is not true although it is, it is treated as negative falsely.

- True negative: means that the system correctly said it is negative and it is negative.

	Correct	Not correct
selected	true positive(tp)	false positive(fp)
not selected	false negative(fn)	true negative(tn)

Table 8.1: 2-by-2 contingency table

According to the table, *accuracy* is defined as the percentage of correct information the system recognizes.

$$Accuracy = \frac{tp + tn}{tp + fp + tn + fn}$$

However, accuracy is not enough to evaluate any systems especially if what we are looking for is uncommon. For example, in the case of classifying incoming tweets to tweets related to traffic or not, a very high accuracy can be achieved simply by classifying each tweet as non-traffic. Consequently, two other measures, precision and recall, are used to evaluate these infrequent cases that need to be detected.

- *Precision* is defined as the fraction of the results returned by the system relevant to the information need

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

  In contrast,

- *Recall* is the fraction of relevant or correct items in the collection that are returned by the system. A system with 0% is not interesting as it finds none of the information we are looking for.

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

There is always a trade-off between *precision* and *recall* as an improvement in one of these metrics will result in the decay of the other and vice versa. Therefore, it should be decided and discussed beforehand which value is most important depending on the application. In various applications such as legal applications where there is a need to find all evidences, a system with high recall is the required system. However, in other applications, it might be more useful to show the correct information which can be achieved with a high precision system. That explicit trade-off is a useful concept in building NLP systems.

However, a problem arises when having multiple systems and there is a need to decide which one is better. It is hard to assess the best system by having to compare *precision* and *recall*. Therefore, *F-measure* is a combined measure that assesses the P/R tradeoff in a weighted harmonic mean. Weights are assigned to the values involved in the mean according to the need to boost or decrease them depending on the application.

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

where P is *precision*, R is *recall* and α is a weight and β is a control parameter which is the emphasis imposed on precision versus recall.
A balanced *F-measure* is used by setting *alpha* to $\frac{1}{2}$ or *beta* to 1 resulting in the following equation:

$$F_1 = \frac{2PR}{P+R}$$

However, F_1 is a measure for the performance of the classifier in one class. But we also want a single number that measures the aggregate performance over all classes in the collection. For this reason we use Macroaveraging to give equal weights to the classes by computing F_1 for each of the C classes and average these C numbers.

8.2 Relevance Classification Evaluation

In this type of classification, we classify each incoming tweet as *traffic* or *junk* as described in section 5.6.3.
Example of traffic tweet:

> San Lorenzo: S880 after Hesperian, a load of lumber blocks the right lane. Traffic backs into San Leandro. 238 backs to Castro Valley.

Example of Junk tweet:

> RT @YouTube: Astronaut Neil Armstrong, the first man to walk on the moon, leaves behind a legacy at age 82

8.2.1 Data Preparation

Tweets are retrieved from Twitter through the streaming API. The training data and the testing data is common for all classifiers used to allow a fair evaluation. A set of 3188 *traffic* tweets are collected during May 2012 from different Twitter users and traffic channel accounts. The full list of accounts can be found in Appendix A.4. A second set of 3333 *Junk* tweets are obtained consecutively from Twitter public timeline and manually checked to belong to *junk* class. These two sets are divided randomly into training and test data. The training data consists of 2000 tweets manually classified as *traffic* and 2000 as *junk*. The test data consists of 2521 manually annotated tweets. 1188 tweets belong to *traffic* class, 1333 tweets belong to *junk* class. The annotations are performed by the author and a computational linguistics student. The metrics used are accuracy, precision, recall and F_1-measure.

8.2.2 Naive Bayes Classifier

MALLET[1] (MAchine Learning for LanguagE Toolkit) is a Java based package providing implementation of various algorithms used in NLP. It contains several tools for document classification such as Naive Bayes and Maximum Entropy as well as tools for classifier evaluation using the above mentioned metrics. The library is open source and published under Common Public License. In our task of text classification, a tweet represents a document. A model is trained with the training data set once and saved. It is then loaded for each classification to allow the functionality of the system in real-time. Precision, recall and F_1-measure for the relevance classification are shown in Table 8.2.

8.2.3 Maximum Entropy Classifier

Maximum Entropy Classifier included in MALLET[1] package is used in our implementation of relevance classification module. The classifier is trained with the train instances once and used repeatedly. Evaluation results are shown in Table 8.2.

8.2.4 SVMs

SVMlight[2] is an implementation of Support Vector Machines (SVMs) in C. As described from the library website

> The main features of this library are:
>
> - fast optimization algorithm
> - use of folding in the linear case
> - handles many thousands of support vectors
> - uses sparse vector representation

[1]http://mallet.cs.umass.edu/
[2]http://svmlight.joachims.org/

However, since Java is the programming language used in our application, we used JNISVM-light-6.01[3]. It is a Java Native Interface that includes a pre-compiled shared libraries for windows and Linux and supports text classification as well as other functionalities. The evaluation results are shown in Table 8.2.

8.2.5 Voting System

A voting system that combines the three classifiers is then used as described in section 5.5. The system results are shown in Table 8.2.

8.2.6 Results

As can be seen in Table 8.2, the highest precision, recall and F_1-measure for each of the classes of *traffic* and *junk* are marked in bold. The voting system led to the best results, hence we use this combined system in our relevance classification task.

Classifier	Class	Precision	Recall	F_1	Macro-Averaged F_1	Accuracy
Naive Bayes	Traffic	0.825	**1.0**	0.9041	0.8998	0.900
	Junk	**1.0**	0.81095	0.8956		
Maxent	Traffic	0.9948	0.6540	0.7892	0.8270	0.8353
	Junk	0.7637	0.9969	0.8649		
SVM	Traffic	0.9957	0.9898	0.9926	0.9930	0.9932
	Junk	0.9910	0.9962	0.9935		
Voting System	Traffic	**0.9974**	0.9915	**0.9944**	**0.9947**	**0.9948**
	Junk	0.9925	**0.9977**	**0.9950**		

Table 8.2: Text classification results for *traffic* and *junk* classes

8.3 Incident Classification Evaluation

We further make a distinction between *positive traffic* tweets and *negative traffic* tweets:

- Positive: A tweet containing information about open lanes and clear traffic e.g.

