

Characterization of traffic events using social media

Master's Thesis

Elwin Dokter

Characterization of traffic events using social media

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE
TRACK SOFTWARE TECHNOLOGY

by

Elwin Dokter
born in Dordrecht



Web Information Systems
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
<http://wis.ewi.tudelft.nl>

Characterization of traffic events using social media

Author: Elwin Dokter
Student id: 1275909
Email: E.Dokter@student.TUdelft.nl

Abstract

Accidents, malfunctioning matrix signs, oil on the road and a bridge that is not able to close are examples of traffic events that happen every day on the Dutch roads. In the Netherlands we are able to measure traffic speeds and calculate travel times on highways and most important secondary roads using a network of connected traffic measuring points. The data that is collected using these points is available for anyone interested and distributed every minute. This distribution frequency makes it possible to detect near real time traffic disturbances. However, the traffic data does not provide information about *what* is actually happening on the road. During rush-hour or particular city events traffic disturbances are expected, but there also exist many disturbances of the unexpected kind: accidents and car or truck breakdowns for example.

In this study we focus on the characterization of traffic events by using Twitter as an information source. Using open traffic data as a traffic event provider we link tweets to traffic events using different linkage strategies in order to extract traffic information. Related traffic tweets can then be used to extract cause types and enrich traffic events.

We developed a demonstrating system for the Netherlands that is able to extract traffic cause types using two different Dutch Twitter datasets. The system uses a set of detected traffic events as its input.

Thesis Committee:

Chair: Prof. dr. ir. G.J.P.M. Houben, Faculty EEMCS, TU Delft
University supervisor: Dr. ir. A. Bozzon, Faculty EEMCS, TU Delft
Committee Member: Dr. Oded Cats, Faculty CEG, TU Delft

Preface

I started my Master Computer Science in 2012. Because of particular courses I followed and my job as web developer I became more and more interested in the Web Information Systems (WIS) specialization. I am particularly interested in data gathering, processing and visualization. Courses that I followed related to these topics are Information Retrieval, Web Data Management and Web & Semantic Web Engineering. Furthermore I like software (re)-engineering and coding. During my Master I therefore also took programming courses in functional- and reactive programming. The data and programming domains are both heavily presented in my Master thesis.

During the Web & Semantic Web Engineering course I wrote a paper about how social media can play a supportive role in detecting traffic disturbances and provide additional information on top of traditional traffic measuring equipment. A couple of months later I received a mail from Geert-Jan Houben in which he asked me if I was interested in a collaboration to make a scientific paper on my topic. After a chat with Houben and Alessandro Bozzon this eventually resulted in a topic for my Master thesis.

I've learned a lot doing my Master Thesis. One of the most important things was the importance of working in a structured, organized and disciplined way. This wasn't always easy. When doing research you sometimes get stuck or spend too much time on peripheral issues. It is therefore important to constantly have a plan or goal in mind where you can work to.

I would like to thank Alessandro Bozzon for guiding me through the whole process and advise me where needed. I would also like to thank Geert-Jan Houben for its inspiring courses and Oded Cats for being a member of my thesis committee. Finally I thank my family and friends for all the great years that I have had as a student at the Delft University of Technology.

Elwin Dokter
Delft, the Netherlands
August 3, 2015

Contents

Preface	iii
Contents	v
List of Figures	ix
1 Introduction	1
1.1 Subject matter	1
1.2 Approach	2
1.3 Contribution	2
1.4 Outline	3
2 Background	5
2.1 Real world event detection using social media	5
2.2 Open traffic data	8
2.3 Traffic information extraction and classification using social media . .	10
2.4 Conclusion	11
I Detection of traffic events	13
3 Detection	15
3.1 Introduction	15
3.2 Experimental Setup	15
3.3 Results	16
3.4 Conclusion	19
II Characterization of traffic events	21
4 Methodology	23
4.1 Collection	24
4.2 Extraction and analysis	26
4.3 Linkage	30

4.4	Classification	35
4.5	Conclusion	37
5	Implementation	39
5.1	Architectural overview of developed system	39
5.2	Data presentation	41
5.3	Data collection	44
5.4	Data extraction	47
5.5	Data linkage	49
5.6	Data classification	50
6	Experiments	53
6.1	Dataset description	53
6.2	Collection	55
6.3	Extraction and Analysis	56
6.4	Filtering and linking	60
6.5	Classification	72
III Conclusions and future work		75
7	Conclusions	77
7.1	Conclusions	77
7.2	Discussion/Reflection	78
7.3	Contributions	79
8	Future Work	81
8.1	Bot detection algorithm	81
8.2	Collection	81
8.3	Linkage	81
8.4	Classification	82
Bibliography		83
A	NDW Dataset examples	85
A.1	MST	85
A.2	TFTS	87
A.3	TT	89
A.4	VILD	91
A.5	Eventinfo	93
B	DBPedia Spotlight example results	97
C	Sentiment services example results	99
D	Linked event example	101
E	Experiment 2 - Results	103

F Glossary

107

List of Figures

2.1	Example of detection of local festivals[3]	7
2.2	Overview of the collection of traffic data	9
2.3	Point on road described relative to VILD points	10
3.1	Detection spot overview for South Holland	16
3.2	Locals are trying to make a ‘trending topic’	17
3.3	Results of collected geo tweets	18
3.4	Tweets per day part	18
4.1	High level overview of the collection steps	25
4.2	High level overview of the extraction steps	29
4.3	High level overview of the linkage steps for the VILD strategy	31
4.4	full-VILD strategy	33
4.5	High level overview of the classification steps	36
5.1	Architectural design of the developed system	40
5.2	Live map with data.	41
5.3	Link tweets to events experiments interface	43
6.1	Word cloud of size 1000 for the ‘file’ keyword	54
6.2	Timespan of the created datasets	55
6.3	Annotation results using DBPedia Spotlight	59
6.4	Word cloud of annotation results	59
6.5	Related-real tweets distribution of events	65
6.6	Cause type distribution of related-real and other-real tweets	66
6.7	<i>hasPicture</i> distribution of related-real and other-real tweets	66
6.8	<i>hasUrl</i> distribution of related-real and other-real tweets	67
E.1	Number of events with at least one ‘related-real’ tweet for different cause types and linkage strategies	104
E.2	Spread of tweets related to timespan of traffic events	105

Chapter 1

Introduction

1.1 Subject matter

In the Netherlands people are dealing with traffic jams every day. From casual rush hour jams to big accidents that cause total congestion for several hours. Different traffic variables, such as traffic speed and traffic flow are measured in this country at different points on the main roads every minute. This kind of numerical traffic information enables us, for example, to make an estimation about the total travel time needed for a route. Furthermore it can tell us at which places there are traffic delays. However, during traffic congestion this data does not provide us any information about the cause of the delay, possible lane closures, etc.

In the Netherlands the National Data Warehouse for Traffic information (NDW¹) is responsible for collecting and distributing the traffic data to consumers and other interested. Since the first of September 2013 this data has become Open Data, which means any interested person can retrieve the data for free. At this moment there are more than 25.000 measuring points in the country that provide reliable and precise measurements every minute.

The NDW data makes it possible to detect disturbances on roads. For single lane provincial roads various algorithms have been developed to accomplish this task, as researched in [8]. The drawback of this data is that although detected disturbances provide you information *that* something is happening on the road, it does not provide you precise information about *what* is happening. When comparing the speed slowdown patterns of rush hour jams and accidents it might be possible to make a distinction between causes of disturbances, but you can't know for sure. Imagine you are on your way home in the middle of the night and suddenly you get totally stuck. The matrix signs above the road are off. You and the rest of the jam have no clue about what is going on. Most of the people now are curious about why they are standing still at this strange point in time.

Nowadays many mobile traffic applications provide the possibility for users to give real-time feedback about traffic situations. Google maps recently added traffic reports from Waze [9]² to their maps to give other road users the opportunity to avoid congested roads. This user generated content is reported by users using the Waze app on

¹<http://www.ndw.nu>

²<http://waze.com>

mobile devices. By integrating this service to a road map you add semantics to road disturbances. In this research I propose a similar approach to use Twitter as social media platform to enrich traffic disturbances. Twitter is a popular platform, the messages are always short and it has a real time nature. These properties seem to make the platform suitable for the traffic enrichment task. The main research question in this thesis is the following: *'To what extent can social media support traffic information during uncommon disturbances on the road?'*. In the first part of the thesis we investigate to what extent social media can be used to *detect* traffic disturbances. The second part is devoted to the characterization of disturbances.

1.2 Approach

Using the Twitter Streaming API³ we constructed two datasets. The first set contains tweets that are retrieved using a keyword search with traffic related terms for a period of 6 months. The second dataset consists of tweets that were received using a geo bounded box covering the whole Netherlands. We've also collected traffic event information for the same period. This information is collected by Rijkswaterstaat and distributed by the NDW roughly every 10 minutes. The traffic events consist of all kind of information, such as queue length of disturbances and the locations of events. The location is given as a combination of latitude and longitude coordinates, but is also described related to VILD points. VILD points, also used for communication in navigation systems using TMC, are included in VILD tables. These tables link VILD points to all kind of information that describe the points: road names, location descriptions (bridge, intersection, main road, etc), road numbers, etc. In this thesis we will often refer to one collection of distributed traffic events by Eventinfo set.

The traffic measuring equipment is installed at the locations of the VILD points. Therefore, together with the knowledge that NDW data can be used to detect traffic disturbances[8], we make the assumption that the traffic disturbances described in our third dataset could have been detected using the raw traffic data that is distributed every minute. With this assumption we use the Eventinfo set from the NDW together with the information from the VILD points in our research to connect the collected tweets to the events described in the Eventinfo set.

1.3 Contribution

The following contributions are made:

- In the literature there has been quite some research about people acting as social sensors, and a few about social sensors in traffic [6][7][1]. Our work is different in the sense that it aims to combine precise, reliable and real-time numerical open traffic data with textual information that can be retrieved from social media. Other works focuses on one particular social media platform that is used for the detection and classification of traffic events. We believe that the power of social content is in detailed descriptions of traffic events and multimedia items, such as pictures, it can include.

³<https://dev.twitter.com/docs/streaming-apis>

- In the scientific field, in the Netherlands, the combination of open traffic data and social media content is new.
- A demonstration software system has been developed that extracts causes of disturbances by linking and classifying tweets to traffic disturbances. Furthermore the software can be used to evaluate the results of several experiments that have been done.

1.4 Outline

The structure of the remaining parts of this thesis is as follows. Chapter 2 gives information about related work that forms the background for the thesis. In the first part we use social media to detect traffic disturbances. This is described in chapter 3. The second part is about the characterization of traffic events. Chapter 4 describes the different steps of the developed methodology, including several algorithms. Chapter 5 is about the implementation of the methodology. It gives more insights in the architectural structure of the software that has been developed. Chapter 6 is about several experiments that have been performed together with the results. Finally, in the last part, the chapters 7 and 8 conclude the research and provide several tasks that remain as future work.

Chapter 2

Background

2.1 Real world event detection using social media

One of the properties of Twitter is its real time nature. Because the microblogging service forces you to make posts within 140 characters, updates are often placed very quickly. In combination with the popularity that Twitter has some say that the platform acts as a mirror of the real world with all the information living inside. Political debates, live updates of sport events, but also critical information about natural disasters like storms, fires and earthquakes are examples of information that can be found on Twitter. The problem is how to retrieve and extract exactly that information from Twitter you are interested in.

2.1.1 Query-based approach

In [6] the authors use Twitter to detect earthquakes in real time. The fact that people are tweeting about earthquakes immediately after they occur makes these users act as social sensors. The goals that Sakaki et al. have in [6] can be summarised as follows:

- to investigate the real-time interaction of events like earthquakes using Twitter
- to propose a method to monitors tweets and detect potential earthquakes
- to develop an earthquake reporting system for Japan that sends emails to registered users as soon as an earthquake is detected

To find relevant information they use a *query based approach*, searching for tweets containing either 'earthquake' or 'shaking'. To train a classifier to make the distinction between positive and negative tweets they look at the keywords used, the the number of words in a tweet and the words before and after the search query words. To estimate the centre location of the earthquake they use location filtering algorithms such as Kalman filtering and particle filtering[6].

The experiments show that the developed reporting system in general detects earthquakes much faster than the default broadcast service that is used in Japan. 96% of the quakes with a seismic intensity scale of 3 or larger are detected by the system. One side-note to place here is that Japan is a perfect country for such a system to perform well. It has a high density of Twitter users; the number two country of Twitter traffic

at the time the paper was written. Furthermore there are many earthquakes in Japan to detect. Nevertheless the method of Sakaki et al. have shown that information about disaster events can be found and extracted, near real-time, from Twitter.

2.1.2 Geo-based approach

Fujisaka et al. [3][2] argue with the query based approach to detect events as used in the previous described method [6]. To adjust the system to detect other events than earthquakes; like storms, fires and traffic jams, the input query and the extraction techniques needs to be manually adjusted. This forms a limitation for a query based event detection system. Therefore they propose a different method, focusing on the location awareness of social media applications on mobile devices. Using the geographical meta-data of social content and social media users, irregular behaviour can be observed. This indicates that something might be happening. Their contribution is to develop a geo-social event detection system for Japan by monitoring crowd behaviour indirectly through Twitter. According to the authors it is easy to detect unusual incidents such as town festivals or unexpected natural disasters with the proposed event detection system [3][2]. In order to observe irregularities, the geographical regularities of local crowd behaviours needs to be registered. In short, the method of Fujisaka et al. can be summarized as follows:

- Divide the region of interest into smaller regions and investigate the usual status of crowd behaviour: record the number of tweets composed and the Twitter users tweeting per region and estimate the normal amount by averaging these numbers.
- A sudden increase in the number of tweets, the number of people tweeting or the movement of a crowd in or between regions forms a trigger that something might be happening.
- To find out if indeed something is going on the content of the ‘suspicious tweets’ needs to be investigated.
- By looking at the geographical history of tweets from a user within a short time range, movements of crowds can be observed.

One of the most interesting parts of the methodology is that the authors define several possible cases for specific combinations of the following properties:

- the number of tweets composed in a region (#Tweets)
- the number of Twitter users in a region (#Crowd)
- the movement of a crowd (#MovCrowd)

For example, a sudden increase of tweets, new twitter users and movements of a crowd for a short amount of time can indicate that a festival is taking place. An increase in the number of tweets, but a stable number of twitter users (or crowd) in a region can indicate a local festival, as demonstrated in figure 2.1.

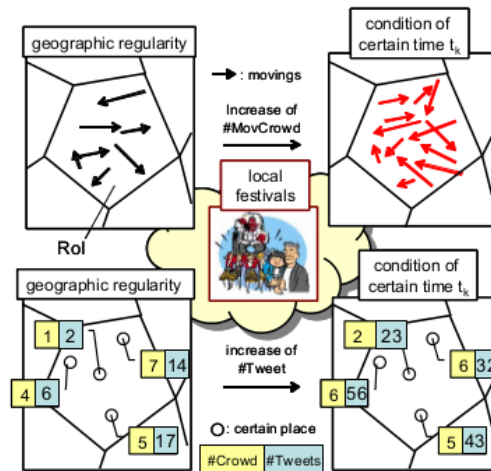


Figure 2.1: Example of detection of local festivals[3]

In order to observe irregularities the authors work with time frames of 6 hours each: morning, afternoon, evening and night. Depending on the combination of stable and changing properties the system makes a final decision: the situation is either *normal* or *abnormal*. To test the performance of the system the authors used a list of 15 known festivals that took place in Japan. 13 out of 15(87%) were detected by the system. Since they only tested a finite small set of just 15 festivals the recall of the system was 1.8%. Besides the festivals they also found other interesting events with the system: soccer games, earthquakes and thunderstorms.

It is nice to see that a relatively easy method can be used well to detect (unknown) events. The detection works completely independent from the actual content of the tweets and therefore you do not have to use complicated extraction techniques. The drawback of the method is that a lot of preparing work needs to be done in advance. The situation of interest needs to be split up in smaller regions in a smart way. Furthermore, the geographical regularities need to be determined, which will cost some time. Since significant changes take more time to detect than searching specific query terms in a tweet, it can sometimes be more difficult to do fast event detection using this approach. It is good to see that the method can be used to detect all kinds of events, however the content needs to be analysed afterwards to see what actually is happening. The authors did not take this into account in their method and let it be manual work.

2.1.3 Conclusion

Research have shown that social media can be used to detect real word events [6][3][2]. With a query-based approach real world events can be detected fast [6]. As soon as social content arrives that contain suspicious terms, like ‘shaking’ or ‘earthquake’ in the research of Sakaki et al., an event has been detected and an early warning system can be triggered to update users about the potential disaster. With a geo-based approach it takes more time, in general, to detect events. A significant change in active users or amount of social content composed in a region needs to be monitored first in order to

detect a potential event [3].

With a query-based approach the input query needs to be manually adjusted as soon as you are interested in a different kind of event. A geo-based approach can detect all kinds of real-world events, since it only make use of geographical properties of social content and social media users in order to perform detection.

In case of earthquake detection the authors make the assumption that there occurs only one earthquake at a time [6]. In our work, where we deal with traffic congestion, we have multiple traffic events at the same time. Furthermore we need a very precise location of detected traffic events in order to be useful. In the next section we give an introduction to the open traffic data that is used in the Netherlands.

2.2 Open traffic data

The National Data Warehouse for Traffic information (NDW) is a collaborative partnership of 24 road operators in the Netherlands that have the main goal to develop and maintain one joint database for all traffic data in the country¹. This data has become open data since September 2013 and can therefore be retrieved by any interested with an internet connection. Different techniques, such as license registering camera's, loops beneath roads and the detection of bluetooth devices are used in order to collect the traffic data from road users, as shown in figure 2.2. This data is collected at more than 25.000 measuring locations throughout the country and distributed every minute. The near real time traffic data is an extension to the European DATEX-II format² and include the following items:

traffic speed (TS) the average speed of vehicles that pass a measuring point in km/h

traffic flow (TF) the number of vehicles that pass a measuring point during a period of one hour

travel time (TT) the estimated travel time in seconds between two measuring points

For some measuring points the traffic data contains detailed traffic speed, -flow and travel time information for specific lanes and different vehicle lengths. Traffic speed and traffic flow values are collected in one XML dataset. The travel time information can be found in a separate XML file. Records in TT and TFTS files are referring to measurementSiteReference id's. These id's can be found in the Measurement Site Table- or MST-XML file. This dataset describes where the measuring points are located and what their properties are.