Update: All lanes are now clear on South US-101 in San Francisco.

[3]http://www.mpi-inf.mpg.de/mtb/

- Negative: A tweet announcing information about incidents, blocked lanes or road construction e.g.

Accident blocks lanes of westbound CA-12 after Kelly Rd in Napa.

8.3.1 Data preparation

The training data and the testing data is common for all classifiers used to allow a fair evaluation. The tweets in these data set are the same *traffic* tweets collected for the relevance classification explained in section 8.2.1. The training data consists of 2500 tweets manually annotated. 2456 are manually classified as *Negative* and 544 as *Positive* as explained in section 5.6.4. The percentage of positive tweets is much lower than negative tweets because people usually tweet or complain when they are stuck in traffic. The test data consists of 500 manually annotated tweets. 198 tweets belong to *Positive* class, 302 tweets belong to *Negative* class. The annotations are performed by the author and a computational linguistics student.

8.3.2 Classifiers

We use the Naive Bayes, Maximum Entropy and SMV classifiers provided by the same libraries used in relevance classification module as described in section 8.2.
Each of the results of the three used classifiers are presented in Table 8.3.

8.3.3 Results

The results of Naive Bayes and Maxent classifiers were not very promising. This may be due to the unbalanced data sets of the two classes. SVM outperforms the other classifiers and for this reason we decided to use SVMs for this type of classification. The highest precision, recall, F_1-measure of each class *positive* and *negative* are marked in **bold** as can be seen in Table 8.3.

8.4 Named Entity Recognition Evaluation

Our NER system consists of two phases:

Classifier	Class	Precision	Recall	F_1	Macro-Averaged F_1	Accuracy
Naive Bayes	Positive	0.5217	0.1655	0.2513	0.5034	0.6314
	Negative	0.6461	0.9094	0.7555		
Maxent	Positive	0.3	0.6206	0.1028	0.4207	0.5953
	Negative	0.6201	0.9135	0.7387		
SVM	Positive	**0.9619**	**0.8939**	**0.9266**	**0.9406**	**0.9440**
	Negative	**0.9335**	**0.9768**	**0.9546**		

Table 8.3: Text classification results for *Positive* and *Negative* classes

- First phase: we use Stanford NER system to recognize LOCATION class
- Second phase: we created our own dictionary that can be found in appendix A.2 to recognize HIGHWAY, DIRECTION, PREPOSITION and VOCABULARY classes

The named entities are described in section 6.3.

8.4.1 Data Preparation

The data set consists of the 3188 tweets used in the relevance classification module and manually annotated to be tweets related to traffic in San Francisco area as explained in section 8.2.1. The same 2000 tweets used to train the text classifiers (Naive Bayes, Maxent and SVMs) are used to train the NER model. A tokenizer is used to break the tweets into tokens with each token on a separate line. The training data contains 27780 tokens manually annotated with named entity classes. All the annotations for the training and test data sets are performed manually by the author and a computational linguistics student. Each token is annotated with one of the following named-entity labels: LOCATION, HIGHWAY, DIRECTION, PREPOSITION, VOCABULARY or OTHER. The definitions of these categories can be found in section 6.3.1.
We test the system with two data sets:

- The initial test set consisting of 1188 *traffic* tweets used in relevance classification module. This data set consists of 16725 manually annotated tokens.
- 1174 tweets out of the initial test set of 1188 *traffic* tweets. The 1174 are the tweets correctly classified as *traffic* by the voting system as described in section 8.2.5. This data set consists of 16555 manually annotated tokens.

We test our system with these two test sets because the output of each module is the input of its successor and there is a need to get a cumulative final result.

8.4.2 Baseline

The originally trained Stanford CRF model for CoNLL and MUC datasets on English language performs very poorly on tweets due to the unstructured nature of tweets data. Therefore we train the system with tweets. The results of the provided models by Stanford system when tested on tweets are shown in Table 8.4.

Entity	Precision	Recall	F_1
LOCATION	0.5840	0.2126	0.3118

Table 8.4: Stanford NER Evaluation Metrics results trained using MUC and CoNLL data sets

8.4.3 First Phase: Stanford NER

Stanford NER is based on CRFs as described in section 6.1. To train the Stanford CRF classifier, the data is tokenized, each line has one token tab separated from its annotation. The classifier requires a properties file that mentions the location of the train file, the train data description e.g it contains two columns, the first column is the word and the second is its class. We use the embedded feature extraction methods used by default in the classifier, such as words feature, words shapes and label sequences. The Stanford NER classifier tags the tokens with a rate of 6827.24 tokens per second. We use the Stanford NER to only recognize the LOCATION class. Therefore, the tokens in the training data are manually annotated with either LOCATION or OTHER. The first and second test data sets evaluation results are presented in Table 8.5 and Table 8.6 respectively.

8.4.4 Second Phase: Dictionary-based NER

The second phase of NER module is responsible for recognizing other named entities related to the application such as DIRECTION, HIGHWAY, PREPOSITION, VOCABULARY as described in details in section 6.3.2. The list of words available in our dictionary is shown in appendix A.2. Example of a tweet after being processed by the NER module can be found in Table 8.8.

8.4.5 Results

Table 8.5 shows the evaluation results of Stanford NER system trained with tweets and tested of the first data set (1188 tweets). The performance highly improved compared to the results of the baseline shown in Table 8.4. The second test data set evaluation results

Entity	Precision	Recall	F_1
LOCATION	0.9701	0.9325	0.9510

Table 8.5: Results of Stanford NER system when tested with the original test set consisting of 1188 *traffic* tweets

are shown in Table 8.6. Precision slightly reduced.