Examples of snippets of the described datasets can be found in Appendix A.

Besides the near real time traffic data the NDW also distributes another dataset about the availability of the road, which consists of manually entered events such as traffic jams, accidents and road construction zones. This dataset is maintained by Rijkswaterstaat³ and distributed roughly every 10 minutes. In the rest of the thesis we will refer to this dataset as *Eventinfo* or *EI*. An example of an EI dataset can also be found in Appendix A.

¹<http://www.ndw.nu>

²<http://www.datex2.eu/>

³<http://www.rijkswaterstaat.nl/>

Verkeersgegevens up-to-date

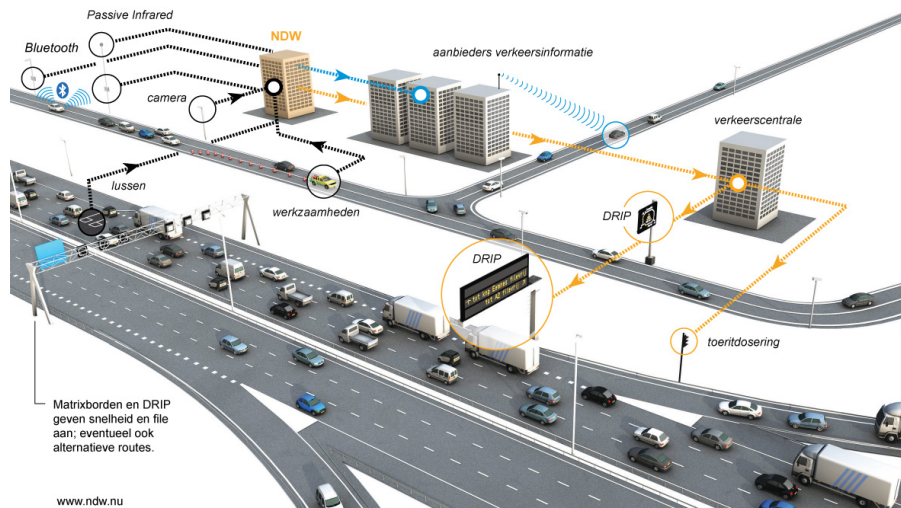


Figure 2.2: Overview of the collection of traffic data

The location of measuring points is besides precise coordinates also described relative to points in the traffic information location database or VILD table. The VILD table describes a network of points, lines and areas that contain all kind of information such as road numbers, road names and types of points (intersection, highway, exit). The TMC table which is often used by car navigation systems to inform road users about the location of traffic events is derived from the VILD table. In Code A.6 in the appendix is displayed how the location of a measuring point is structured, starting at line number 71.

- The *locationForDisplay* tag describes the precise coordinates in WGS84 format.
- The *alertCPoint* tag refers to the point in the VILD table: the first three tags within the *alertCPoint* (*alertCLocationCountryCode*, *alertCLocationTableNumber*, *alertCLocationTableVersion*) describe the specific version of the table that is used. The *AlertCDirectionCoded* describes the direction. The *offsetDistance* describes the offset in meters between the measuring point and the VILD point in the *specificLocation* tag.

In the above example the record is referring to just one VILD point, indicated by the *csi:type="Point"* property. Other possibilities are *line* (2 points) and *area* (>2 points).

In figure 2.3 two measuring points are shown relative to the closest VILD point (9240) that is indicated by the orange dot.

Part of the methodology in chapter 4 make use of the information from the VILD table.



Figure 2.3: Point on road described relative to VILD points

2.3 Traffic information extraction and classification using social media

In 2.1 we showed how researches were able to extract different real world events from social media using different approaches: query-based and geo-based. In this section we describe how social media has been used to extract and classify traffic information.

2.3.1 Sakaki

In the work of Sakaki et al. Twitter is used to extract and classify traffic events [7]. Like the previous work on earthquake detection of Sakaki et al. in [6], this is again a query based approach.

An Intelligent Transportation System or ITS supports drivers in choosing the best route by providing traffic information in terms of traffic speed and traffic flow. The previous discussed data from the NDW is an example of the data that is used in ITS. Sakaki et al. believe there are some drawbacks on the use of these techniques. In the first place, the system can only be used at preassigned areas where the measuring equipments has been installed. Second, the data only provides numerical information. Social media can provide additional valuable information like road- and weather conditions. The target region of [7] is Japan. The ITS in Japan is good enough to verify traffic event detections using social media and there is enough traffic congestion to do such a research.

In the first phase of the methodology *positive tweets* that are related to traffic events are extracted using Support Vector Machine. The authors searched for the follow target information: traffic restrictions, police checkpoints, rain and mist. For this classification experiments achieved an average precision of 0.71 with a recall of 0.91. The dataset from this experiment from November 2010 consisted of 540 tweets.

The second phase of the methodology includes the conversion of geographically related terms, that can be found in the tweet text, to geographical coordinates using Natural Language Processing. Only 0.1% of the collected tweets contained GPS coordinates. Therefore the search for location related terms in tweet texts is crucial to increase the set of detected traffic events. The authors constructed a location dictionary using 26.893 place names from Wikipedia and 24.619 from Hatena Keywords. When a tweet does not contain precise geographical information the constructed place dictionary is used to see if there is a matching place with the terms from the tweet. Experimenting on a dataset of 540 tweets from January 2011 an average precision of 0.72 and a recall of 0.48 was achieved on location estimation using the pre-constructed dictionary.

2.3.2 Wanichopong

In the work of Wanichapong et al. the authors classify traffic information found on Twitter in either a point category (e.g. a car accident at place X) or a link category (e.g. a traffic jam between A and B)[1]. In order for a tweet to be relevant it must contain information about *what* is happening and *where* it is happening. To find these two properties they tokenize tweets and define several categories for each token or a group of tokens: place, road, verb, ban, start- and end preposition. A question mark is an example of a 'ban' token, because the authors believe that traffic information should be informative, factual and definitive. There is no place for questions. They use dictionaries to categorize the tokens. When a tweet fulfils to specific rules regarding the different categories it is considered to be a *positive tweet*. To convert place tokens found in tweets into precise geographical coordinates Wanichapong et al. use a similar approach as the one found in [7]: a combination of an own constructed dictionary and the Google Maps geocoding API⁴. Experiments classified 2942 tweets into the point category and 331 into the link category with an accuracy of respectively 76.85 and 93.23%. The average accuracy of the location estimation step was 92.20%. This seems to be 20% higher than the result of Sakaki et al. in [7].

2.4 Conclusion

Several researches show that social media seems suitable to detect all kind of real world events such as storms, fires, earthquakes, traffic jams and riots. Natural language processing techniques enable us to extract the information we are interested in from social media content. With a geo-based approach we can detect all types of events without expensive textual processing techniques. However, if we want to know what causes a sudden irregularity in the amount of social media users or social content we do

⁴<https://developers.google.com/maps/documentation/geocoding/>

need to take the content into account. Furthermore real-time detection is hard, because it takes some time to detect irregularities events and prevent false detections.

With a query based approach you search, by using well thought search terms, for more specific predefined events. Events can be detected faster, but all social media content needs to be processed or classified in order to become useful. Support vector machines, which are trained using test datasets, are often used to support this task.

In the Netherlands the NDW provides near real time numerical traffic information with traffic speeds, traffic flows and travel times on different trajects. This information is reliable, precise and has a fixed temporal resolution of one minute. The disadvantage of the data is that it does not provide you additional information about what is going on during (unexpected) traffic disturbances.

The work of Sakaki et al. and Wanichapong et al. have shown that Twitter can be used to extract and classify traffic information [7][1]. Using a query based approach together with a collection of predefined dictionaries the researches were able to correctly extract the location of events with an accuracy $\geq 72\%$

In our research we seek for the right balance between the reliable and precise open traffic data and the rich traffic information that can be found on social media to detect and characterize traffic disturbances in the Netherlands.

Part I

Detection of traffic events

Chapter 3

Detection

3.1 Introduction

In the first part of the thesis we focus on the detection of traffic disturbances using Twitter. Using a geographical approach, related to the work of Lee et al. [3][2], we investigate if the geographical data in tweets reflects places that are getting more crowded. Therefore we've chosen some spots from where we know *that* and *when* events will take place. In contrast to the work of [3] we only investigate how the number of tweets changes over time: before, during and after an event. We do not first determine the residents or consider all *Twitter users* in the areas that tweet with geo meta data enabled. The reason for this is that we just want to know how much information we could retrieve from Twitter and if the amount of tweets is related to the traffic flow information that can be retrieved from the open traffic data from NDW.

3.2 Experimental Setup

As said during the introducing section we would like to monitor tweets at 'hot spots'; locations where we know that events will take place frequently. We created the spots to collect tweets with gps coordinates at the following popular places where often soccer games or concerts take place:

- Ahoy (Rotterdam)
- Ziggo Dome (Amsterdam)
- De Kuip, Feyenoord (Rotterdam)
- GelreDome, Vitesse (Arnhem)
- Parkstad Limburg, Roda JC (Kerkrade)
- Kyocera, ADO Den Haag (Den Haag)

To collect tweets in these areas we use the Twitter streaming API¹. We use the soccer stadiums and concert halls as centres for our recordings and record in a circular

¹<https://dev.twitter.com/streaming/overview>

area with a radius of one kilometre. For South Holland this is visually displayed in figure 3.1. The spot centred at the Ziggo Dome in Amsterdam also includes the popular Heineken Musical Hall and the Amsterdam Arena, because those buildings are within one kilometre radius of the Ziggo Dome.

When one wants to use the Twitter streaming API's to record such geographically determine areas it is possible that Twitter will not reveal all tweets for free [4]. To reduce the chance of loosing too much tweets we want to do some recordings in dual. We will start these recordings at DataSift², using the software trial possibility the platform offers.

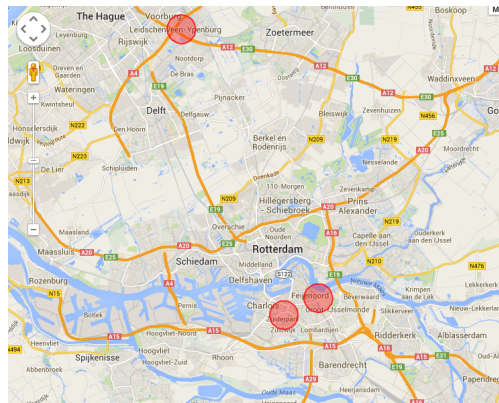


Figure 3.1: Detection spot overview for South Holland

3.3 Results

The amount of received tweets is very low in all areas, as can be seen in figure 3.3 where the number of tweets per hour are plotted for four locations during three full recording days in February. Unfortunately we did not get any results from our free recordings at DataSift. Investigating the content of our own received tweets during the spikes in the evening reveal that soccer events took place. With one exception: the yellow bars at the Kyocera spot in The Hague at February 6th. Here a couple of teenage girls in a street close to the stadium tried to create a ‘trending topic’ on Twitter at exactly 7 O’clock in the evening. The spike during that hour is higher than the spike at the same hour during the day that ADO The Hague played their home soccer game at February 4th. This simple example shows how false events could be detected when we only take the number of tweets into account. However, since all tweets during the spike at February 6 are created by residents in the Kyocera area the tweets could be distinguished from the ones that are created by people visiting the stadium where the biggest part of the people will be from outside the area. According to the work of Lee et al. in [3] this situation will be considered as a *local event*. Preliminary research should be done when we want to determine the resident in all areas. As said during the introducing section we only wanted to see if Twitter can reflect traffic data as the data from the NDW does. Therefore we did not determine the residents in the recording

²DataSift is an paid on-line service where you can easily start some Twitter recordings by using a graphical interface (<http://datasift.com/>)

areas. The data from NDW forms the ground truth for our detecting study using social media, since the data is reliable and precise.

In figure 3.4 the number of tweets per day part (6 hours) are plotted for the Ziggo Dome spot for 12 days. Two spikes, one in an afternoon and one in an evening are clearly visible here. Both spikes are caused by soccer games that were played at the Amsterdam Arena. These numbers are of the same order as the results in [3], where the authors also work with time frames of 6 hours. It shows that it is indeed fairly easy to detect events using a geo-based approach.

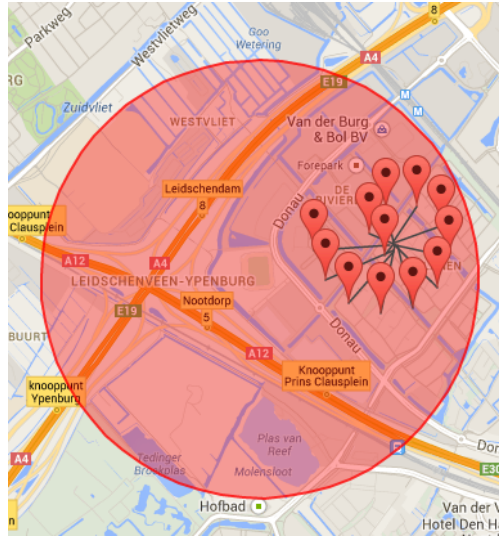


Figure 3.2: Locals are trying to make a ‘trending topic’

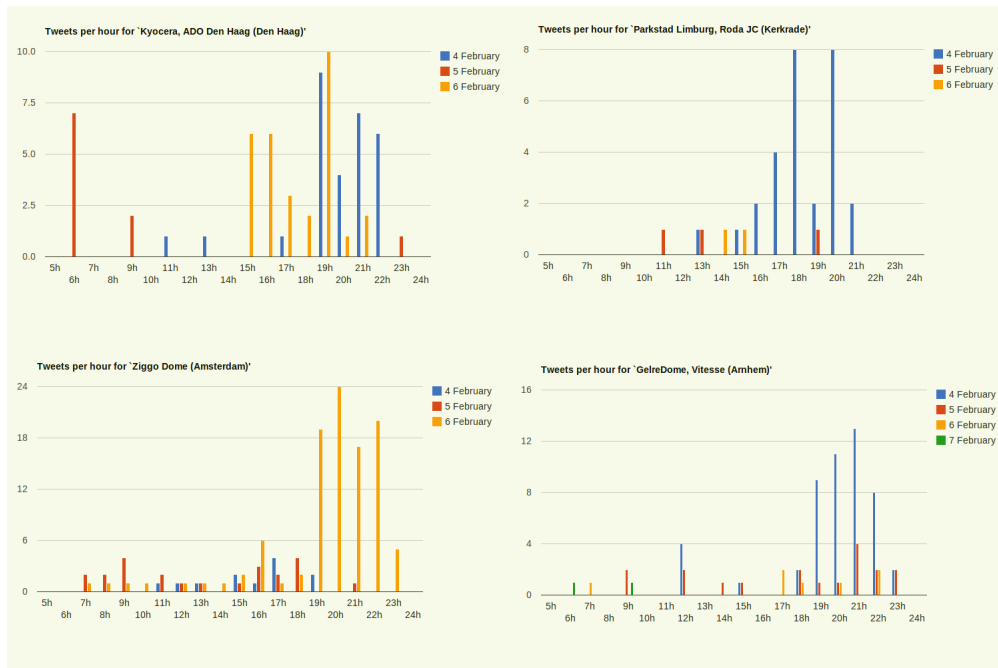


Figure 3.3: Results of collected geo tweets

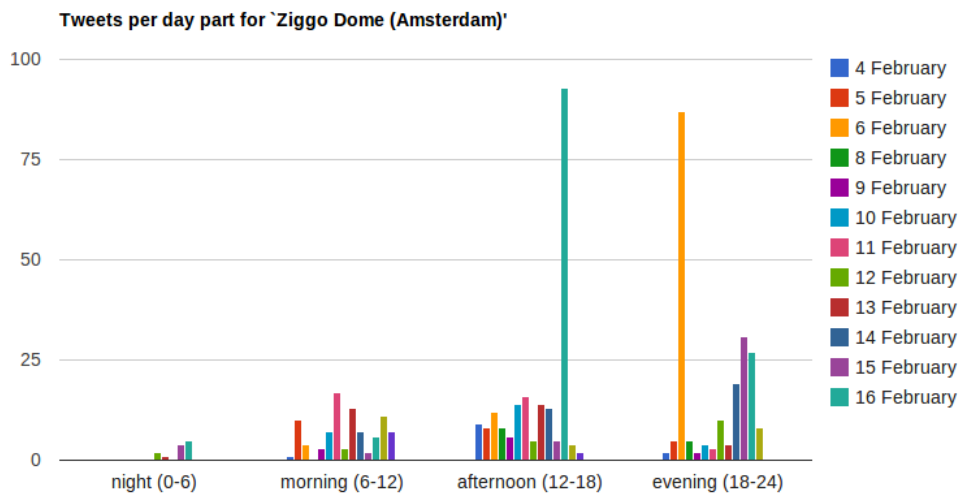


Figure 3.4: Tweets per day part

3.4 Conclusion

When there are events like soccer games we observe that the amount of composed tweets increase. The data from the NDW show denser traffic during the recorded spikes. So, we might say that a geo-based approach using Twitter *can*, to some extent, reflect the real traffic situation. However, the amount of tweets we received is far too low to compete with the data from the NDW, which provide more detail and has a fixed temporal resolution of one minute. The event detection using a geo-based approach seems to be more suitable to detect different kind of real world events *at a high level*. It is relatively easy to setup recording areas and one of the nicest thing of the method is that, since you only rely on geo metadata of social media content, you can detect all kind of events. No difficult extraction and filtering techniques are needed in order to detect that *something* is happening. For the detection of disturbances in our country were the ITS is well developed the method just didn't work out. The low amount of received tweets cannot compete with the real time open traffic data that provide detailed and reliable data every minute.