Entity	Precision	Recall	F_1
LOCATION	0.9600	0.9325	0.9508

Table 8.6: Results of Stanford NER system when tested on 1174 (out of 1188) tweets that were correctly classified as *traffic* by the relevance classification module

Example of NER for a Tweet San Mateo Bridge: W92 before the high rise, accident blocks the left lane.

Table 8.7 shows the annotated tweet tokens after the first phase of NER. All the tokens

Token Value	Token Type
SAN	LOCATION
MATEO	LOCATION
BRIDGE	LOCATION
W92	OTHER
BEFORE	OTHER
HIGH	LOCATION
RISE	LOCATION
ACCIDENT	OTHER
BLOCKS	OTHER
LEFT	OTHER
LANE	OTHER

Table 8.7: Stanford NER Results for the tweet example

are annotated with their named entity classes after the two phases of NER as shown in Table 8.8. A postprocessing step is needed to capture entities composed of more than one

Token Value	Token Type
SAN	LOCATION
MATEO	LOCATION
BRIDGE	LOCATION
W	DIRECTION
92	HIGHWAY
BEFORE	PREPOSITION
HIGH	LOCATION
RISE	LOCATION
ACCIDENT	VOC
BLOCKS	VOC
LEFT	VOC
LANE	VOC

Table 8.8: Second phase of NER Results for the tweet example

word, for example "San Francisco". IO encoding is applied so any two or more consecutive words of the same named entity type or token type are combined into one token having the respective word and assigned the same token type. The results for the example tweet are shown in Table 8.9.

Token Value	Token Type
SAN MATEO BRIDGE	LOCATION
W	DIRECTION
92	HIGHWAY
BEFORE	OTHER
HIGH RISE	LOCATION
ACCIDENT	VOC
BLOCKS	VOC
LEFT	VOC
LANE	VOC

Table 8.9: Postprocessing of the second phase NER Results for the tweet example

8.4.6 Information Extraction (IE)

In this section we describe our procedure to evaluate the IE module and show the results.

Data Preparation

A test set consisting of 1000 *traffic* tweets is used to evaluate the information extraction. The template explained in Section 6.4 has the following slots: start point, end point, ref-point(s), city, highway(s), direction(s) and request(s). Each tweet is used to fill the slots of a template manually and then by our hybrid approach of rule-based IE and NER module. The manual filling of the template are performed by the author and a computer science student.

Results

We built a two by two contingency table for every slot in the template and then we calculated the precision, recall and F_1 measure shown in Table 8.10.

A tweet might result in several requests if it contains multiple locations. Based on

Template slot	Precision	Recall	F_1
Start Point	0.975	0.975	0.975
End Point	0.956	0.965	0.960
Ref-point	0.974	0.916	0.944
Highway	0.985	0.983	0.983
Direction	0.995	0.990	0.992
County	0.984	0.988	0.985

Table 8.10: Evaluation results of template slots

our data set of 1000 tweets, 1778 requests should be extracted. Table 8.11 shows our results for false positives, false negatives, true positives and true negatives. False positive refers to falsely formulated requests, true positive refers to correctly formulated requests, false negative refers to missing requests and true negative denotes that no requests were extracted because the tweet does not contain enough information.

	Correct Request	Incorrect Request
selected	1533(tp)	135(fp)
not selected	59(fn)	51 (tn)

Table 8.11: 2-by-2 contingency table for the *Request* slot

We consider the precision and recall of *Request* slot as the output of the IE module. Table

8.12 shows the precision, recall and F_1 of the *Request* slot based on the the contingency table in 8.11. These values are considered the evaluation results for the IE module as shown in Table 8.12.

Module	Precision	Recall	F_1
IE	0.9190	0.9629	0.9404

Table 8.12: IE module results

8.5 Routing

In this section, we analyze a tweet processed by our system and show the results of routing before and during a traffic jam.

8.5.1 Geocoding

We use the following tweet as an example for showing the extracted locations and the requests sent to the Geocoding API as shown in Table 8.13.

Tweet #Oakland occupy protestors have broadway completely blocked on BRDWY BOTH NB/SB at 14TH ST #traffic http://t.co/iCaIKcDy

The returned responses are plotted on the map by placing markers as shown in figure 8.1.

Request	Response
14TH Street, Oakland, CA, USA	(37.809835,-122.287618)
Broadway, Oakland, CA, USA	(37.827637,-122.256554)

Table 8.13: Requests and responses of a tweet

8.5.2 Visualizing the extracted locations

JXMapViewer is used to place markers on the map depending on the geographical locations consisting of longitude and latitude returned from the Geocoding API. A tweet may result in several markers plotted on the map depending on the amount of information extracted.