We were not able to get enough detailed traffic information from Twitter using a geo-based approach. Events were detected, but we did not receive detailed information about which corners and roads were more crowded than others. Something that the open traffic data in the Netherlands can provide.

Part II

Characterization of traffic events

Chapter 4

Methodology

In the second part of the thesis we focus on the characterisation of traffic events. While the open traffic data can be used to detect traffic disturbances it does not provide us any information about *what* is going on. Images from traffic monitoring camera's can be used for this purpose, but it's quite a long way from the capturing camera to the road user that wants to avoid traffic jams and is interested in these images. *Someone* needs to be constantly focused on the images of the traffic monitoring camera in order to observe a potential traffic disturbance. Afterwards it may take quite a while before the information reaches the relevant road users: traffic info providers need to know about the disturbance and forward this information. It is quite a long path, with many manual work involved. People that rely on broadcast messages from radio stations, which are typically 4 times per hour during rush hour, this information stream is slow. What if we could produce, process and deliver the information about traffic disturbances by the road users *themselves* to *other* road users? That is the philosophy behind the developed methodology: using information hidden in Twitter streams to *collect* and *extract* traffic problems and bring the useful (*filtered*) information about these problems to the *right* road users.

In this chapter we present the methodology we developed in order to enrich detected traffic disturbances with information from real road users using Twitter.

The methodology consists of four different steps or phases. The purpose of each step can be summarized as follows:

Collection

During the collection step all required data that will be used to enrich detected traffic disturbances is collected. This includes tweets from Twitter, eventinfo datasets (EI) and VILD tables from NDW.

Extraction and analysis

The goal of this phase is to extract useful information from the data to retrieve first insights into the data. DBpedia Spotlight¹ can be used to extract entities from the collected tweets to get a better understanding of what is talked about. Using sentiment analysis on the retrieved tweets we hope to find out a little bit on how people react during different traffic disturbances. With simple pattern

¹<http://spotlight.dbpedia.org/>

matching we can extract roads from tweets and using external databases, like Geonames², we can extract places.

Linkage

In the linkage phase the collected tweets are linked to traffic disturbances that are detected by Rijkswaterstaat³ and distributed by the NDW. We will refer to this traffic disturbances in the rest of the chapter as *events*. Different linkage strategies are developed in order to find the best results for this linkage.

Classification

In the classification phase we determine the cause types for detected traffic events, when possible, using the linked tweets from the previous step.

In the next sections we will discuss the methodology steps in more detail.

4.1 Collection

In the collection step all data that is needed for the succeeding steps of the methodology should be retrieved. A high level presentation of the collection process is shown in figure 4.1. Two different streams of two different data providers are processed and the resulting data is stored into a database. Because we decided to use the JSON-based CouchDB software for our data storage⁴ all data can be perfectly stored into one single database.

4.1.1 NDW

As explained in the introduction (1.2), we assume that traffic disturbances, as described in the Eventinfo set or EI, can be detected using the raw traffic speed and traffic flow values the NDW provides for over 25.000 points in the Netherlands every minute. The thesis work of Jasper Vries[8] is an example that uses this values to detect disturbances on single lane provincial roads. Based on this assumption our methodology heavily relies on the properties, availability and correctness of the traffic events described in the EI dataset. Working with a predefined set of events has several advantages. Each event has a short cause description that tells us why the traffic disturbance is happening. It also is bounded in time: the event has a start- and an end time. And finally, each event has a precise location: one or more points with latitude and longitude coordinates. Because the events describe *what*, *where* and *when* something is happening they are ideally as a ground truth.

The bottom row in figure 4.1 describes the path of the collection of the NDW data. Since September 2013 the data has become Open Data and is accessible from file servers using the file transfer protocol (FTP). Most of the files distributed by NDW are in XML-format, except the VILD tables which appear as CSV documents. For all data this means that some process or module needs to be created that parses this data

²<http://www.geonames.org/>

³Rijkswaterstaat, as an executive agency of the Ministry Department of Infrastructure and Environment, has to make sure that road traffic on the Dutch highways flows as quickly and save as possible. <http://www.rijkswaterstaat.nl/>

⁴<http://couchdb.apache.org/>

so that it can be stored in our CouchDB database. Examples of NDW datasets can be found in appendix A. The data is described in section 2.2.

4.1.2 Twitter

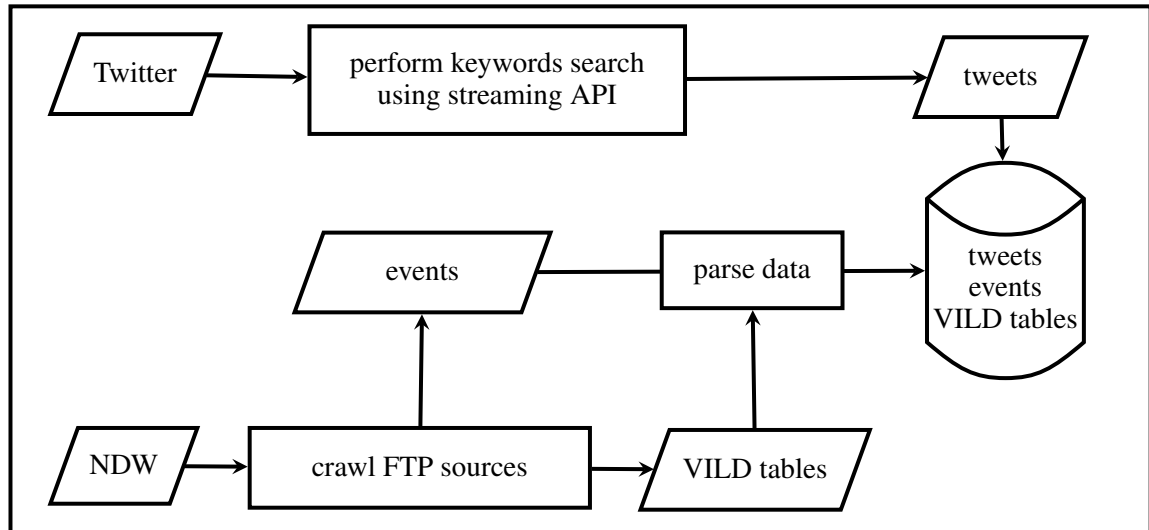


Figure 4.1: High level overview of the collection steps

In the top row in figure 4.1 the Twitter collection path is shown. An important question related to the collection of tweets is: how do we retrieve the highest possible amount of potential interesting tweets? In other words: how do we get the highest recall. From literature, and our performed experiments described in the first part of the thesis, we know that a geographical search approach on Twitter gives poor results. For a large area, for example the whole Netherlands, we are expected to receive no more than 1% of all composed tweets [4]. We could do ‘smart things’, like partitioning the country into smaller pieces and execute multiple crawlers. However, with such a setup it is still unknown how many tweets are retrieved and more important: how many are not. Furthermore, the overhead of non-relevant tweets with his approach is large and will require more data storage.

A better approach is to perform a search that is more focused on the data that we actually want to retrieve: those describing traffic disturbances. A keyword based approach seems suitable for this tasks. We are aware of the fact that this approach makes it less flexible to adapt the methodology to other domains than traffic, because the input queries are only working for the particular domain of interest[3][2]. The fact that we can search much more focused is decisive.

The public streaming API’s of Twitter allows you to search up to 400 different keywords. It is hard to come up with a set of traffic related terms that much, so when we start our search with manually chosen traffic related terms this limit should not be a problem. However, it is still possible that in the future we want to add other terms to our search set, like certain places or road names. Therefore it is important that we choose our keywords wise. In [5] is explained that in many cases there exists terms

/ keywords that are co-occurring all the time in a text. It is good to recall this fact at times that limitations, like API restrictions, play an important role in the obstruction of your data retrieval. After the keywords search is set up the tweets that are obtained are saved into the database. Because the Twitter streaming API's can return JSON documents of tweets, it is fairly easy to store these documents into our database.

4.2 Extraction and analysis

The main goal of the extraction and analysis phase is to retrieve first insights and get a better understanding of the data. What type of traffic disturbances are people tweeting about? And when are they tweeting about traffic problems? The pipe-line of the extraction and analysis phase is shown in figure 4.2. The extraction steps can, in theory, occur directly after a tweet have been received during the previous collection step. Because we have restricted access to several services that we would like to use in the extraction and analysis phase we don't want to send all collected tweets to those services. A set of *ban words* is maintained that consists of non traffic-related terms. A tweet is skipped for extraction when it contains at least one ban word. Furthermore we want to eliminate tweets that are composed by '*bots*'.

4.2.1 Detecting bot accounts

Along with people that tweet about traffic accidents and traffic jams there are also a lot of accounts from news or traffic media providers that use Twitter as a platform to make traffic announcements. Tweets from those users are considered as spam in the developed system. The reason for this is that the source for those users is almost always derived from views from traffic monitoring camera's. Since the goal in this study is to use observations from 'real people' that are in some way involved with the traffic disturbance, the messages from the news providers does not fit in. Therefore their messages are disregarded. The accounts that provide the messages are called bots. The twitter messages composed by bots are stored to the database with a *bot_status* flag set to true. The bot status for a twitter user is determined *once*: as soon as the first tweet of the user reaches the system. For every received tweet we check if the 'bot status' of the twitter user already exists in our database. This is the *get bot status* process in figure 4.2. If the twitter user is not yet known to the system a *bot detection algorithm* is executed to determine if a user is real user or not. Afterwards this value is stored into our database. The pseudo algorithm of the bot detection algorithm is given in algorithm 1.


```

if user already in database then
  | return status according to database
else if username contains one of the following suspicious terms: 'file',
'verkeer', 'news', 'nws', 'nieuws', 'weer', '112', 'headline' then
  | return true
else
  | Retrieve last 25 tweets from user.
  | if at least 50% of the retrieved tweets contain at least one of the search
  | keywords then
  | | return true
  | else
  | | return false
  | end
end

```

Algorithm 1: Determining bot status

So, a user is considered to be a bot when its username contains suspicious traffic or news related terms or at least half of its last 25 tweets contains at least one traffic related term. Some quick tests showed that this fairly easy algorithm does the job quite well. The algorithm should work in other domains too when using different keywords. The only modifications that needs to be done is the change of the suspicious terms. An estimated performance of algorithm 1 is given in chapter 6.

4.2.2 Annotations and sentiments

To automatically extract entities from the tweet texts each tweet is annotated by making a call to the Dutch DBPedia Spotlight⁵. Before the call is made the hash-tag symbol(#) is removed from the tweet, because the service is not able to recognise terms that start with a hash-tag. The rest of the tweet message is unaffected. The goal of the annotation is to get an idea of the distribution of topics our collected tweets are about.

After the tweet is annotated the next step enriches the data with sentiment scores and labels. By doing this we hope to get a little bit of an idea on how people react to different traffic problems. We found two different sentiment services for the Dutch language: text-processing⁶ and ai-applied⁷. Because of some API restrictions it is expected that not all collected tweets can automatically be accompanied with sentiment scores and labels.

4.2.3 Roads and places

The last extraction steps of our methodology extract places and road names from tweet texts. We use Geonames⁸ as place provider. To increase performance all place names that are available in the Geonames database are stored in our own local database. An

⁵<http://nl.dbpedia.org/spotlight/rest/spot/>

⁶<http://text-processing.com/api/sentiment/>

⁷http://api.ai-applied.nl/api/sentiment_api/

⁸<http://www.geonames.org/>

index is constructed that maps the Geonames place names (Rotterdam, Amsterdam, Utrecht) to pairs of latitudes and longitudes (51.9225, 4.47917). Before we extract roads and places we strip tweet texts by removing special characters and punctuation symbols. Afterwards we tokenize the tweet. For all tokens we search for matching places in our places index and we perform a simple pattern match to recognise 'A' and 'N' roads.

4.2.4 Overview extraction steps

The following list summarises all extraction steps.

1. Retrieve tweet from database that is not already extracted
2. Check if *bot status* for twitter user is present in local database and return accordingly. If not, perform bot detection algorithm on user: save status for user in database and return
3. A tweet is skipped for extraction if it is composed by a bot
4. Remove hashtag from tweet, so that terms can be recognised by DBPedia
5. Add annotations by using DPPedia Spotlight
6. Add sentiment scores and labels by using the *text-processing* and *ai-applied* services
7. Replace upper case letters in the tweet message by lower case variants. This makes it easier to match with place names
8. Remove special characters and punctuation symbols
9. Tokenize tweet
10. The following pattern is used to extract highways and provincial roads from tweet message: $[A|N][0-9]^+$.
11. Tokens with 3 or less characters are removed
12. A tweet is skipped for further extraction if it contains at least one ban word.
13. For the remaining tokens check if they have a matching place name in local database

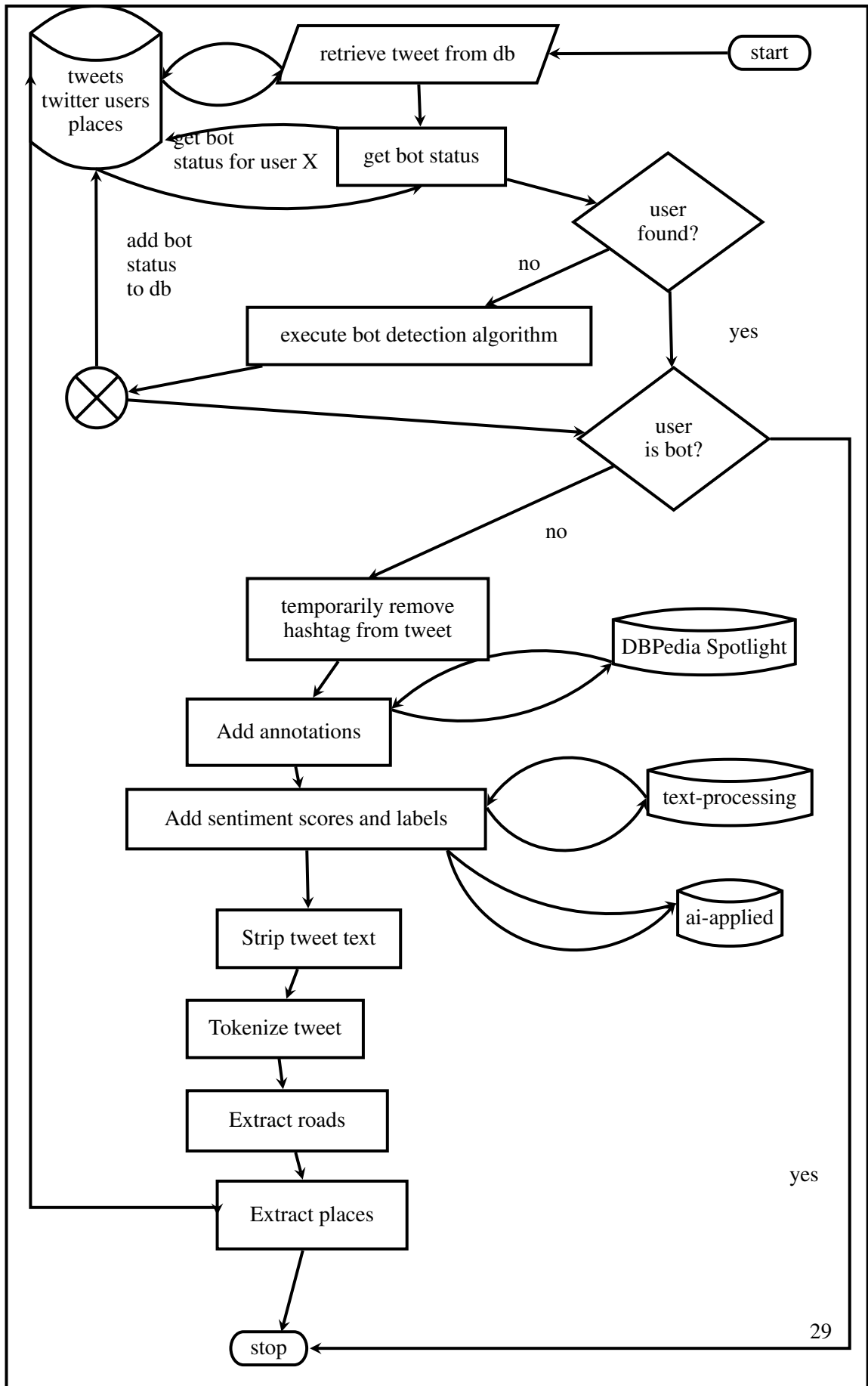


Figure 4.2: High level overview of the extraction steps

4.3 Linkage

In the linkage phase all retrieved data comes together: tweets are connected to events using information from the VILD tables. When we perform a time-based matching from tweets to events it is expected that only a very small fraction of tweets is actually saying something about the event. Many tweets will not be about the matched event or are not even traffic related at all. In the extraction phase we extracted roads and places from our collected tweets. This information seems valuable and usable to link tweets to the appropriate events. However, the information that is stored in the VILD tables, which also includes road numbers and place names, is even richer and therefore probably even more suitable to use for linkage.

As described in section 2.2 the location of events in the EI dataset are besides precise coordinates also described relative to points in the traffic information location database or VILD table. A location of an event is described as a *point*, *link* or *area* location. The area location is used to describe weather events like *fog* in a larger area in the country. ‘Normal’ traffic events, like casual rush hour jams and accidents, are described as point or link location. Often when a traffic event, like an accident, happens on the road it is first described as a *point event*; indicating the accident *itself*. If the accident results in a significant traffic jam the traffic event becomes a *link event* and is described by two points: the start and end of the event. Each location contains references to *VILD points*. As said before the information that these VILD points provide, together with a basic time filtering, can be used to filter potential relevant tweets for the event. The VILD tables are discussed in more detail in section 2.2.

In the following sections we describe different linkage strategies that are derived from the principal of matching using time and VILD points. We start the following section with the most basic variant: ‘the VILD strategy’.

VILD strategy

An overview of the VILD strategy steps are described in figure 4.3. We start by retrieving an event from our database. For this event we want to link tweets that are created between the start- and end time of the event. We collect the following *VILD terms* or *VILD properties* from the VILD points to use for our matching:

loc_des the location description that describes the *type* of the point. Examples: viaduct, exit, bridge.

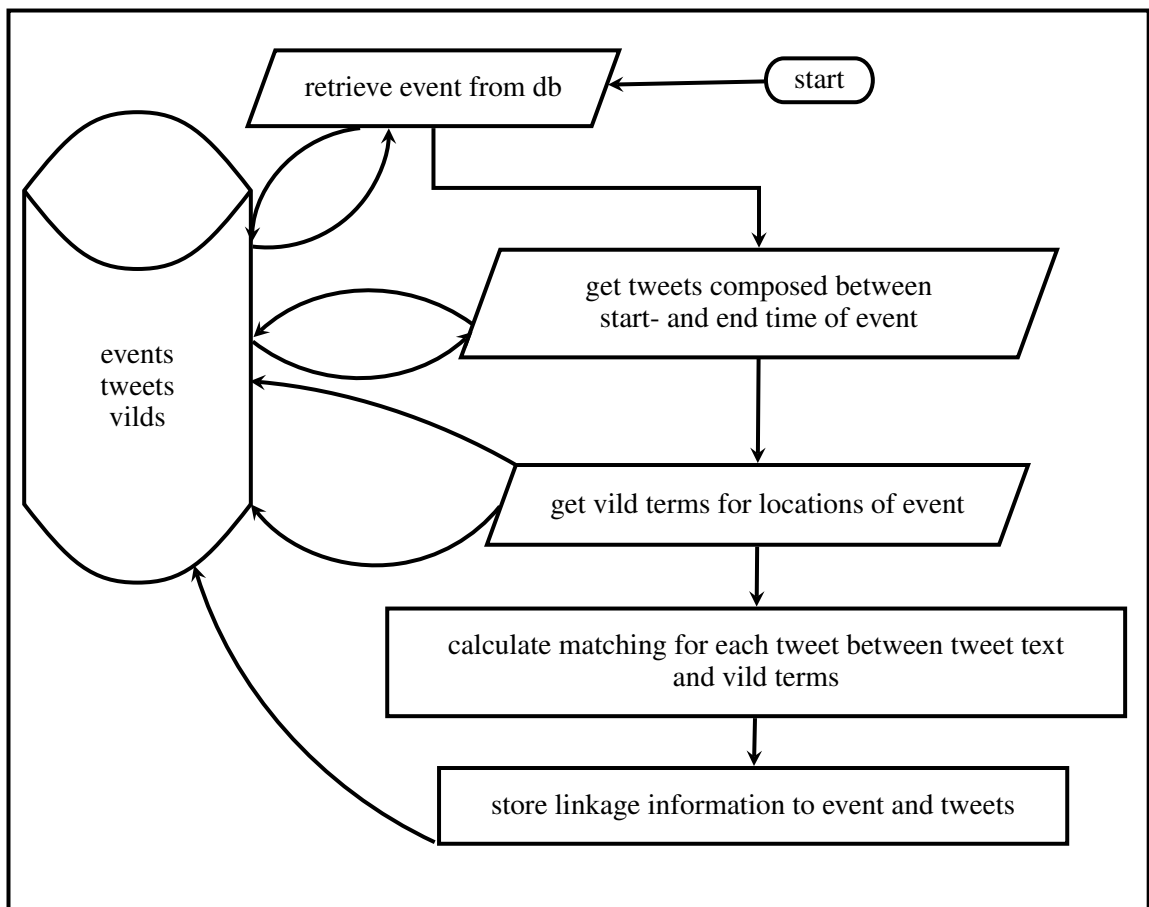
first_name The primary name of the point. The name of a bridge, viaduct, exit, etc.

secnd_name The secondary name of the point. This name is not always used. Most of the time it describes the name of an intersecting road or the name of the road the primary point is leading to (in case the point is an exit).

roadnumber The roadnumber. Can be empty. Examples: A13, A4, A16, N210.

roadname The name of a road. Can be empty. Examples: Ring Rotterdam, A16 Hoofdrijbaan, Traverse Maastricht.

Figure 4.3: High level overview of the linkage steps for the VILD strategy



The next step of the linkage phase is the calculation of the matching between the VILD terms and the tweet text. An example of a match percentage of 75% is given in 4.1.

Table 4.1: Example of the match percentage calculation

Tweet text	'Groot ongeluk op de A13 vlakbij afrit Tu Delft'
Vild terms	'Afrut', 'A13', 'TUDelft', 'N470'
Matching terms	'A13', 'TUDelft', 'afrit'
Match percentage	75%

An important note to place here is that we do not expect that the actual number of the match percentage (75% in the previous example) is a performance measure in some way. It is not related to the chance that the tweet is actually about the linked event, that it is more interesting than a tweet with a lower match percentage, or whatsoever. The only assumption we make is that as soon as the match percentage is higher than zero we believe it has a high chance (> 50% ?) that the tweet is about the linked event and an even higher chance (> 75% ?) that it is at least about *some* traffic event. In chapter 6 we discuss the performance of the linkage strategies to see if this assumption is justified.

After the match percentage has been calculated the match information is added to both the tweet and the event. We also store the information for tweets that have a match percentage of 0%. Since these tweets are still composed during the time the event was active, it is still possible that it is a *related tweet* e.g. has something to do with the event. When we construct a ground truth, needed to evaluate the performance of our linkage strategies, we also need to consider those tweets. The Dutch preposition 'van' occurs quite a lot in the VILD tables. An example of a term containing these preposition is 'van Brienoord'; a bridge in Rotterdam. People are also referring to this bridge as 'Brienoord', without the preposition. When we search for the precise terms in the tweet texts, tweets that do not include the preposition will be ignored. This will reduce the recall of our methodology. To prevent this loss we also include the term without the preposition. Just removing the 'van' preposition and only put the result in the set is not a good solution, because the preposition can also be in the middle ('poort van Groningen').

The pseudo-code of the VILD strategy is described in algorithm 2.

full-VILD strategy

Linking tweets to events using VILD terms in combination with a time constraint seems a good starting approach to find the right tweets. Now consider the following situation. We have an event that starts at point *A*, ends at point *C* and crosses point *B*. We have a tweet about the accident related to the event that is composed at point *B* and mentions the *first name* of the corresponding VILD point. Since the previous described strategy only retrieves the VILD points of *A* and *B*, the start and end point of the event, the tweet will not match. To deal with this problem and increase the recall of our linkage strategy we developed the 'full-VILD strategy'. Do not only include the terms from the start- and end point of the event, but add the series of all points on the route of the event, including the start- and end point. In this way the tweet of the

```

foreach traffic event  $e$  do
  Find tweets  $t$  with creation time:  $e_{start} \leq t_{createdAtDate} \leq e_{end}$ 
  forall the VILD points  $v_i \in V$  mentioned in event  $e$  do
    retrieve the loc_des, first_name, secnd_name, roadnumber and
    roadname properties describing  $v_i$  using the VILD tables.
    If a property contains the preposition ‘van’, add property without ‘van’
    as additional property
    Calculate the match percentage between the unique properties from the
    previous step and the tweet texts, for all tweets  $t$ .
  end
  \\ Make cross links for tweets and events
  Add linked tweet to event (together with match percentage score)
  Add linked event to tweet (together with match percentage score)
end

```

Algorithm 2: Linkage algorithm

previous example will be matched. In general we expect an increase in recall by using this strategy. The idea of the full-VILD strategy is given in figure 4.4.

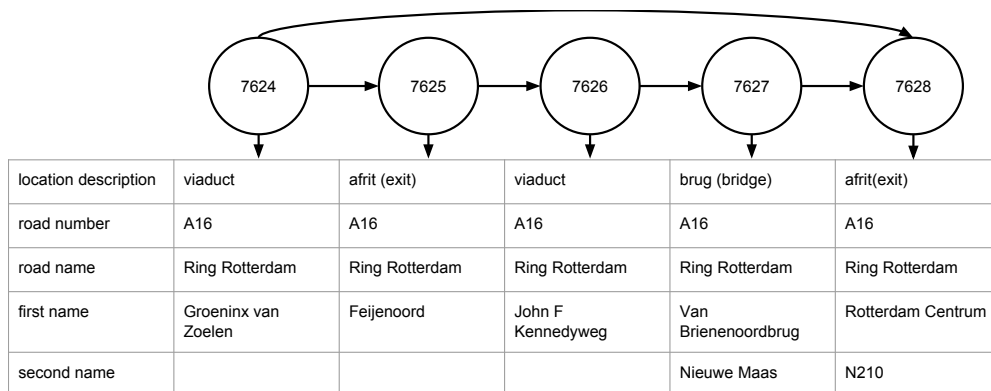


Figure 4.4: full-VILD strategy

The original event starts at VILD point 7624 and ends at 7628. The complete chain of VILD points this events passes is: $\{7624, 7625, 7626, 7627, 7628\}$, as illustrated in figure 4.4. The table beneath the VILD points displays the corresponding properties that we use to calculate the matching. The unique set of terms is the following:

$$\cup_{fullVILD} terms = \{\text{viaduct, afrit, brug, A16, Ring Rotterdam, Groeninx van Zoelen, Feijenoord, John F Kennedyweg, Van Brienoordbrug, Rotterdam Centrum, Nieuwe Maas, N210}\}$$

The additional set of terms that is used with respect to the VILD-strategy is the following:

$$\cup \text{fullVILD}_{terms} - \cup \text{VILD}_{terms} = \{\text{brug, Feijenoord, John F Kennedyweg, Van Brienenoordbrug, Nieuwe Maas}\}$$

In this example the additional 5 terms potentially increases the recall of the linked tweets. The drawback of the increase of recall could be a decrease in the *precision* of our methodology. In chapter 6, where we discuss the performed experiments, we evaluate the impact on the precision.

full-VILD+ strategy

Besides the possible loss in precision of the full-VILD strategy the time-constraint from the start (e_{start}) to the end (e_{end}) of an event also can form a limitation to the performance of the (full)-VILD linkage strategy. The problem of the time filter is that we are only able to link *direct tweets*: tweets that are composed *during an event*. The events that we use from the EI dataset are *detected events*: events that are in some way spotted and registered. As soon as an event is registered for the first time it gets the start time that we use in our methodology: e_{start} . The *actual event* occurs a little bit earlier, since it costs some time to *detect* and *register* the event. With the (full)-VILD linkage strategy we are not able to find tweets that are composed between the start of the actual event and the start of the detected event:

$$\forall t \in T | e_{actStart} \leq t_{createdAtDate} < e_{detectStart}$$

In this formula is T the set of all collected tweets, $createdAtDate$ the creation time of a tweet, $e_{actStart}$ the start time of the actual event and $e_{detectStart}$ the start time of the detected event as described in the EI dataset.

Besides tweets composed between $e_{actStart}$ and $e_{detectStart}$ there also exists tweets about events *before* they *actually* happen. Consider for example the following tweet:

@vid @ A1 links thv hmp 92.4 loslopende honden op snelweg beginnende file. (@vid @A1 left nearby hmp 92.4 stray dogs on highway: starting traffic jam)

In case this situation is actually evolved into a serious traffic disturbance we could say that this tweet is of a *predictable kind*: it is about a traffic event that did not start yet. It is not expected that we received many tweets of this kind. Because the complexity and availability of this kind of tweets they are beyond the scope of our research.

Besides the tweets composed before the actual event and the tweets composed between the actual and the detected event there is a third kind of tweets that our current linkage strategies can not detect. This is the kind where people tweet about a traffic disturbance, *after* it happened. People tweet about accidents they have seen or the jams they were into at a later point in time; when they are done driving and reached the place of destination. By that time, the actual event might be over. Because our goal is to enrich traffic events in a more real time setup, e.g. at the moment the event

is *valid*, this kind of tweets are less interesting. Therefore our methodology disregards this kind of tweets.

Of the three different types of tweets we just described that is not yet covered by a linkage strategy, the interesting one for our methodology is the first one: tweets composed between the start of the actual event and the start of the detected event. Therefore we introduce the full-VILD+ strategy: an extension to the full-VILD strategy. We still use all chained VILD points between the start- and end location of the event, but we also include tweets that are composed 30 minutes before e_{start} .

In chapter 6 we will experiment with the different linkage strategies and investigate the consequences of the increased time frames.

4.4 Classification

The result of the linkage step, regardless of the used linkage strategy, is that a subset of all retrieved tweets is linked to events and each linked tweet has a match percentage between 0 and 100 with ‘the event’. Each linked tweet is composed by either a bot or real user. Furthermore the tweet can be related to the linked event, related to another traffic event or not related to traffic at all. Combining these properties gives us the following 5 combinations, as displayed in table 4.2: $\{related - real, related - bot, other - real, other - bot, not - related\}$. Tweets not related to traffic are, by definition, not interesting for our research. There we do not distinguish between ‘not-related-real’ and ‘not-related-bot’, but just merge those two into the ‘not-related’-category.

Table 4.2: Possibilities for linked tweets (candidates)

	related to linked traffic event	related to other traffic event	not related to traffic at all / unknown
tweet from ‘real user’	<i>related - real</i>	<i>other - real</i>	<i>not - related</i>
tweet from ‘bot’	<i>related - bot</i>	<i>other - bot</i>	<i>not - related</i>

In the classification step we try to *characterize* traffic events by *deriving* information from tweets of the *related-real* category. We try to determine the cause type of a traffic event based on the information available in the related-real tweets. The cause type of an event is something that cannot be directly derived from the open traffic data and plays a very important role in the estimation of the *severity* of a traffic event. We created a finite set of possible cause types. This set is displayed in table 4.4.

The high level overview of the classification phase is given in figure 4.5. We start by retrieving a linked event from our database. This event has, because of the previous linkage phase, zero or more connected tweets. Each tweet is of exactly one of the categories described in 4.2. We make the assumption that the useful *related-real* tweets have a match percentage higher than zero. We collect all tweet with a match percentage > 0 . We strip their tweet texts by removing all punctuation symbols: dots, hashtags, etc.

The next step in the classification phase is retrieving an ordered set of *traffic terms* from our database. This set forms our *traffic dictionary*. We will construct this dictio-

nary semi-manual by investigating the tweets that are linked to events by the linkage module of our developed system. The ordered set maps traffic cause types to related traffic terms.

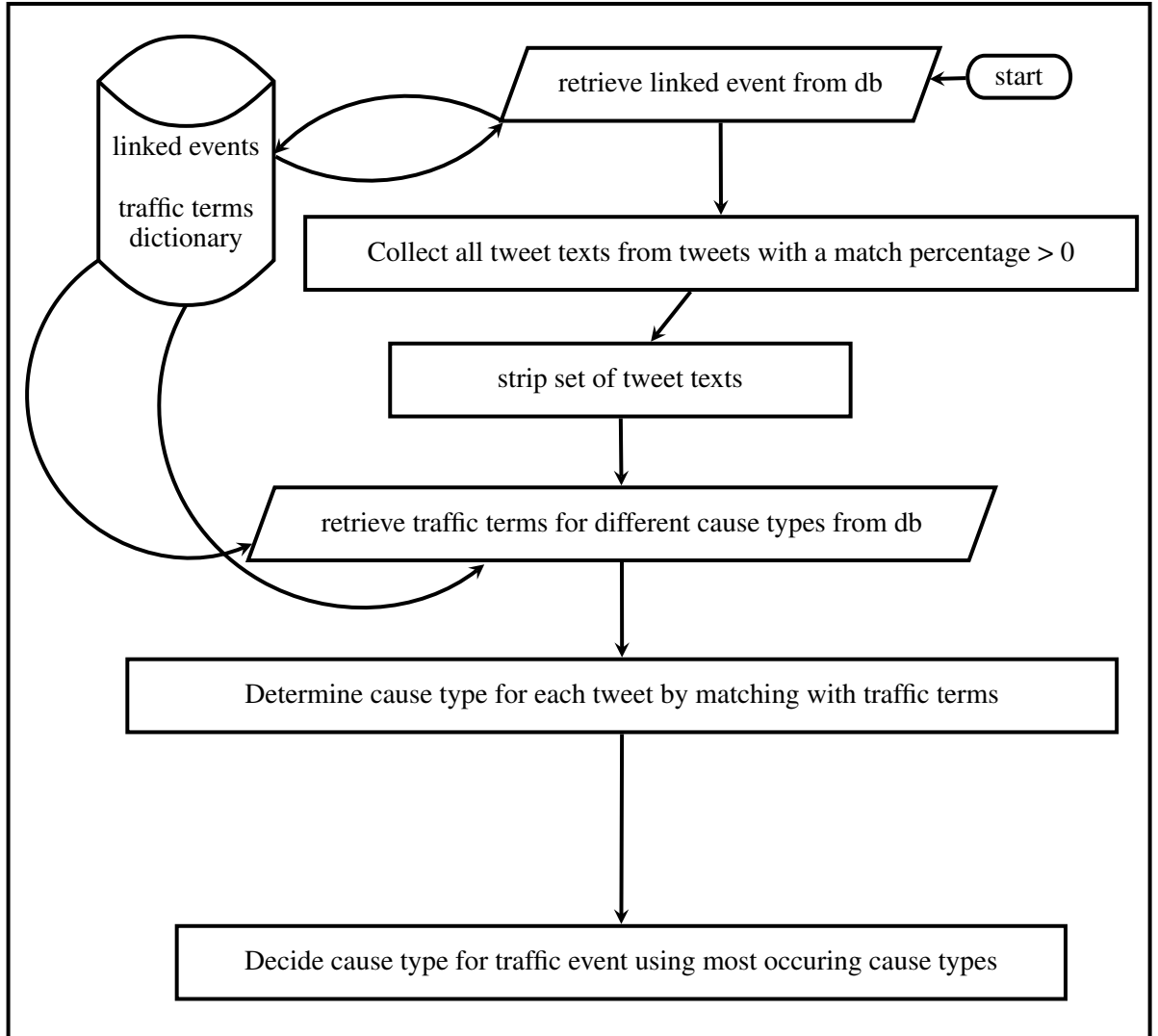


Figure 4.5: High level overview of the classification steps

An minimal example of an ordered set of traffic terms is the following:

accident {accident, crash}

technical related road problem {matrix, open bridge}

heavy weather {snow, rain, mist}

The cause type for each tweet is estimated by matching the stripped tweet text with traffic terms from the dictionary. The most occurring cause type is used as final cause type decision for the classification.

tweet 1	What a major <i>accident</i> at the <u>A13</u> , near exit <u>Delft</u>
tweet 2	Car <i>accident</i> at # <u>A13</u> [photo]

Table 4.3: Example of two ‘accident tweets’. Matching terms are underlined, traffic terms of dictionary are italic

Table 4.4: Set of cause types

Cause type key	Cause type	Examples
rush-hour	(casual) rush hour jam	
accident	(earlier) accident	
event	Event	Festival
non-technical	Non technical related road problems	Mud or oil on road Animals on road
technical	Technical related road problems	A bridge that is not able to close Matrix signs that do not function correctly
construction	Construction zones	Construction zones / road works
weather	Bad weather	Snow Many rain mist
breakdown	Breakdowns	Car breakdowns / defect vehicles
other	Other	A causetype not in any of the previous categories
unknown	Not applicable / unknown	Tweet is not about congestion or cause type is not included

4.5 Conclusion

In this chapter we discussed our proposed methodology to characterize traffic events by using social media. In the next chapter we discuss our developed demonstration system: an implementation of the methodology. Chapter 6 than discusses the performance of several parts of this system.