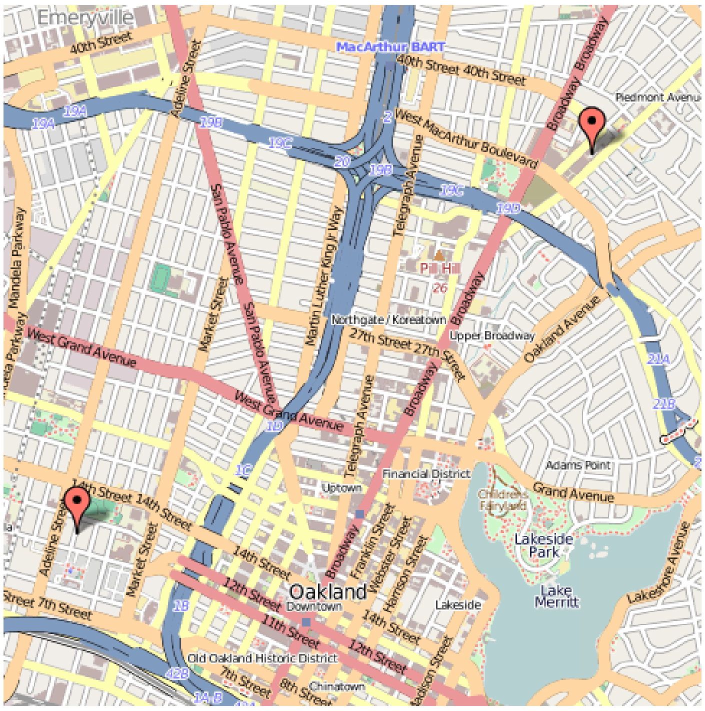
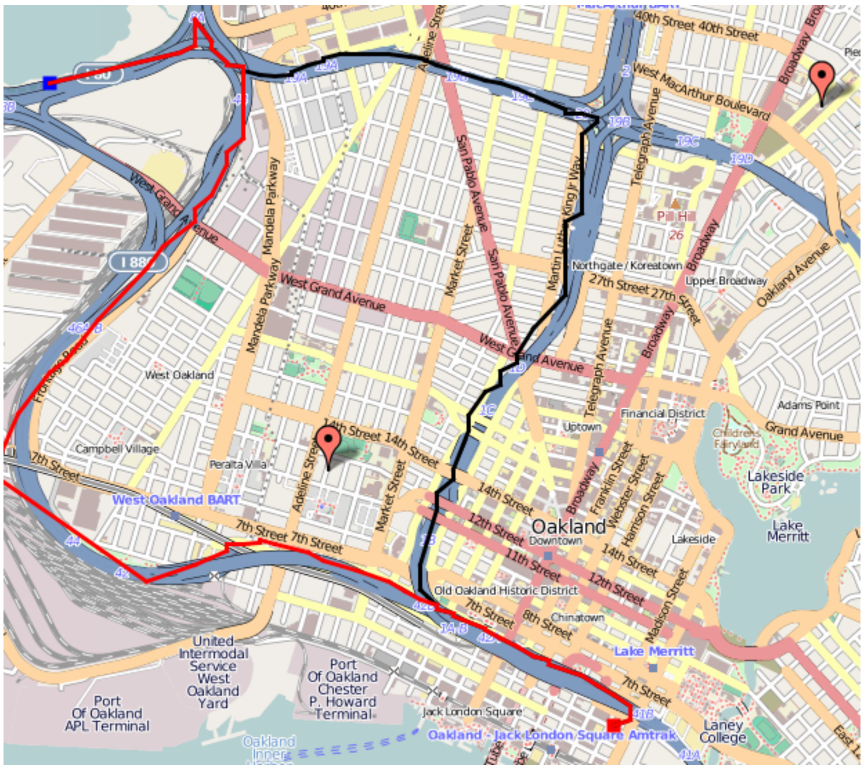


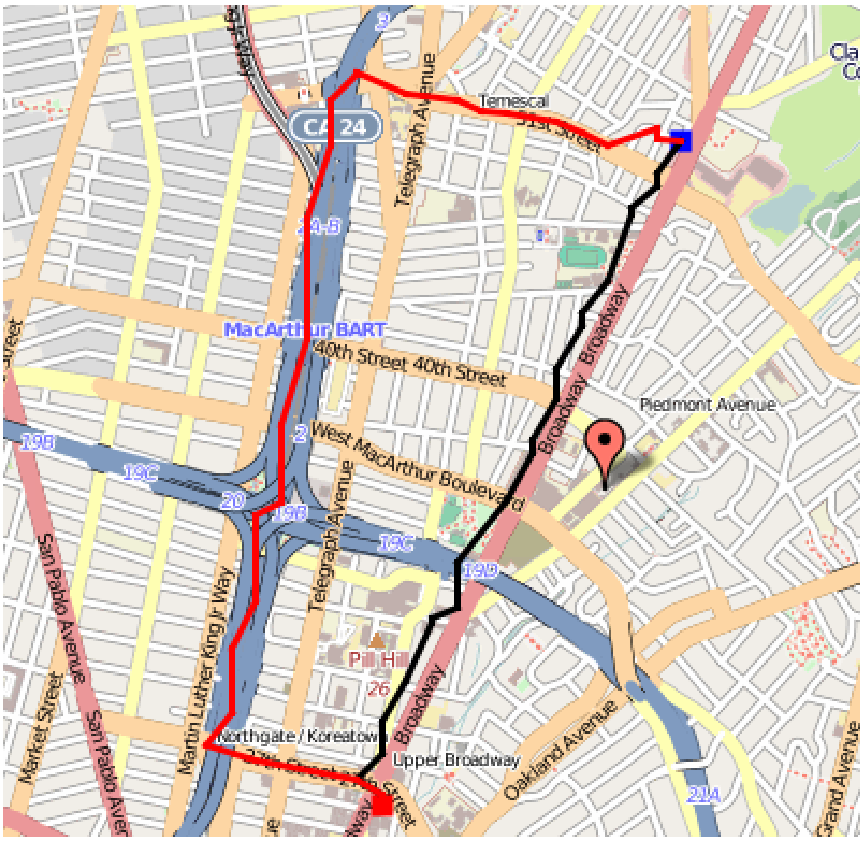
Figure 8.1: Markers at locations returned by the Geocoding API

8.5.3 Results

Figure 8.2 (a) shows the route between two points passing by "14th Street, Oakland" before any incident has been taken into consideration and after the traffic jam occurred. Our implementation of Dijkstra, shortest path algorithm, resulted in a path length of 28826m and 166 nodes shown by the black line. After penalizing the edges around "14th Street", the route length between the source and the destination is 29664m and the path includes 151 nodes. Figure 8.2 (b) shows the routes when considering the incident at "Broadway".



(a) Incident at 14th Street



(b) Incident at Broadway

Figure 8.2: The red map markers are placed at center of the locations extracted from the tweet where traffic problems exist. The blue dot denotes the start point of a route and the red dot denotes the destination point. The black line refers to the shortest path between the start and the destinations points if there is no traffic problems. The red line denotes the alternative route suggested after considering the traffic jams happening at the red map markers.