Chapter 5

Implementation

In this chapter we discuss the technical details of the developed systems. All collected data is stored into Apache CouchDB databases¹. We started to deploy the software on a Virtual Private Server (VPS), but switched to local development and deployment later on in the research to increase performance.

CouchDB allows you to create *CouchApps* that enable you to create web applications that are stored within the same database as where your data lives. We created a CouchApp to monitor our data flows during the collection phase, to analyse our data afterwards and to perform our linked events experiments. With Couch you can create *views*: advanced indexes of your data. We created many indexes for our data: simple ones and more complex ones. We constructed several user interfaces within the CouchApp. Besides the CouchApp we also constructed several other scripts in the PHP and Python language to collect and link data.

In the first section we describe the architectural overview of the developed system. In the other sections we describe the details of the different modules.

5.1 Architectural overview of developed system

The architectural design of the developed system is illustrated in figure 5.1. The vertical line between the different modules separates the *presentation layer* or *front-end* from the *data access layer* or *back-end*.

¹<http://couchdb.apache.org/>

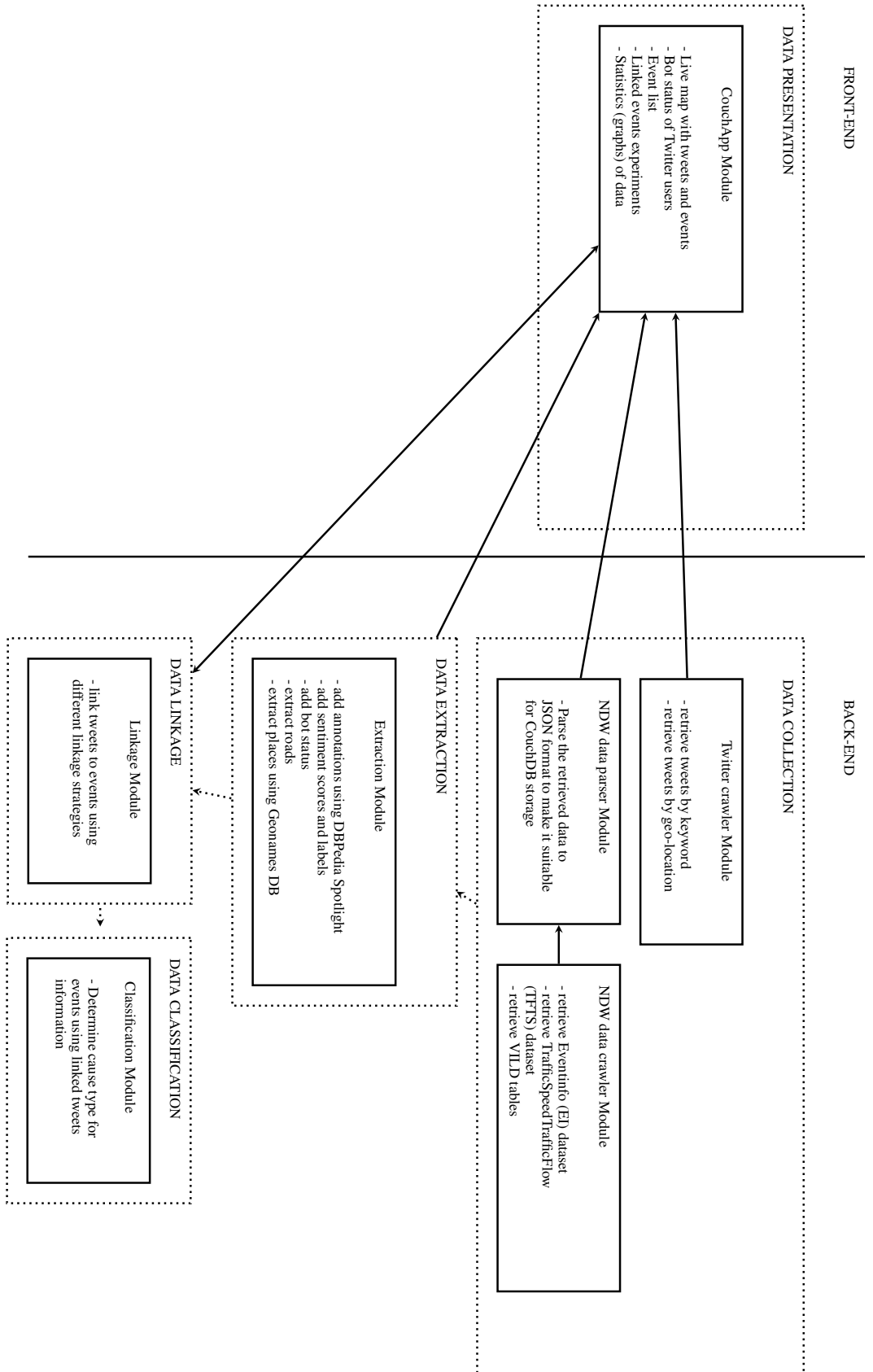


Figure 5.1: Architectural design of the developed system

5.2 Data presentation

The presentation layer is powered by the CouchApp which has a (local) website as front-end to display the collected data and to perform analysis with. As can be seen in figure 5.2 the developed web-interface has 7 different tabs, each with an other purpose.

5.2.1 Map

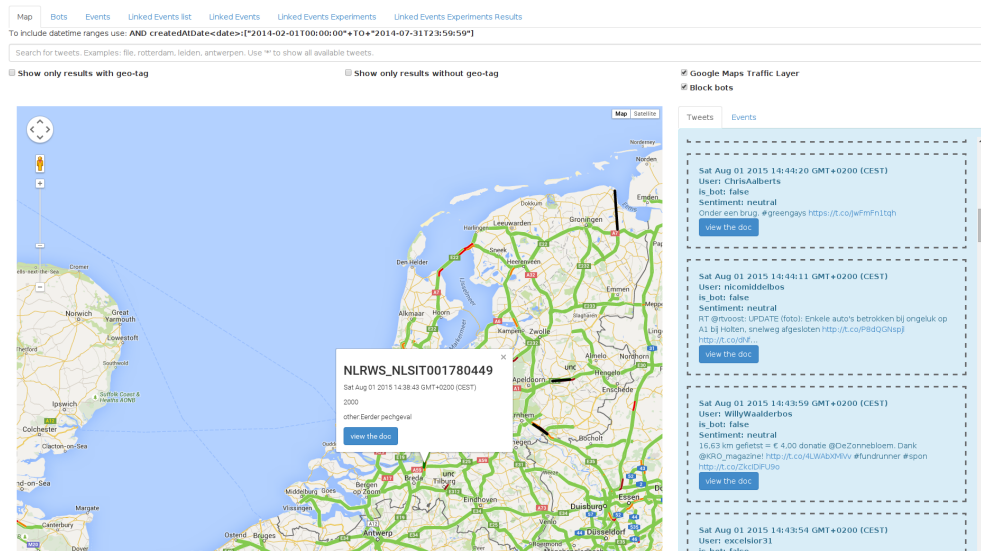


Figure 5.2: Live map with data.

The first tab, map, illustrated in figure 5.2, is mainly used to display (live) events from the Eventinfo (EI) dataset from NDW and to display tweets. This panel was created and used during the Twitter detection approach, as explained in the first part of the thesis. Furthermore in the current methodology it was used during the collection and extraction & analysis phases. It was a powerful tool to retrieve (first) insights in the retrieved data. Where are crowded regions in the country? How do traffic disturbances develop throughout a (common) work day?

Tweets that are geo-tagged can be plotted directly to the map. Tweets without a geo-location will occur only in the right column of the panel. During the collection phase tweets that were retrieved by the Twitter crawler module and stored into the Couch database were almost instantly visible in the web interface thanks to the changes feeds that CouchDB offers². Thanks to the attached Lucene indexer we could search for tweets by creation date and by tweet texts. We also built in a feature to perform a geo bounding box search which enabled us to quickly reveal all tweets composed at a specific place during a certain amount of time.

We created filtering options to filter geo- and non-geotagged tweets and to block tweets composed by bots using the results of the bot-detection algorithm described in section 4.2.1. Together with the real-time behaviour of the system and the attached

²<http://docs.couchdb.org/en/latest/api/database/changes.html>

Lucene indexer the tool was very useful to monitor traffic jams and tweets during rush-hour: we enabled the ‘block bots’ filter, searched for ‘file’ (traffic jam) and monitored the incoming tweets that full-filled this condition. This gave us great first insights about the kind of information people talk about when they are in a traffic jam.

5.2.2 Bots

The bots panel is a simple panel that displays the bot status for the collected Twitter users. It is a simple table with 4 different columns: *screen_name*, *is bot?*, *reason* and *recent tweets*. The *screen_name* column displays the Twitter user names available in our database. The second column, *is bot*, displays the boolean value *true* or *false* indicating if the system identified the user as a bot or not. This bot status is determined once for every user: as soon as the first tweet composed by this user is crawled by our Twitter crawler Module. The *reason* column can have three different values: *null*, *screen_name* and *recent tweets*. ‘Null’, when the user is not identified as bot. The ‘*screen_name*’ value is used when the Twitter user is identified as bot based on suspicious terms in the *screen_name*. The last value, ‘*recent tweets*’, is used when the user is identified as bot based on its 25 most recent tweets. Too many of the most recent tweets contain query terms, making them too suspicious to be composed by *real users*. More details about the bot detection algorithm can be found in section 4.2.1.

5.2.3 Events

The events panel gives a list of the retrieved events from the EI dataset. When the NDW data crawler- and data parser modules are active this list is automatically refreshed as soon as a new EI dataset is retrieved and parsed. The events panel displays the following information in a table view: the situation version time of the event, the *situationId* of the event, the length of the queue (if available) and the cause description of the event. Furthermore each event has a ‘view the doc’ button. When clicking this button all parsed data for the particular event is shown in a pop-up window. This data consists of all available data for the event. An example of this data can be found in appendix A.5.

5.2.4 Linked Events list

The linked events list panel was a panel built for visualising the first results of the linkage module. It is very similar to the events panel with as only difference an additional column which list the VILD terms of the event and for every linked tweet the matching terms and the tweet content. The VILD terms consist of the location description, the first- and second name and the road number. More information about the VILD terms is available in section 2.2 and in section 4.3.

5.2.5 Linked Events

The linked events panel is the graphical version of the linked events list panel. It displays the linked events on a map in the same way that ‘normal events’ are plotted in the map panel (figure 5.2). When clicking on a linked event a pop-up windows is

shown that displays the tweets that are linked to the event. For each tweet the raw data can be viewed by clicking on the 'view the doc button'.

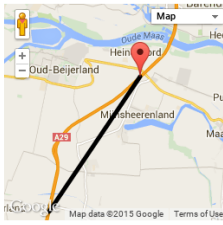
5.2.6 Linked Events Experiments

The linked events experiments tab differs from all other tabs in the sense that the data is not only displayed, but also extended by user interactions with the system. The panel is used to construct the ground truth for the linkage experiments and to test the different linkage strategies. The linked events experiments panel is displayed in figure 5.3.

Linked events experiments

- Skip events with same situation id (use latest)
- Skip events without linked tweets
- Skip events with only '0%' matches
- Skip events from bots
- Skip events that are fully annotated (task complete)

NLRWS_NLSIT001614127 | 47cd799d533855868232b0f43f62381f | Events skipped: 1275 (unique: 44)



Ongevalsonderzoek (NonManagedCause) | certain | undefined

[view the doc](#)

overallStartTime	Wed May 21 2014 14:43:00 GMT+0200 (CEST)
situationRecordVersionTime	Wed May 21 2014 16:24:05 GMT+0200 (CEST)
overallEndTime	Thu May 22 2014 16:24:00 GMT+0200 (CEST)
queueLength	4000
vildTerms:	
- first_name	["bormmelskoussedijk", "mijnshereerland", "groene weg", "hm 89.9 = 23.3", "oud-beijerland", "oude kreek", "stougjesdijk", "moerkerken", "numansdorp"]
- secnd_name	["n217", "n487"]
- loc_des	["viaduct", "parkeerplaats", "hectometersprong", "afrit"]
- roadname	[]
- roadnumber	["a29"]

Search map

[Previous event](#) [Next event](#)

Linked tweets

thijsblom

469096445552103424_TWEET

is_bot:false | false (parsetime | twitter user db)

time:Wed May 21 2014 14:44:42 GMT+0200 (CEST)

Matches:n217

Verkeer aan zuidkant Heineoordtunnel wordt via N217 naar gratis Kiltunnel geleid http://t.co/Z31M8t8xi

- containsPicture Yes No
- isRelated Yes No
- containsUrl Yes No
- isActualBot Yes No
- isRelatedToOtherTrafficEvent Yes No
- causeTypeAccordingToTweet
 - (casual) rush hour jam
 - (earlier) accident
 - Event: people going to a festival or theme park
 - Non technical related road problems: mur or oil on road, animals on road
 - Technical related road problems: bridge not able to close, malfunctioning matrix signs
 - Bad weather: snow, many rain, fog
 - Other
 - Not applicable / unknown

[Done](#) [view the doc](#)

Figure 5.3: Link tweets to events experiments interface

5.3 Data collection

The topmost block on the right side of figure 5.1 is the data collection layer. The modules in this layer must make sure that all required data, as described in section 4.1, is gathered. The Twitter crawler Module is responsible for collecting all tweets needed for the other phases. The advantage of the Twitter Streaming API's³ is that they are able to return the data in the JSON file-format, making it very easy to store into our Couch database. The open traffic data can not be retrieved in the JSON-file format and therefore we need to *convert* this data in order to save the data in our database. The NDW data parser module is responsible for this conversion. In the next sections we discuss the implementation details of the different data collection modules.

5.3.1 Twitter crawler Module

The Twitter crawler module is written in the Python2 language⁴. We used the *Twython* package⁵ to easily setup a connection with Twitter and retrieve tweets. The *Twython-Streamer* module of the Twython package enables you to directly communicate with the Streaming API's of Twitter. Using the Twython package we were able to create two different Twitter crawlers: a *keyword crawler* and a *geo crawler*. The final version of the keyword crawler crawls the Twitter streaming API with the following set of keywords:

```
{ file (traffic jam), ongeluk (accident), pech (failure), brug (bridge), langzaam
rijden (slow traffic), traag rijden (tardy traffic), spits(rush-hour), km, verkeer (traf-
fic), gekanteld (tilted), gekantelde (tilted), aanrijding (collision)}.
```

Because of the used Twython package crawling the keyword streaming API of Twitter is as easy as executing a few lines of code, as displayed in code block 5.1.

```
1 stream.statuses.filter(
2   track="file,ongeluk,pech,brug,langzaam rijden,traag rijden,spits,km,verkeer,gekanteld,gekantelde,
   aanrijding",
3   language="nl",
4 )
```

Code 5.1: Performing a keyword search using the Twitter streaming API and the Twython package.

For our geo crawler we need to provide a rectangular bounding box. We choose our four coordinates in such a way that the whole Netherlands is covered. As described in literature we expect that we receive at most 1% of all tweets that are actually composed in this area[4]. To perform a geo-based search on Twitter is again fairly easy when using a library, as shown in 5.2.

³<https://dev.twitter.com/docs/streaming-apis>

⁴<https://www.python.org/>

⁵<https://twython.readthedocs.org/en/latest/>

```

1 stream.statuses.filter(
2     locations=GEOCODE
3 )

```

Code 5.2: Performing a geo-based search using the Twitter streaming API and the Twython package. GEOCODE is defined as '3.0322265625 50.54747185651962 7.53662109375 53.67637481706787'

5.3.2 NDW data crawler- and data parser modules

The NDW crawler module is responsible for collecting the open traffic data that the NDW is distributing. Most of the distributed data comes in the XML data format. Since we decided to store our data into a Couch database, which stores documents in the JavaScript Object Notation data format (JSON⁶), we need to *parse* and *convert* the NDW data to make storage possible. The NDW data parser Module is responsible for this parsing and conversion.

The raw NDW datasets are available on a FTP server⁷. The XML datasets are stored on this server in compressed format to minimize the download time from the server to other computers. For comparison: 'measurement.gz', containing information about all traffic measurement points is 2.9 megabytes in compressed form, whereas the uncompressed or unpacked version is 151 megabytes. The raw datasets are crawled using a simple PHP script. The script monitors the last modification dates of the datasets on the FTP server. As soon as a datasets modification date has changed the script starts to download the new version of the dataset. The crawled datasets are picked up by the parser module which transforms all data into the JSON format and stores the data into the databases.