8.6 System Performance On Selected Tweets

In this section, we select 10 tweets and show the results of processing them through all modules of our system. Section 8.6.1 shows results of 3 tweets that were correctly classified as *traffic* by our system (true positives). Section 8.6.2 shows examples of 2 tweets that were falsely classified as *junk* by our system although they are *traffic* tweets (false negatives). Section 8.6.3 shows examples of 3 tweets correctly classified as *junk* (true negatives). Finally, section 8.6.4 shows examples of 2 tweets that were falsely classified as *traffic* although they were *junk* tweets (false positives). We create a table for each tweet that carries the results of all system modules and then provide a screenshot from our application for visualizing the tweets. Each table has the following entries:

- Relevance classification result: *traffic* or *junk*.
- Incident classification result: *positive* or *negative*.
- NER module results with each token annotated with its entity class.
- IE results with each slot in the template filled or none if no information is extracted for the slot.
- Geocoding requests and results.

8.6.1 True Positives

Table 8.14 shows the results of a *traffic positive* tweet where all the information has been correctly extracted. A screenshot of the locations can be found in Figure 8.3.

Tweet#1 All Lanes are Open on CA-12 in Sonoma County.

Text Classification	**Relevance Classification**	**Incident Classification**
	Traffic	Positive
NER	**Token Value**	**Token Type**
	LANES	VOC
	OPEN	VOC
	ON	PREP
	CA-12	HIGHWAY
	IN	PREP
	SONOMA COUNTY	LOCATION
Template	**Slot**	**Value**
	Start point	None
	End point	None
	Ref-point	None
	Highway	CALIFORNIA 12
	Direction	None
	City	SONOMA COUNTY
Geocoding	**Requests**	**Responses**
	California 12, Sonoma County, CA, US	38.39860560, -122.53740210

Table 8.14: Results of tweet#1 after passing through all system modules

Table 8.15 shows the results of a *traffic negative* tweet where all the information has been correctly extracted. A screenshot of the locations can be found in Figure 8.3.

Tweet#2 Cupertino accident SB 280 at Hwy 85 blocking 3-right lanes, only left lane open. Traffic slow from Foothill Expwy.

Text Classification	**Relevance Classification**	**Incident Classification**
	Traffic	Negative
NER	**Token Value**	**Token Type**
	CUPERTINO	LOCATION
	ACCIDENT	VOC
	SB	DIRECTION
	280	HIGHWAY
	AT	PREP
	HWY 85	HIGHWAY
	3-RIGHT	OTHER
	LANES	VOC
	LEFT	VOC
	LANE	VOC
	OPEN	VOC
	TRAFFIC	VOC
	SLOW	VOC
	FROM	PREP
	FOOTHILL EXPWY	LOCATION
Template	**Slot**	**Value**
	Start point	FOOTHILL EXPWY
	End point	None
	Ref-point	CALIFORNIA 85
	Highway	INTERSTATE 280
	Direction	SB
	City	CUPERTINO
Geocoding	**Requests**	**Responses**
	California 85, Cupertino & Interstate 280	37.33119090, -122.05587250
	Foothill Expwy, Cupertino, CA ,USA	37.33559680, -122.06820270

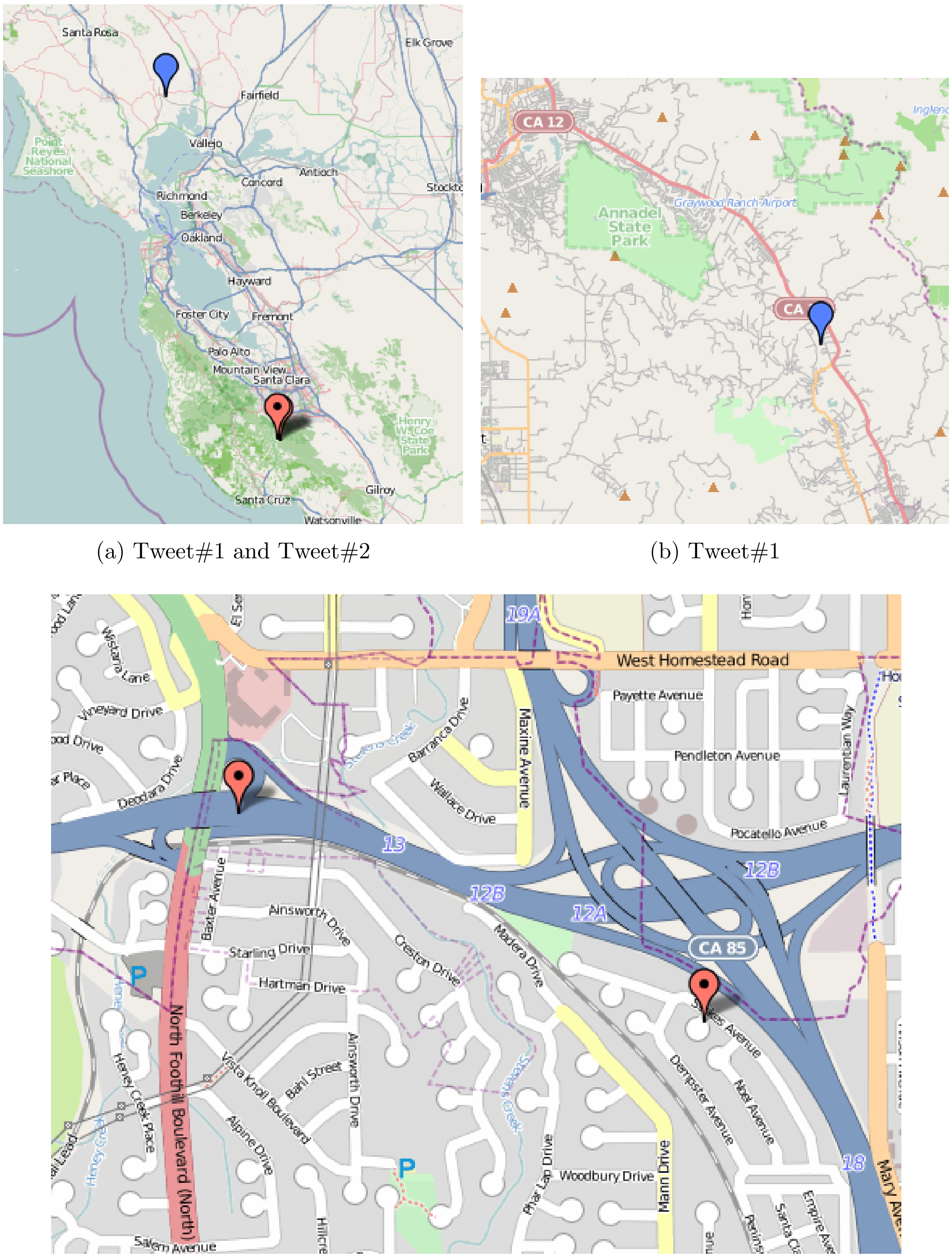
Table 8.15: Results of tweet#2 after passing through all system modules