In the following sections we provide details about the most important datasets for our system: the VILD- and the EI datasets.

VILD

The VILD tables, which come as semicolon separated CSV files with a file size of just a few megabytes are parsed using the Python language. To convert the rows in the CSV-files to JSON objects we use the table *header row* as *keys* for the object and the body rows as *values*. After each bulk conversion of 50 rows into JSON objects we save a set of parsed VILD points into our Couch database. Examples of the VILD data are given in Appendix A.4. Codeblock A.4 shows a few rows of the raw data as delivered by the NDW. Codeblock A.5 is an example of how a converted row looks like in the JSON format.

EI

As explained in section 2.2 the Eventinfo or EI dataset covers all kind of *traffic events* that are *happening* on the road around the time the events are being distributed. The distribution of the events in the EI set are of a real time behaviour and therefore very

⁶<http://json.org/>

⁷At time of writing the following server is used: <ftp://83.247.110.3/>

useful for our experiments to use as a ground truth when we are searching tweets for reasoning about events. An example of how an EI dataset looks like is provided in appendix A.6. We initially started parsing events using a parser written in the PHP language. There are a couple of difficulties to overcome when parsing XML files from the NDW. First of all many "out of the box" libraries for the PHP language operate by loading the complete XML tree into the memory. Since many files are quite big, this is a memory intensive operation that requires many resources. Since these resources are not always available it is wiser to parse the relatively big XML files 'line by line'. We therefore written or own parser. Furthermore there is no standard 'rule' for parsing XML into JSON: there are multiple possibilities. We choose a common used method that is intuitive and consistent.

To illustrate this conversion an example of a piece of the EI dataset and its converted JSON counterpart are given in code block 5.3 and 5.4.

Code 5.3: A small piece of XML in an EI dataset

```
1 <values>
2 <value lang="nl">NLRWS</value>
3 </values>
```

Code 5.4: The conversion of XML into JSON of a piece of an event

```
1 {
2   "nodes": [
3     {
4       "nodes": [
5         "NLRWS"
6       ],
7       "name": "value",
8       "attrs": {
9         "lang": "nl"
10      }
11    }
12  ],
13  "name": "values",
14  "attrs": {}
15 }
```

This resulted in quite a verbose JSON object with lots of nesting and hidden data, as you can already see from the previous simple example. It is time inefficient when you have to search your whole data object each time that you are looking for a particular property of your data. To make some properties of events more accessible we implemented a feature in the parser that made important properties direct accessible under 'special keys'. This enabled us to get quick access to properties as creation time, latitude, longitude and cause types. When we needed more data of the EI dataset to be quickly accessible we rewrote the parser. This became a very time consuming job, because we needed more and more properties. Furthermore it resulted in inconsistent data, because we were already collecting data while the parser changed over time. Therefore we decided to write a 're parser' in Python that made all properties directly

available. We started with the data available under the ‘nodes’ part of the previous created JSON objects and could therefore maintain the data that had been already parsed. The new JSON structure of our stored events made them much more practical to work with.

5.4 Data extraction

The data extraction module was created to obtain first insights in the retrieved data. To extract entities from the tweet texts and get a better understanding of what people talk about we annotated a part of our retrieved tweets using DBPedia Spotlight for the Dutch language. Furthermore we added sentiments scores and labels to get insights on how people react to different traffic circumstances using two different services. Finally, in the data extraction module we extract roads and places from tweet contents and add a ‘bot status’-flag for each Twitter user from who we retrieved at least one tweet. All data extraction features are implemented using the Python language. In the following sections we discuss some noteworthy details about the different features.

5.4.1 Annotations

DBPedia Spotlight is an annotation service that is able to find different kind of entities for the Dutch language. We use two different endpoints⁸ to annotate our tweets. The first endpoint identifies entities and provide the offset of the occurrence of the entity in the provided text. The second endpoint provides a list of candidate entities for the given input text. The endpoints are able to present their results in JSON format, which makes them perfect to be stored along the tweet data documents themselves, as they are in JSON too, thanks to our Couch DB storage engine. The results of the different services on the following tweet text can be found in appendix B.

@fileinformatie_ Morning! Afrit #28 naar #A1 #hoevelaken > A'dam is een drama. A1 bij 41,5 gaat nu weer rijden.. langzaam.. ANWBverkeer

During our research we did not notice any API restriction or limitation of DBPedia Spotlight: it was able to annotate all the tweets we put in. The result of the annotations are provided in chapter 6.

5.4.2 Sentiments scores and labels

To get insights on how people react to different traffic circumstances we added sentiment scores and labels to our data. We found two semi-free services for the Dutch language: text-processing⁹ and AI Applied¹⁰. Both services return their results in JSON, demonstrated in appendix C.

⁸<http://nl.dbpedia.org/spotlight/rest/spot?spotter=SPOTTER&text=TEXT> and <http://nl.dbpedia.org/spotlight/rest/candidates?text=TEXT> are used to annotate Dutch tweets

⁹<http://text-processing.com/api/sentiment/>

¹⁰http://api.ai-applied.nl/api/sentiment_api/

AI-applied provides one *sentiment label* or *class* per tweet text, together with a confidential score. Text-processing provides sentiment scores for all three labels: negative, neutral and positive. The highest score determines the final sentiment label.

The advantage of having multiple services for the same purpose is that you can compare the results of the services on your dataset. The comparison can increase or decrease the confidence one has in a service.

5.4.3 Bot status

As explained in section 4.2 a bot detection algorithm is needed in order to be able to filter out tweets composed by traffic news- or media providers. The bot status for each unique Twitter user is determined ones by the bot detection algorithm. The status of a Twitter user is saved into our CouchDB database after is has been determined. The following code block demonstrates a Twitter user that is classified as bot, because it has a ‘suspicious screen name’. The bot detection algorithm is described in algorithm 1.

Code 5.5: Example of a Twitter user database document that is classified as ‘bot’

```

1 {
2   "_id": "news1117_twitter_user",
3   "_rev": "1-258eb707365aa7294a428d51f73d7d24",
4   "screen_name": "news1117",
5   "is_bot": true,
6   "reason": "screen_name",
7   "time": 1398873447378,
8   "tweets": [
9     ],
10  "type": "twitter_user"
11 }
```

As document `_id` we use the Twitter screen name concatenated with the string ‘`_twitter_user`’. Since Twitter screen names are unique it is impossible to get conflicting documents in our database when storing Twitter users¹¹. Besides the bot status itself we also store information about the time the bot status is saved into the database, the type of the document, the reason why a user is classified as bot and potentially the 25 most recent tweets when a user is classified as bot based on too much occurrences of suspicious terms in its most recent tweets. In the given example the tweets array is empty, because the user was already classified as bot based on its screen name. The suspicious terms are saved in a separate Couch document:

Code 5.6: Suspicious keywords that lead to ‘bot status is true’ when used in Twitter screen names

```

1 {
2   "_id": "BOT_KEYWORDS",
3   "keywords": [
4     "file",
```

¹¹Each document in a CouchDB database needs to have a unique document id

```

5     "verkeer",
6     "news",
7     "nws",
8     "nieuws",
9     "weer",
10    "112",
11    "headline"
12  ]
13 }

```

5.4.4 Roads

We wanted to extract roads from tweet texts, because during manual exploration we got the presumption that many interesting traffic related tweets contained references to road numbers. By pattern match each retrieved tweet to the regular expression $[A|N][0-9]^+$ we were able to enrich the data with arrays of road numbers that start with either an A (all highways in the Netherlands) or and N (provincial roads) and is followed by at least one number. To detect all kind of appearances of these roads, such as #N24, a12 and A12, we remove punctuation and hash-tag symbols from tweets texts and add the 'IGNORECASE' option to our extraction module.

5.4.5 Places

When places are used in tweet texts they potentially refer to the location the text is talking about. This is particularly handy in an environment with a very low rate of precise locations: the amount of tweets containing a precise geo location in our keyword dataset is just 3.39%. By extracting places from tweets you potentially add a location to them, making them much more valuable. To extract places we used the free available Geonames database¹². On the Geonames website you can download datasets containing names and geographical coordinates of city centres for places of many countries. We downloaded the dataset for the Netherlands and stored all information in our own database. Afterwards we created a CouchDB view to map all places to their corresponding geo location. This gives us a quick accessible index of places.

5.5 Data linkage

It is the task of the linkage module to link tweets to traffic events using four different linkage strategies. The VILD strategies are explained in chapter 4. A couple of mathematical functions were created in order to perform measurements with geo locations. The two most important ones are the 'distance' and the 'boundingBox' functions. The *distance* function takes four input arguments (latitude1, longitude1, latitude2, longitude2), representing two different geographical points, and outputs the distance between the two points in metres. The *boundingBox* function takes three arguments (latitude, longitude and offsetInMetres) as input and calculates and outputs the four coordinates that represent a bounding box around the input point at the given

¹²<http://www.geonames.org/>

input offset. This function is used to create the boundaries for ‘geo tweets’ in order to be linked to a particular event. To increase performance and reduce the time needed to link all our retrieved traffic events to tweets all VILD points and Twitter user bot statuses are loaded into memory at the start of the linkage execution. Furthermore tweets are linked to events in batches of 50 events before the linkage data is saved to our database. Reducing the amount of reading from and writing to the database is a significant improvement on the performance of our linkage. For each event all different linkage strategies are executed after each other, because strategies share metadata that is needed to calculate the matching between tweet texts and VILD terms. A special *state* database document is maintained to keep track of the linkage status. This document contains statistics about performed linkage runs and information about the last linked event. Thanks to the state document the linkage module can continue where it left off in case of system crashes or whatsoever. After many optimizations we were able to link our complete set of 316.649 events to our keyword dataset and geo-dataset consisting of respectively 825.928 and 5.667.04 tweets in approximately 4 hours. The structure of the relevant data part of a traffic event, after it has been linked can be found in appendix D.

The keys of the `linkedTweets` object in structure D.1 represent the different linkage strategies: *geoTweets*, *timeAndVildBased*, *timeAndFullVildBased*, *timeAndFullVildBased+*. The *timeAndFullVildBased+* object is fold out to demonstrate how linked tweets are stored. The other strategy object are collapsed since they have a similar structure. The ‘10%’ key represent the matching percentage of the tweet with the VILD terms and is also available as ‘10’ under the *matchPercentage* key. The *vildTerms* key displays all VILD terms for the event categorised by VILD type: *first_name*, *secnd_name*, etc. *matchTerms* contains the VILD terms that exists in the tweet text of this tweet composed by ‘yourvipdriver’. As can be seen in the structure the matching terms consists of two items for this tweet: "afrit" and "a1". The total number of VILD terms is 14. However, terms in the ‘loc_des’ category are ignored for matching. Therefore ‘afrit’ in *matchTerms* is excluded and the total number of VILD terms is 10 instead of 14. Hence the match percentage for this tweet is $1/10 \times 100\% = 10\%$. The reason for excluding location descriptions (loc_des) was born during experimenting and is not included in the original developed methodology. The reason is explained in section 6.4.3.

All other properties speak for themselves. More information about the function of VILD points can be found in section 2.2: a section about open traffic data. The experiments that use the `linkedTweets` data are described in chapter 6.

5.6 Data classification

The last module of the developed system is the *classification module*. The classification module tries to extract the traffic cause types from tweet texts. The module starts by loading the semi manual built cause dictionary (see 4.4) into memory. This dictionary maps the different cause types (rush-hour, accident, event, non-technical, technical, construction, weather, other and unknown) to a set of related words or word groups. For example *ongeluk* (*accident*) and *aanrijding* (*collision*) are included in the dictionary for the *accident* cause type. After the initialization of the dictionary the

classification module is ready to be fed with (tweet) texts.

Each tweet text of a related-real tweet is parsed to the *getCauseTypesForTweet* function of the classification module. The function starts by removing all punctuation from the tweet text. This removes all hash tags, mentions, etc. The resulting string is more suitable for matching with words and word groups in our dictionary. After the tweet text has been stripped the module searches for matches between the stripped tweet text and words/word groups in the dictionary by using regular expressions. The module generates a set of cause types that matches with the input text. It is possible that a text matches with multiple cause types. So the following set could be a perfect valid set, although it is a strange combination: $\{accident, technical, rush - hour\}$. When no matches are found the set: $\{unknown\}$ is returned for the particular (tweet) text.

The classification module combines the results of the *getCauseTypesForTweet* function for all related-real tweets that are linked by a certain strategy to a specific event. The final cause type for an event is then given by the most occurring cause type.

Chapter 6

Experiments

In this chapter we describe the results and analysis of several experiments that have been done. This involves both quantitative and qualitative results. We start by describing the different datasets that we created for our experiments: two Twitter datasets and the traffic events dataset. More information about the structure of the traffic event dataset can be found in section 2.2. In the remaining sections we describe and analyse the results of the experiments with reference to the different steps of our methodology, as described in chapter 4. In this chapter we will refer to the following list of tweets categories:

{related-real, related-bot, other-real, other-bot, not-related}

This set of possibilities is explained in section 4.4. Furthermore we will refer to different traffic cause types, as explained in 4.4, using the following cause type keys:

{rush-hour, accident, event, non-technical, technical, construction, weather, breakdown, other, unknown}

6.1 Dataset description

6.1.1 Twitter

We collected two different Twitter datasets.

Geo dataset

The first dataset consists of tweets that contain a geo tag somewhere in the Netherlands. For this we used a geographical bounding box search on the Twitter streaming API with the following coordinates: (50.54747185651962,3.0322265625) for the bottom left of the box and (53.67637481706787,7.53662109375) for top right. This box includes some parts of Belgium, France and Germany, but since we only take tweets into account that are close enough to traffic events in our Eventinfo dataset this is not a problem.

Based on the literature[4] we assume that we receive at most 1% of the total amount of geo-tagged tweets composed in the area of interest.

Keyword dataset

The second dataset used consists of tweets collected using a keyword based search on the Twitter streaming API. The initial set of traffic related terms is the following:

{ file (traffic jam), ongeluk (accident), pech (failure), brug (bridge), langzaam rijden (slow traffic), traag rijden (tardy traffic), spits(rush-hour)}.

The most important keywords in this set are ‘file’ (traffic jam) and ‘ongeluk’ (accident). These search words are likely to provide us more relevant tweets about traffic events than the other keywords. In order to increase the set of potentially interesting tweets we added additional keywords to the search set by creating word-clouds from the tweets collected up to a certain amount in time that either contained ‘file’ or ‘ongeluk’. An example of such a word cloud can be found in figure 6.1 The generated



Figure 6.1: Word cloud of size 1000 for the ‘file’ keyword

word-clouds provided us the following set of search words that we added to the system:

{km, verkeer, gekanteld, gekantelde, aanrijding}

6.1.2 Traffic events

For our experiments we make use of the Eventinfo dataset. During this chapter we will refer to this dataset by using the abbreviation *EI*. The traffic events that are described in these datasets consists for the most part of manually entered traffic information. The traffic events are described as situation records in which each unique traffic event on the roads gets a unique situationId. This unique situationId is repeatedly used in successive EI datasets as long as the situation is active or valid. Furthermore a situation record has a version time and an overall start time that we use to bound our events in time. In the linkage phase we combine the location points that each event has with terms that occur in VILD tables. The structure and use of the VILD tables are described in chapter 2. More information about the EI dataset can be found in section 2.2. An example of an EI dataset can be found in appendix A.5.

6.2 Collection

We collected 5.666.878 ‘geo tweets’, 825.928 ‘keyword tweets’ and 315.994 events in the overall period from 22 February 2014 to 4 August 2014. Due to limitations in disk space and the fact that our thoughts about what data we needed changed over time the time span of the different data sets are not the same. This can be seen in figure 6.2. Due to some technical issues with the Python library that we used to crawl tweets using the Twitter streaming API there are some days that we were not able to receive tweets for the complete day or did not receive any tweet at all. The lack of tweets on some days should not be a problem as long as we construct our test datasets wise. However, we should be aware of this fact when analysing the results to prevent us to make wrong conclusions.

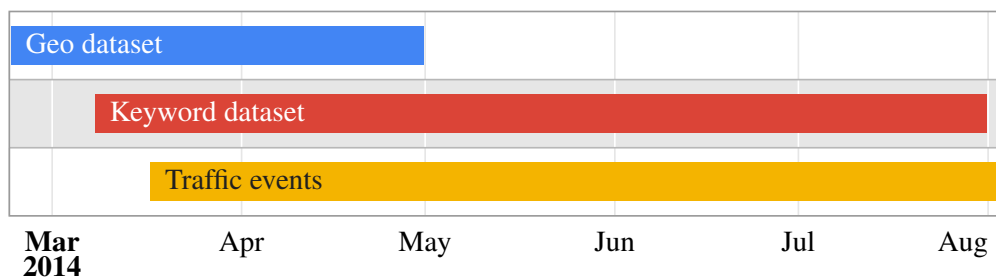


Figure 6.2: Timespan of the created datasets

6.2.1 Quantitative results of Twitter datasets

Table 6.1: Results of geo dataset

February	479.563	Avg per month (Feb 22th - May 1st)	2.537.408(2.698.513)
March	2.499.923	Avg per day (Feb 22th - May 1st)	82.129(87.183)
April	2.651.214	Median per day	90.186(90.560)
May	36.304	Std dev per day	26.064(16.562)
		Min per day	0(20.112)
		Max per day	106.666
	5.666.878		

Table 6.2: Results of keyword dataset

March	121631	Avg per month (Mar 8th - Aug 2nd)	170.882(206.482)
April	231753	Avg per day (Mar 8th - Aug 2nd)	5619(6821)
May	176286	Median per day	7284(7316)
June	161639	Std dev per day	2317(2117)
July	131710	Min per day	0(564)
August	2909	Max per day	10.199
		Geo-tagged tweets	27.960(3.39%)
	825.928		

The quantitative results of our collected tweets are in table 6.1 and 6.2. Due to technical issues of our collection implementation there were days we did not receive any tweets: 4 days for the ‘Geo collector’ and 25 days for the ‘keyword collector’. This explains the *min per day* values of ‘0’ in the tables. To get an idea of the impact of the technical issues on the statistics, we included numbers where we filtered out the ‘issue days’. These numbers are between parenthesis. As often in series, it is clearly visible that the averages are much more affected than the medians. Also the standard deviation show a significant decrease when we exclude days with zero tweets. The days that our collection module was not able to receive any tweet from Twitter should not affect our experiments. Therefore we should construct test datasets that do not include any of the days that we did not receive the full amount of tweets.

6.2.2 Quantitative results of collected traffic events

In table 6.3 the quantitative results of our collected traffic events are presented.

Table 6.3: Results of collected traffic events

March	25.195	Avg per month (Mar 17th - Aug 4th)	68.644(73.649)
April	91.136	Avg per day (Mar 17th - Aug 4th)	2.154(2.412)
May	60.706	Median per day	1.920(2.111)
June	71.391	Std dev per day	2.955(2.998)
July	62.775	Min per day	0(126)
August	4.791	Max per day	32.586
	316.649		

As explained in section 2.2 each event contains a *situationId* which is increased for every new unique event: an event that is not directly related to any already existing one. For example: an accident usually starts as an *point event* in the EI dataset and as soon as a traffic jam occurs as a consequence of the accident a *link event* is created with the same *situationId* of the previous point event. A point event is an event that takes place at one location. A link event has a start- and an end point. Later on in this chapter we discuss experiments that make use of bundling events with the same *situationId*. It is therefore interesting for us to know how table 6.3 will look like for *unique events*: grouping events that share the same *situationId*. This table is given in 6.4. The numbers in this table are significantly lower than the ones in table 6.3: on average just above 11.1% of the non-unique variants. Events in EI of the type ‘road construction’ are the main cause of large amounts of repeating events. For example: when a bridge is broken for a long amount of time (a couple of days, weeks) the event ‘bridge is open’ is repeatedly put in all EI datasets that are distributed as long as the bridge is malfunctioning. This example actually happened for a bridge in Waddinxveen.

6.3 Extraction and Analysis

In this section we describe the results of the extraction and analysis step of the methodology. We start by providing an estimation of our bot detection algorithm, continue

Table 6.4: Results of unique traffic events

March	3.013	Avg per month (Mar 17th - Aug 4th)	7.190(7752)
April	8.406	Avg per day (Mar 17th - Aug 4th)	236(254)
May	6.695	Median per day	216(237)
June	8.687	Std dev per day	202(198)
July	6.118	Min per day	0(23)
August	156	Max per day	738
	33.075		

with road and place extraction and end with providing the results of the annotation and sentiment analysis.

6.3.1 Detecting bot accounts

To estimate the performance of our bot detection algorithm we randomly selected 100 ‘bots’ and 100 ‘no-bots’ out of our collection of 180.694 Twitter users. The results are presented in table 6.5.

Table 6.5: Results of bot status for 100 randomly selected ‘bots’ and 100 ‘no-bots’

	actual true	actual false
classified true	89	11
classified false	12	88

In this experimental setup our bot detection algorithm is able to detect *true positives* with a precision of 89% and a recall of 88%. This shows that our algorithm is capable of detecting bot users, but the values must be put in perspective. Since many bot users produce a lot of traffic tweets, each user that is not detected by our algorithm pollutes our data and has a big impact on the system. Therefore any improvement to the bot detection algorithm is welcome. This is considered future work.

6.3.2 Roads and places

The described method in section 4.2 for extracting roads and place names from tweets is quite simplistic. The road extraction rule is only able to extract roads that start with either a *N* or an *A* symbol, followed by one or more numbers. This means the rule is able to detect highways and provincial roads for the Netherlands. When a tweet contains at least one road we say the *hasRoad* property for that tweet is *true*, otherwise *false*. The similar property for a place is called *hasPlace*. The ‘hasRoad rule’ has the following advantages and disadvantages.

Advantages:

1. The rule is (almost) always right: the chance that a term in a tweet like ‘A13’ or ‘N206’ is not referring to a road is negligible.

2. The rule is easy to implement: just match the set of tokens a tweet consist of against the regular expression of the road rule.
3. It does not take many time to determine the *hasRoad* property for the complete Twitter keyword dataset

Disadvantages:

1. The rule can only detect highways and provincial roads. City street names, names of bridges, etc. can not be detected.

To search for place names in tweets we make use of the freely available Geonames database¹. Similar to the road extract rule we use tokenized tweets for matching. The method performs quite well in practise. However, it has some problems that should be taken into account. First of all place names that consist of more than one word, for example ‘Den Haag’, cannot be found by this method. This is because of the tokenization of the tweet text and the exact matching that searches for place name matches for each individual token. Furthermore only precise place names can be matched. When someone makes a spelling mistake or uses a different / secondary name for a place name, the place is simply not recognised.

Another problem is that some places in the Geonames database are ambiguous: besides places in the Netherlands they are common Dutch words. Because these common words occur much more often than that they are used as place names we decided to remove them from the database to increase the performance. The following places were removed:

{*dorp, brug, echt*}

With a resulting amount of 21713 place names still available in the database for matching the potential loss of information because of this removal is minimal.

After the extraction phase 119.298(24%) of the tweets in our keyword database has the property *hasRoad* and 330.653(40%) has the property *hasPlace*.

6.3.3 Annotations and sentiments

In the previous extraction step we extracted places and roads from tweet texts, because we believe that tweets containing either a road name or a place are potential relevant tweets that talk about traffic events. To push the extraction a step further and get a better understanding of what is talked about in our Twitter dataset we used DBPedia Spotlight to annotate our dataset. In total we annotated 106.237 tweets from our keyword dataset. Figure 6.3 shows the quantitative result of the automated annotation task. The large majority of our data, 76.38%, contain no terms recognised by DBPedia. We were a little bit disappointed about this number and expected a bit higher.

A percentage of 76.38 of zero recognised terms automatically means that still over 23%, more than 25.000 tweets of our test set, has at least one recognised term by DBPedia. To get an overview of the most recognised terms we constructed word cloud

¹<http://www.geonames.org/>

6.4 and the related term frequency table 6.6 of the 15 most occurring terms of our dataset.

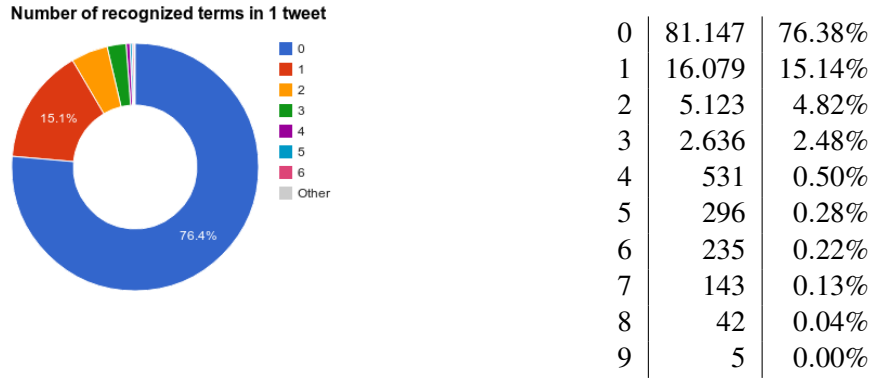


Figure 6.3: Annotation results using DBPedia Spotlight

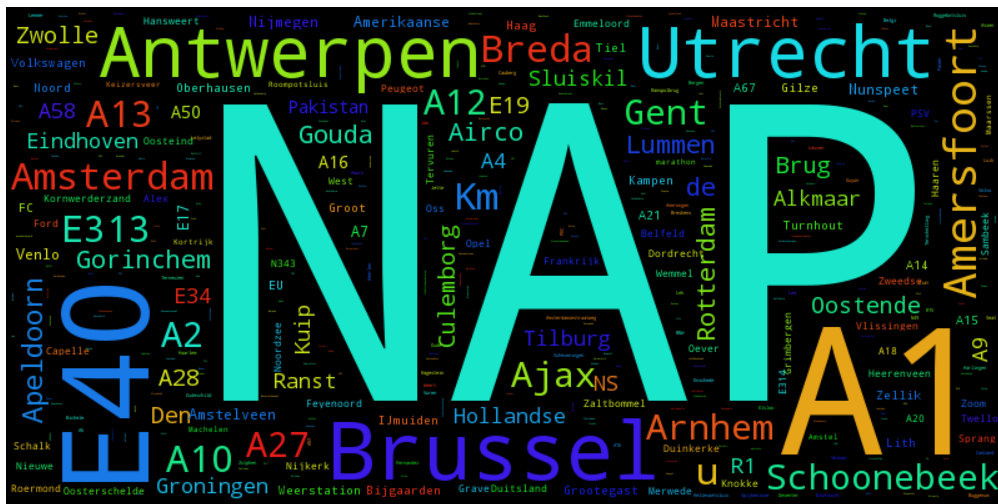


Figure 6.4: Word cloud of annotation results

Besides ‘NAP’ and ‘Km/u’ all terms are roads or place names: the elements we already extracted ourselves in the previous step! ‘NAP’ and ‘Km/u’ occur a lot in our test dataset, because of a large amount of *weather tweets*. This could be interesting tweets for road users. However, since our methodology starts with the detection of traffic events using open traffic data, weather information is not included.

Unfortunately the automated annotation task using DBPedia did not provide us much new insights in our data than we already earned using road and place names extraction.

To get insight in how people react to different traffic disturbances sentiment analysis can be supportive. We

tried to add sentiment scores and labels to a subset of our keyword dataset using the API’s of two Dutch sentiment services: ‘AI Applied’ and ‘Text processing’. AI Applied had a restriction on the amount of API calls per day and was therefore not able to enrich many of our collected tweets with sentiment evaluation. Text processing did not know this limitation, but classified almost all tweets as ‘neutral’, which feels like a ‘save choose’, but do not provide us any information on the reaction of people on traffic events. We concluded that sentiment analysis for the Dutch language, at least without paying, is not on a level that it is useful for research.

6.3.4 Conclusion

After the extraction and analysis phase we were a bit wiser of what our large set of tweets is talking about. Unfortunately, the sentiment analysis and automated annotation tasks did not produce the desired result. Thanks to the road and place extraction we saw many tweets about traffic disturbances; some of them with relevant pictures attached. We also know that it is not easy to find the relevant tweets about traffic disturbances. Tweets are noisy. In the next section we discuss experiments related to the *linkage* phase of our methodology.

6.4 Filtering and linking

One important step in our methodology is the linkage step. For what type of events do we have relevant tweets? And when there are relevant tweets for a particular event, how do we find these? To recall from chapter 4, a tweet can be related to the linked traffic event, related to another traffic event or not related to any traffic event. Furthermore the tweet can be composed by a news- or traffic announcement provider, a bot, or it can be from a ‘real’ user. The abbreviations for all combinations of these properties are:

Table 6.6: 15 most occurring terms in annotation results

#	Occurrences	Term
1	408	NAP
2	331	A1
3	238	E40
4	227	Brussel
5	216	Utrecht
6	205	Amersfoort
7	192	A13
8	188	Amsterdam
9	186	A10
10	183	E313
11	182	Antwerpen
12	178	Km/u Schoonebeek
13	177	Breda
14	147	A12
15	137	A27

related-real, *other-real*, *related-bot* and *other-bot*. Besides these properties we also try to detect the cause type of events based on the information that can be retrieved from the context of the linked tweets.

In this section we describe the results of the different strategies that linked tweets to the traffic events from the EI dataset. It is important to collect quantitative results and perform analysis on them. Related questions to the quantitative aspect of the linking are the following:

- Which part of the verified events have at least one tweet of the *related-real* category? This question tells us which fraction of the events have correct traffic related tweets linked to them.
- How often are pictures attached to traffic related tweets?
- What is the relation between the collected traffic related tweets and the different cause types; e.g. how many (correctly linked) tweets did we find for all different cause types.

Besides quantitative results it is also important to do a qualitative analysis. When there is a photo attached to a tweet it makes quite a difference when that photo reveals useful information about a road construction zone or it is just an ordinary car in some traffic jam. We consider the first case as being *relevant* and the latter as *not-relevant*: it does not provide any useful additional information for the traffic event. However, we must be careful to not classify pictures too easily in the non-relevant category. This qualitative analysis also applies to tweet texts: ‘the A16 is closed in both directions near exit 10 due to an accident’ provides a lot more useful information than ‘Oh no, I’m stuck in a traffic jam again’. Relevant questions related to the qualitative aspect are the following:

- When tweets of *related-real* and *other-real* category provide more information about the event than the descriptions in EI, what kind of additional information do they provide?
- Are the tweets only more relevant when there are relevant pictures attached in most cases or can the tweet texts themselves also be useful?
- Is there a relation between relevant tweets (*related-real* and *different-real*) and the extracted information: annotations, sentiments, roads and places?

6.4.1 Constructing a ground truth

During the extraction and analysis phase we noticed that tweets containing either a place or a road are quite often related to traffic events. However, basic filtering rules that search for these properties do not automatically bring the right tweets to the right events. Therefore we constructed several linkage strategies for this purpose. The first thing we need to do is to construct a solid and correct ground truth. We assume that the point and link locations of traffic events, as well as the cause descriptions and comments, as described in the EI dataset are correct. A precise location together with a short description describing the cause of the traffic disturbance provides us enough

information to verify or falsify linked tweets. Additionally, sometimes the events have general public comments attached to them with information about alternatives routes. This information is also useful in determining the correctness of linked tweets.

6.4.2 Precision and recall

To calculate the precision of our different linkage strategies we need to check for a significant test dataset which tweets are correctly linked to events. As described in the previous section this verification is based on the information that can be retrieved from the EI dataset. To estimate the recall of the different strategies we use one basic filtering rule: time. All tweets that are collected and available in our database with a creation date or ‘tweet time’ within the start and end time of an event can potentially provide traffic information:

$$e_{start} \leq \text{tweet time} \leq e_{end}$$

All tweets composed in this time frame that are related to the traffic event, but not linked by a certain link strategy are reducing the recall. We’ve chosen for this estimation for the recall, because it’s too hard for manual investigation to take tweets into account that are composed outside the time constraint. Furthermore there will always be tweets related to the event that are not captured by our crawling services and therefore not available in our database.

6.4.3 Testing different linkage strategies

The first strategy tested is the *VILD strategy*. Tweets are linked to events when the tweet texts have a non-empty intersection with the VILD terms that describe the start and end point of the event in the EI dataset. The strategy is explained in more detail in section 4.3. Without calculating actual values for the precision and recall yet, first insights of the performance of this strategy, using the test interface from figure 5.3, showed that the precision and recall were not good at all. Many related traffic tweets were not linked and many tweets not related to traffic at all were linked to our collection of traffic events. The main reason for the bad precision was the fact that many *location descriptions* of the VILD terms, like bridge, intersection, highway and exit, are too generic. Many tweets were linked to events because the tweet texts had matching terms with these generic location descriptions. In case of *bridge* quite many tweets were related to traffic events, but very sporadically to the linked event. Intersection is an example of a term that caused many not traffic related tweets to be linked to events. We decided to remove the location description in the calculation of the match percentage, though we still keep track of them.

The next strategy that we tested more thoroughly was the *full-VILD strategy*. Besides the removal of the location description for match determination we also included VILD terms from VILD points that are in between the start and end point of a traffic event. The reason for this was too increase the recall of the VILD strategy without reducing the precision too much. The full-VILD strategy is described in more detail in section 4.3.

The initial idea was to use the experimental linkage setup, as described in section 5.2, to construct the ground truth, determine the precision and recall of the full-VILD strategy and annotate the cause type of the events based on the information found in the tweets all in one huge experiment. The construction of the ground truth in this experimental setup is possible, because every linkage strategy also includes tweets within the time frame of the event that have 0% matching (so related tweets not recognised by a strategy are also tested). This was our basic filtering rule that we decided to use for building our ground truth. When we test all these properties the outcome will become more reliable when we evaluate more and more events.

full-vild+ strategy

The only difference between the full-vild+ and the full-vild strategy is that tweets that are composed 30 minutes before the start time of the event in the EI dataset are also linked:

$$e_{start} - 30 \text{ minutes} \leq \text{tweet time} \leq e_{end}$$

It would be nice if we could discover some tweets related to traffic events that are constructed before the event was included in an EI dataset.

We expect that the overall recall increases when we use the full-vild+ strategy to link tweets, but the precision might suffer from it. Because decreasing all events by 30 minutes might introduce *overlapping events*. Tweets therefore have a higher chance to be linked to the wrong events.

Link rules and restrictions

Each event can contain up to 250 linked tweets. This limit was introduced during the development of the linkage module to make sure that it could link enough events and tweets within a reasonable amount of time. Furthermore this limit enables the construction of the ground truth and the testing of the linkage strategies, which requires lots of manual work.

The properties *containsUrl* and *containsPicture* are automatically annotated based on the meta data of the collected tweets.

Related traffic tweets without relevant information are not categorised as ‘related’. Suppose a tweet is geo-tagged and it’s tweet text is something as ‘I am in a traffic jam’. This tweet only tells us that there might be a traffic disturbance. As we discussed in part one this *detection* can be done much more precise and more easily by using open traffic data.

It often happens that a traffic disturbance at place A is causing another disturbance at place B, because people are avoiding the main traffic disturbance. We decided that tweets that are related to the disturbance at place B are also related to the event at place A when they clearly show a relation between the two events. Since the two traffic disturbances are directly related to each other these kind of tweets tell us that potential redirecting routes to bypass a traffic disturbance are useless.

Traffic jams in the opposite direction of another event, often caused by people staring at the original traffic event, are *not* considered as related.

As explained in 2.2 event records in the EI dataset are accomplished with a *situationId*. As long as the event is valid, according to Rijkswaterstaat and NDW, the *situationId* is maintained in the successive EI datasets. Since the start time of events that share the same *situationId* are the same, the tweets linked to these events are also very similar. To speed up the linkage experiments and prevent double work we only annotate every last event in a range of events that share the same *situationId*. This event has the largest timespan and therefore the most linked tweets.