Table 8.16 shows the results of a *traffic* tweet but does not mention useful information about traffic status. NER module falsely classified one token as LOCATION although it is OTHER (marked in **bold** in Table 8.16). Consequently, the "city" slot in the template was incorrectly filled. However, no requests have been sent to the Geocoding API because most of the template slots are empty.

Tweet#3 Hella traffic but I'm in the carpool lane tho!!! #winning #Hipstamatic #LuciferVI #BlankoFreedom13

Text Classification	**Relevance Classification**	**Incident Classification**
	Traffic	Negative
NER	**Token Value**	**Token Type**
	HELLA	LOCATION
	TRAFFIC	VOC
	I	OTHER
	M	OTHER
	IN	PREP
	CARPOOL	OTHER
	LANE	VOC
	THO	OTHER
	WINNING	OTHER
	HIPSTAMATIC	OTHER
	LUCIFERVI	OTHER
	BLANKTOFREEDOM13	OTHER
Template	**Slot**	**Value**
	Start point	None
	End point	None
	Ref-point	None
	Highway	None
	Direction	None
	City	**HELLA**
Geocoding	**Requests**	**Responses**
	None	None

Table 8.16: Results of tweet#3 after passing through all system modules



(a) Tweet#1 and Tweet#2

(b) Tweet#1

(c) Tweet#2

Figure 8.3: Visualization of selected tweets. Red marker denotes *negative* tweet, blue marker denotes *positive* tweet

8.6.2 False Negatives

The following tweets are falsely classified as *junk* although they are *traffic*:

- The bag of words classifier fails in Tweet#4 because the words are not strong cues for traffic.
  Tweet#4 San Mateo Bridge: still a decent alternative. WB slows Hayward side but not stop and go, yet. Easy ride rest of the way to Peninsula.

- The reason behind Tweet#5 being treated as *junk* falsely might be due to words like "roadway" and "guardrail" that do not occur in the training data set and hence not considered traffic signals.
  Tweet#5 Hercules: EB 4 at Franklin Cnyn Golf Course; overturned pickup truck with injuries thru the guardrail and off the roadway

However, the percentage of false negatives tweets in our system is very small according to our test data set described in section 8.2.1. Our test data set for *traffic* class consists of 1188 *traffic* tweets and our system classified correctly 1174 tweets out of them. According to this data set, 14 out of 1188 tweets are false negatives.

8.6.3 True Negatives

The following tweets are example tweets that are correctly classified as *junk*:
Tweet#6 I recently saved a ton of money on my car insurance by fleeing the scene of the accident.
Tweet#7 I'm at City of Livermore (Livermore, CA)
Tweet#8 #Oakland The current time in Oakland is 7:45 AM on Tuesday, 25 September 2012

8.6.4 False Positives

In this section, we present two tweets that are falsely classified by our system as *traffic* although they are *junk*. In Table 8.17, NER module correctly recognized the named entities. Consequently, the "ref-point" slot in the template has been filled. However, since the template had only the "ref-point" slot filled, no requests have been sent to the geocoding API and no false information were provided to the user of the system.

Tweet#9 Never on time, always late. On the bay bridge or the golden gate.

Text Classification	**Relevance Classification**	**Incident Classification**
	Traffic	Negative
NER	**Token Value**	**Token Type**
	NEVER	OTHER
	ON	PREP
	TIME	OTHER
	ALWAYS	OTHER
	LATE	OTHER
	ON	PREP
	BAY BRIDGE GOLDEN GATE	LOCATION
Template	**Slot**	**Value**
	Start point	None
	End point	None
	Ref-point	BAY BRIDGE GOLDEN GATE
	Highway	None
	Direction	None
	City	None
Geocoding	**Requests**	**Responses**
	None	None

Table 8.17: Results of tweet#9 after passing through all system modules

Table 8.18 shows the results of a tweet falsely classified as *traffic*. The tweet did not contain any locations, highways or directions so all the template slots were empty and consequently no false information was provided to the system user.

Finally, we can assume that even when tweets are falsely classified as *traffic*, if the majority of the template slots are empty then we do not visualize them and hence do not provide false traffic information.

Tweet#10 On a day when you delay me and have broken elevators, you check that I've paid for this service...

Text Classification	**Relevance Classification**	**Incident Classification**
	Traffic	Negative
NER	**Token Value**	**Token Type**
	ON	PREP
	DAY	OTHER
	DELAY	OTHER
	AND	OTHER
	BROKEN	OTHER
	ELEVATORS	OTHER
	CHECK	OTHER
	I	OTHER
	'VE	OTHER
	PAID	OTHER
	FOR	OTHER
	SERVICE	OTHER
Template	**Slot**	**Value**
	Start point	None
	End point	None
	Ref-point	None
	Highway	None
	Direction	None
	City	None
Geocoding	**Requests**	**Responses**
	None	None

Table 8.18: Results of tweet#10 after passing through all system modules

Chapter 9

Conclusion

In this thesis, we proposed a system that leverages Twitter to extract temporary traffic information and use it for route planning. The system is configured to work for the San Francisco Bay Area. It consists of several modules where the output of each module is the input of its successor module.

The first module is the interface with Twitter, responsible for retrieving tweets published on the public timeline. Tweets are tokenized and then a preprocessor module removes stopwords and URLs. The next module is concerned with text classification. First, we classify each incoming tweet into either *traffic* or *junk*. We adopted three widely known algorithms in text classification: Naive Bayes, Maximum Entropy and SVMs. One of the main challenges was to construct a gold dataset necessary for training and testing the text classifiers. The best result was obtained when combining the three classifiers in a voting system achieving an F_1-measure of 99.4% on our test set. Additionally, we classify each *traffic* tweet into *positive* or *negative*. SVMs outperformed Naive Bayes and Maxent in this classification type achieving an F_1-measure of 94.06%

Next is the NER module. The NER module consists of two steps, the Stanford NER system trained on annotated *traffic* tweets responsible for recognizing *locations* in *traffic* tweets and a dictionary-based step by the means of words extracted from tweets related to traffic in the San Francisco Bay Area for recognizing *highways, directions, prepositions and vocabulary* related to traffic. We tested the NER system with the tweets resulting from the text classification module and achieved an F_1-measure of 95.08%. A template is then filled for each tweet using the extracted information achieving an F_1-measure of 94.04%. Combining NER and IE resulted in an F_1-measure of 94.54%.

Mapping the extracted locations into geographical coordinates is the role of the Geocoding module. Our system prototype uses the Google Geocoding API.

The last module is the routing. We created our own data structure to represent California. It is a graph consisting of vertices V and edges E connecting them. Each edge has a weight equal to the distance between the two points it is connecting. We implemented Dijkstra to obtain the shortest path between two points. We created a new weight function which takes as parameters the distance between two points and a weight assigned to the edge based on the status of the traffic extracted from Twitter. All locations where incidents have been recorded are plotted on a map by placing markers at the specified coordinates using JXMapViewer.

Future Work

Probing deeper, the results in this thesis provide a good foundation for future work. A challenging task would be to build the dictionary in an automated manner instead of our manual approach. This will allow the IE module to be ported to other areas and other languages.

Also, it would be interesting to obtain larger data sets to train and test the classifiers that might lead to better precision and recall values and more generalized tasks.

In text classification module, our voting scheme that combines three classifiers (Naive Bayes, Maximum Entropy and SVMs) is very simple and can be extended into a weighted voting system.

Our system is developed to work on one machine. The system can be extended to work on a cluster for a better real-time performance.

The system can also be integrated with TourenPlaner[1] developed by FMI department at University of Stuttgart.

[1]http://tourenplaner.informatik.uni-stuttgart.de/

Appendix A

Appendix

A.1 List of Stopwords

a	about	add	ago	all	also	an
another	any	are	as	be	because	been
being	big	both	but	by	came	can
come	could	did	do	does	due	each
else	end	far	few	get	got	had
has	have	he	her	here	him	himself
his	how	if	is	it	its	just
let	lie	like	low	make	many	me
might	more	most	much	must	my	never
no	nor	not	off	old	only	or
other	our	out	over	per	put	said
same	see	she	should	since	so	some
still	such	take	than	that	the	their
them	then	there	these	they	this	those
through	too	under	up	use	very	want
was	we	well	were	what	when	where
which	while	who	why	will	with	would
yes	yet	you	your			

Table A.1: Stopwords List

A.2 Dictionary

A.2.1 Prepositions

across	after	at	before	between	beyond	from
in	near	of	on	onto	past	to

Table A.2: Prepositions List

A.2.2 Vocabulary related to traffic

accident	avoid	block	car	cement
clear	close	collision	crash	debris
delay	direction	disable	expect	injury
incident	lane	left	light	metering
middle	now	object	open	right
road	shoulder	slow	tractor	traffic
truck	vehicle			

Table A.3: Vocabulary List

A.3 Pruned Punctuation marks

?	.	:	;	(
)	[	]	_	

Table A.4: List of pruned punctuation marks

A.4 Twitter Request parameters

- `track`: A list of the words available in the dictionary presented in Appendix 1.2
- `follow`: A list of user ids to follow.
  - KCBS Traffic @kcbstraffic

  - 511 Bay Area @511SFBay
  - 511 Rideshare @511Rideshare
  - Golden Gate Bridge @GGBridge
  - Bay Bridge @BayBridgeInfo
  - Eric C. @transbay
  - ActionNewsSF2 @ActionNewsSF2
  - KGO Radio @kgoradio
  - SF Bay Area Traffic @BayAreaCommuter
  - The Highway Monitor @CAI80thm
  - Sal Castaneda @sal_castaneda

- `locations`: longitude and latitude of San Francisco bounding box, North East -122.75 , 36.8 and South West -121.75 , 37.8

Bibliography

[1] S. Asur and B.A. Huberman. Predicting the future with social media. *Arxiv preprint arXiv:1003.5699*, 2010.

[2] N. Chambers and D. Jurafsky. Templatebased information extraction without the templates. In *Proceedings of ACL*, 2011.

[3] J. Chen, R. Nairn, L. Nelson, M. Bernstein, and E. Chi. Short and tweet: experiments on recommending content from information streams. In *Proceedings of the 28th international conference on Human factors in computing systems*, pages 1185–1194. ACM, 2010.

[4] M. Cheong and V. Lee. Integrating web-based intelligence retrieval and decision-making from the twitter trends knowledge base. In *Proceeding of the 2nd ACM workshop on Social web search and mining*, pages 1–8. ACM, 2009.

[5] N. Chinchor, D.D. Lewis, and L. Hirschman. Evaluating message understanding systems: an analysis of the third message understanding conference (muc-3). *Computational linguistics*, 19(3):409–449, 1993.

[6] N.A. Diakopoulos and D.A. Shamma. Characterizing debate performance via aggregated twitter sentiment. In *Proceedings of the 28th international conference on Human factors in computing systems*, pages 1195–1198. ACM, 2010.

[7] Cassaundra Doerhmann. Named entity extraction from the colloquial setting of twitter.

[8] T. Finin, W. Murnane, A. Karandikar, N. Keller, J. Martineau, and M. Dredze. Annotating named entities in twitter data with crowdsourcing. In *Proceedings of the*

NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk, pages 80–88. Association for Computational Linguistics, 2010.

[9] J.R. Finkel and C.D. Manning. Nested named entity recognition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 141–150. Association for Computational Linguistics, 2009.

[10] J. Friedl. *Mastering regular expressions*. O'Reilly Media, Inc., 2006.

[11] W. Hill and L. Terveen. Using frequency-of-mention in public conversations for social filtering. In *Proceedings of the 1996 ACM conference on Computer supported cooperative work*, pages 106–112. ACM, 1996.

[12] B. Huberman, D. Romero, and F. Wu. Social networks that matter: Twitter under the microscope. 2008.

[13] M. Jansche and S.P. Abney. Information extraction from voicemail transcripts. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 320–327. Association for Computational Linguistics, 2002.

[14] B.J. Jansen, M. Zhang, K. Sobel, and A. Chowdury. Twitter power: Tweets as electronic word of mouth. *Journal of the American society for information science and technology*, 60(11):2169–2188, 2009.

[15] A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis.*

[16] B. Krishnamurthy, P. Gill, and M. Arlitt. A few chirps about twitter. In *Proceedings of the first workshop on Online social networks*, pages 19–24. ACM, 2008.

[17] G.R. Krupka and K. Hausman. Isoquest inc.: Description of the netowl (tm) extractor system as used for muc-7. In *Proceedings of MUC*, volume 7, 1998.

[18] J. Lafferty, A. McCallum, and F.C.N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[19] X. Liu, S. Zhang, F. Wei, and M. Zhou. Recognizing named entities in tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2011), Portland, Oregon*, 2011.

[20] X. Liu, M. Zhou, F. Wei, Z. Fu, and X. Zhou. Joint inference of named entity recognition and normalization for tweets.

[21] B. Locke and J. Martin. Named entity recognition: Adapting to microblogging. *Senior Thesis, University of Colorado*, 2009.

[22] K. Makice. *Twitter API: Up and running.* O'Reilly Media, 2009.

[23] C. Manning and D. Klein. Optimization, maxent models, and conditional estimation without magic. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Tutorials-Volume 5*, pages 8–8. Association for Computational Linguistics, 2003.

[24] C.D. Manning, P. Raghavan, and H. Schutze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[25] M.R. Morris, J. Teevan, and K. Panovich. What do people ask their social networks, and why?: a survey study of status message q&a behavior. In *Proceedings of the 28th international conference on Human factors in computing systems*, pages 1739–1748. ACM, 2010.

[26] W. Murnane. *Improving accuracy of named entity recognition on social media data.* PhD thesis, University of Maryland, 2010.

[27] M. Naaman, J. Boase, and C.H. Lai. Is it really about me?: message content in social awareness streams. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pages 189–192. ACM, 2010.

[28] N. Nicolosi. Feature selection methods for text classification. 2008.

[29] D.W. Oard. The state of the art in text filtering. *User Modeling and User-Adapted Interaction*, 7(3):141–178, 1997.

[30] B. Onyshkevych. Template design for information extraction. In *Proceedings of the 5th conference on Message understanding*, pages 19–23. Association for Computational Linguistics, 1993.

[31] O. Phelan, K. McCarthy, and B. Smyth. Using twitter to recommend real-time topical news. In *Proceedings of the third ACM conference on Recommender systems*, pages 385–388, 2009.

[32] A. Ritter, S. Clark, and O. Etzioni. Named entity recognition in tweets: An experimental study. 2011.

[33] Jagan Sankaranarayanan, Hanan Samet, Benjamin E. Teitler, Michael D. Lieberman, and Jon Sperling. Twitterstand: news in tweets. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 42–51, New York, NY, USA, 2009. ACM.

[34] F. Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[35] D. Shamma, L. Kennedy, and E. Churchill. Tweetgeist: Can the twitter timeline reveal the structure of broadcast events. *CSCW Horizons*, 2010.

[36] D.A. Shamma, L. Kennedy, and E.F. Churchill. Tweet the debates: understanding community annotation of uncollected sources. In *Proceedings of the first SIGMM workshop on Social media*, pages 3–10. ACM, 2009.

[37] S. Singh, D. Hillard, and C. Leggetter. Minimally-supervised extraction of entities from text advertisements. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 73–81. Association for Computational Linguistics, 2010.

[38] S.S. Skiena. *The algorithm design manual*, volume 1. Springer, 1998.

[39] NV Sobhana, P. Mitra, and SK Ghosh. Conditional random field based named entity recognition in geological text. *International Journal of Computer Applications IJCA*, 1(3):143–147, 2010.

[40] B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, and M. Demirbas. Short text classification in twitter to improve information filtering. In *Proceeding of the 33rd international ACM SIGIR conference on research and development in information retrieval*, pages 841–842. ACM, 2010.

[41] C. Wagner and M. Strohmaier. The wisdom in tweetonomies: Acquiring latent conceptual structures from social awareness streams. In *Proceedings of the 3rd International Semantic Search Workshop*, page 6. ACM, 2010.

[42] N. Wanichayapong, W. Pruthipunyaskul, W. Pattara-Atikom, and P. Chaovalit. Social-based traffic information extraction and classification. In *ITS Telecommunications (ITST), 2011 11th International Conference on*, pages 107–112. IEEE, 2011.

[43] J. Weng, E.P. Lim, J. Jiang, and Q. He. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 261–270. ACM, 2010.

[44] Dejin Zhao and Mary Beth Rosson. How and why people twitter: the role that microblogging plays in informal communication at work. In *Proceedings of the ACM 2009 international conference on Supporting group work*, GROUP '09, pages 243–252, New York, NY, USA, 2009. ACM.

Erklärung

Ich versichere, diese Arbeit selbstsändig verfasst zu haben.
Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäßaus anderen Werken übernommene Aussagen als solche gekennzeichnet.
Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prünfungsverfahrens.
Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht.
Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

(Mirna Megally)