The following enumeration summarizes the discussed linkage rules:

1. Do not annotate tweets that are composed by bot users. Add bots to a list of ignored users so that future tweets from these users can automatically be excluded.
2. Annotation of the *containsUrl* property is fully automated.
3. Annotation of the *containsPicture* property is automated.
4. Do not annotate tweets that haven't any useful information inside.
5. A tweet related to a traffic disturbance at place B that is related, as a consequence, to a disturbance at place A is categorized as *related* for both the traffic event at place A and place B. Note that this is not always easy to identify.
6. Tweets from a traffic disturbance in the opposite direction of another disturbance (same road) are categorized as *other*.
7. Do not annotate an event when there exists a *more recent* event with the same *situationId*.

Following these rules we hope to construct a ground truth and test the performance of the linkage strategies for enough events within a reasonable amount of time. Because of the linkage restrictions we only have to change the selected values in the experimental setup (see figure 5.3) for only two categories of tweets: *related-real* and *other-real*. In practice this means that many tweets that are not in one of these categories can be 'clicked away' quite fast.

In the next sections we provide the results of several performed linked experiments.

6.4.4 Experiment 1: full-vild linkage strategy

Datasets used Traffic events and keyword dataset

Test datasets description 34 successive *unique events* on 12, 16, 19, 20 and 21 May 2014 and 5339 unique tweets connected to these events using the *full-vild* strategy

Goal of experiment To find out how well, and how often, the full-vild linkage strategy can bring the right traffic tweets to the right events and to see how many traffic tweets reveal information about the cause types of traffic events.

As discussed in the previous section we use the last event in a range of events that share the same *situationId*. To perform this experiment we used the ‘linked events experiments’ tab of our developed system. A screenshot of this tab can be found in figure 5.3. Besides finding out the precision and recall of the *full-vild* strategy we also determine, when possible, the cause type of an event based on the tweet texts. The properties *containsPicture* and *containsUrl* are automatically detected using the meta-data of a tweet.

Results

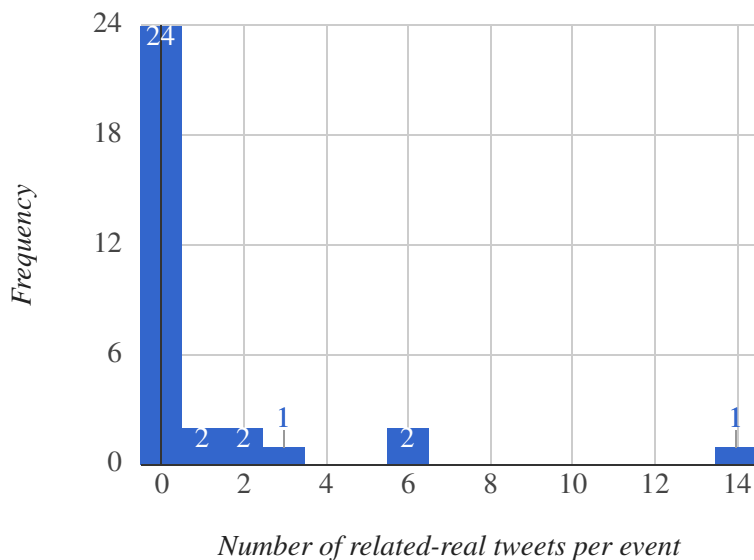


Figure 6.5: Related-real tweets distribution of events

Figure 6.5 shows that we did not find any related-real tweet for 24 out of 32 (75%) tested events. For one event we had 14 unique tweets in the related-real category. This event was a major accident, indicated by the pictures attached to some of the tweets.

For all tweets that we manually classified into the ‘related-real’ or ‘related-other’ category figure 6.6 shows the cause type distribution we extracted from these tweets.

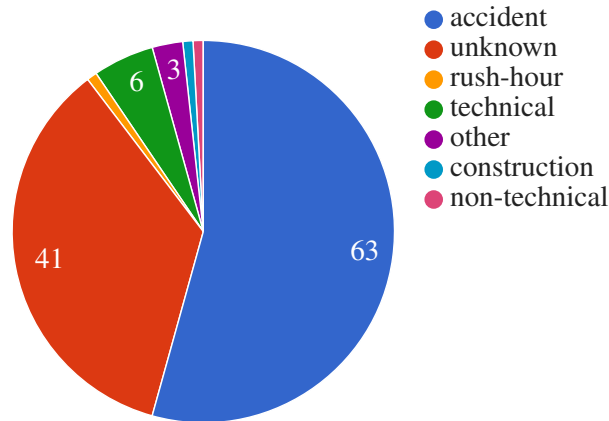


Figure 6.6: Cause type distribution of related-real and other-real tweets

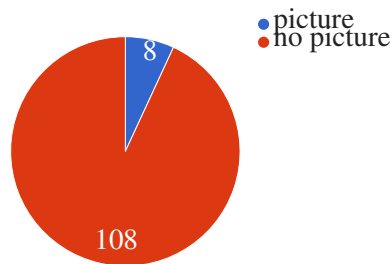


Figure 6.7: *hasPicture* distribution of related-real and other-real tweets

From the 41 ‘unknown tweets’ that did not give us any information about the cause type of a traffic event 0 contained the *hasPicture* property, but 12 tweets contained an url (29%). The information that is provided by these external urls can still provide valuable information about the linked traffic events. The url can for example be a link to an Instagram post. Following urls is outside the scope of our implemented system, but could be an interesting addition.

Figure 6.7 and 6.8 provide the results of the distribution of the *hasPicture* and *hasUrl* property. Pictures can sometimes be very supportive for understanding an event or to add to news articles about serious traffic incidents.

This experiment shows us that tweets can be used to *understand* a traffic event. Tweet texts can be used to extract the cause type of a traffic disturbance. In particu-

lar *accidents* seem to be very suitable to be identified using information from tweets. In the next experiment we investigate to what extent all different cause types, as displayed in methodology section 4.4, could be identified using the information from our Twitter datasets.

In total 51 unique tweets in our test dataset had a positive matching percentage. 49 out of this 51 were actually related to the linked event. In total we found 61 tweets in the related-real category.

This means that the full-vild strategy has precision and recall values of 96% and 80% for this particular dataset. These are high values, but must be put into perspective, because there are only 51 tweets. For comparison: in total we tested 5339 unique tweets. We can conclude that the fraction of traffic tweets composed by real users in our datasets is quite low. One reason for this observation is that many tweets related to traffic are composed by traffic news and media providers (bots). Furthermore tweets can be about different domains than traffic: we received many tweets in the football

and weather domains. This is caused by search keywords that are put into our data crawlers that have ambiguous meanings. ‘Spits’, for example, is a Dutch word for both a rush-hour and a football striker. It is also hard to distinguish ‘real-time traffic tweets’ from generic ones. The following tweet, for example, is a generic tweet about traffic:

@KristinHelene Geen dag gaat voorbij of we staan hopeloos in de file! Gekantelde vrachtwagens, ongelukken met vrachtwagens/auto’s.

In the next experiment we test *randomly selected* unique events happened in the week between March 31 and April 6.

6.4.5 Experiment 2: vild linkage strategies

Datasets used Traffic events, keyword dataset and geo dataset

Test datasets description 100 random unique events equally distributed in the week of March 31 till April 6. All tweets from the keyword dataset that are connected to the selected events using one of the following strategies: *vild*, *full-vild*, *full-vild+*. All tweets from the geo dataset with a location within 500 metres from one of the locations of the event.

Goal of experiment To find out how well, and how often, different linkage strategies can bring the right traffic tweets to the right events. To find out to what extent different cause types of events could be identified using tweets.

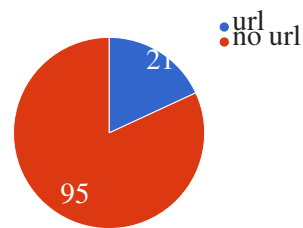


Figure 6.8: *hasUrl* distribution of related-real and other-real tweets

Table 6.7: Number of events and tweets for one week: March 31 - April 6

date	events	unique events	geo	keyword
Mon 31 Mar 2014	1.609	246	89.908	7.806
Tue 1 Apr 2014	3.472	485	92.130	6.443
Wed 2 Apr 2014	2.199	254	90.560	8.217
Thu 3 Apr 2014	3.012	471	85.091	8.441
Fri 4 Apr 2014	3.398	250	86.955	7.352
Sat 5 Apr 2014	1.453	77	90.186	7.143
Sun 6 Apr 2014	397	53	100.248	8.108
	15.540	1.836	635.078	53.510

In table 6.7 is displayed how many data there is available in the week of March 31 till April 6. The number of unique events is significantly lower than the total number of events: $1836/15540 \approx 0.118 \approx 12\%$. Furthermore the number of events in the weekend, especially on Sunday, is a lot lower than during weekdays. This is expected since the EI dataset also include many common rush hour events that happen during weekdays.

We selected 100 events for our test dataset: 14 for weekdays and 15 for Saturday and Sunday. All events were randomly selected, with the only restriction that an event may not enter the test dataset if another event with the same *situationId* is already in.

After the test dataset had been constructed we tested the performance of our linkage strategies using the test interface of figure 5.3. Instead of testing a fixed amount of linked tweets per event, we now test all tweets that are composed 3 minutes before (full-vild+ strategy) or after the start time of the event. This sharp time constraint has been chosen for the following two reasons:

1. We aim to construct a method that can be used in a *near real-time* environment: as soon as a traffic disturbance has been detected we want to collect the related social media content and extract the relevant information for the detected event.
2. It is very time-consuming to check all tweets composed between the start and end time of an event. From experiment 1 we already know that the total amount of related tweets per event is very low.

For geo-related tweets we use a different time constraint: from the start till the end of the event. We test this linkage strategy independent of the vild strategies and are therefore allowed to use a different time span. For each event we first determine the cause type using the traffic information provided by the event description. Afterwards we annotate all tweets which comply with the previous described constraints.

In this experiment we are much more interested in the tweets that are linked to traffic events instead of the ones that are missed. We are much more focused in correctly derive the cause type of a traffic disturbance using information from social media instead of trying to identify as much traffic events as possible. We know that certain traffic events, like rush-hour jams and construction areas, are hard to identify using Twitter. The amount of tweets composed about these cause types are much lower than, for example accidents. Since many of the events that are hard to identify by our

method are *expected events* these events could also be identified using already known (historical) data.

Related questions for this experiment are:

- Which part of the tweets are correctly linked to events?
- To what extend can these tweets provide information about the causes of the traffic disturbances?

Results

Table 6.8: Number of events with at least one ‘related-real’ tweet for different cause types and linkage strategies

	events	combined	vild	full-vild	full-vild+	geo
accident	24	17	13	13	17(6)	2
technical	4	3	3	3	3(0)	1
non-technical	1	0	0	0	0(0)	0
event	2	0	0	0	0(0)	0
construction	33	3	2	2	3(1)	0
rush-hour	29	1	1	1	1(0)	0
breakdown	5	1	1	1	1(0)	0
unknown	2	0	0	0	0(0)	0
	100	25	20	20	25	3

Table 6.7 and figure E.1 (appendix) display for how many events our methodology correctly linked at least one tweet of the *related-real* category. The data is displayed for different cause types and different linkage strategies. The *combined* row and bar show the result when combining all different linkage strategies. The number between parenthesis in the *full-vild+* row show the amount of events for which this strategy found tweets composed before the actual start time of the event stated in the EI dataset. Because our three vild linkage strategies are dependent ($vild \leq full-vild \leq full-vild+$) and the fact that we could not identify additional events using the geo linkage strategy the results for ‘combined’ and ‘full-vild+’ are the same.

For 25 of the 100 tested events the cause type is present in at least one linked tweet with a match percentage > 0 . As explained in chapter 4 *rush-hour*, *event* and *construction* are cause types of the expected kind. Rush-hour events can be more or less predicted using fixed times² and historical data. Construction zones are available as open data³ and also the ‘event’ cause type is available by forehand, although more complicated to collect because you need to combine different data sources. When we exclude the expected cause types: *event*, *construction* and *rush-hour* our tweets contain cause type information for 21 out of 36 (58%) events.

In total we found 57 unique tweets with a match percentage > 0 for the related-real category. 56 tweets were found using a vild linkage strategy. 1 tweet was found using the geo strategy, but could also have been found using a vild matching strategy.

²Usually between 7h and 9h in the morning and 16h30-18h30 in the afternoon/evening.

³The NDW distributes a construction-zone dataset with a similar structure as the EI datasets

The fact that this tweet was only linked using the geo linkage strategy was because it came from our geo database and we only performed the vild linkage strategies to our keyword dataset. 9 tweets out of the 57 (15.8%) contained a picture. In this experiment all tweets matched by the full-vild strategy were also found by the vild-strategy. This means that all matching VILD terms came from the start or end VILD point of the events. The only difference between the two vild strategies are the actual values of the matches. As explained in the methodology chapter a higher matching percentage does not necessarily mean that it is ‘more related’ or whatsoever. We found one tweet containing a cause type that should have been linked to an event, but did not match to any of the VILD terms of the event.

Spread

Another interesting aspect to know is how the related tweets of our experiment are spread with respect to the duration of traffic events. We therefore constructed the graph displayed in E.2 (appendix). In this graph a dark blue line represent the *active* state of a traffic event. An event is bounded by its start and end time according to the information from the EI dataset. In this graph we ignore the dates of traffic events and only take the time into account in order to create a compact graph. For each traffic event we plotted all related tweets that contained cause type information. As can be seen from the graph there are 6 accident events for which we have tweets with a creation time before the start time of the event. The lowest green dot in the graph is another remarkable thing: a correctly linked related-real tweet about a construction event composed before the actual start time of the corresponding event in the EI dataset. The tweet of this green dot clearly stated a traffic jam as a result of a road construction zone:

File #A6 ri Almere #werkzaamheden na Ketelbrug

For 7 traffic events we have tweets with a creation time $< e_{start}$. The time differences between the earliest tweet and e_{start} for these events are: 20, 17, 25, 25, 24, 7 and 22 minutes.

This experiment shows that our linkage strategies are able to correctly link tweets to traffic events. These tweets contain valuable information about the cause type of an event. Especially accidents seem very suitable to identify using social media content. Some tweets are even composed before the event is actually stated in the EI dataset.

The performance of the geo linkage strategy is, as expected, poor. The fact that Twitter does not provide you all tweets for a larger geographical area and the partition of tweets containing geographical coordinates is $\leq 3\%$ are two of the reasons for this performance. In the next experiment we will test a large set of tweets to get a better understanding of the geo linkage strategy performance.

6.4.6 Experiment 3: geo linkage strategy

Datasets used Traffic events and geo dataset

Test datasets description All events (11.666 unique) and geo tweets that are within 50 meters of a traffic event between March 17 and May 1.

Goal of experiment Estimate the performance of the geo linkage strategy and investigate which interesting related tweets can only be found using the geo linkage strategy.

In the previous experiment the geo linkage strategy, with access to two different Twitter datasets, was performing worse than the vild linkage strategies. It was able to find 3 related tweets with cause type information for 3 out of 100 events. This tweets could also have been found using a vild linkage strategy. In this experiment we test a large dataset of events and geo tweets and hope to find some relevant tweets that can not be found using a vild linkage strategy.

In order to achieve a high precision we consider tweets that are within 50 meters of a traffic event. We manually investigate all tweet texts of those tweets.

Results

We found 32 tweets not related to any kind of traffic event, but linked to one by the strategy. Event halls, like the Amsterdam RAI, and other popular places that are close to roads causing non-traffic tweets to be linked to events. To get rid of these kind of tweets you could decrease the event distance of 50 meters to a smaller number, but this automatically also means a decrease in recall, which is already very low.

88 tweets were related to the linked traffic event, but did not provide any additional information. Many of these tweets were check-ins from social media apps like Swarmapp and Foursquare.

19 correctly linked tweets did provide additional information to the detected traffic events.

Without further filtering the precision of the geo linkage strategy with a value of 50 meters close to an event is: $(88 + 19)/(88 + 19 + 32) \approx 0.77 \approx 77\%$. This is a good value, but finding only 19 relevant tweets for 11.666 unique events make this linkage strategy unusable in practice to perform on its own. Even when our 107 related tweets were equally distributed over all tested events we have a coverage of $107/11.666 < 1\%$. We can conclude that in our setup this linkage strategy on its own is not able to find a significant amount of tweets that can be given to the classification module to extract cause types for events.

6.4.7 Conclusion

In this section we described the results of several performed linkage experiments. Experiment 2 showed that our vild linkage strategies were able to correctly link tweets containing cause type information for 25 out of 100 randomly selected traffic events. Experiment 3 showed that the geo linkage strategy on its own, in our setup, is not useful in practice for linking tweets to traffic events. The only way that it can make itself a little bit useful is in a supportive role next to the vild linkage strategies.

In the next section we evaluate the performance of the classification step.

6.5 Classification

6.5.1 Experiment 4: classifications

Datasets used Traffic events and keyword dataset

Test datasets description 116 unique events between March 17 and April 1. Events with a start time e_{start} between 7h-9h and 16h30-18h30 (rush-hour) are excluded.

Goal of experiment To estimate the performance of our cause type classifier.

In this experiment we evaluate the performance of our developed demonstration system regarding the traffic event cause type classification. Each linked tweet is either *related* or *not-related* to the linked traffic event. As discussed in 4.4 all linked tweets using the full-vild+ linkage strategy with a positive match percentage are considered ‘related’. We determine the cause type of detected traffic events using these related tweets. As discussed in section 4.4 we use a dictionary approach to extract the cause types from the filtered set of tweets. To construct this dictionary we investigated five a4 pages with related-real and related-other tweets from experiment 1 and 2 to search for specific frequently occurring traffic cause terms. The system uses all related-real tweets to decide the cause type of each detected traffic event.

From experiment 2 we know that our system is not able to find many tweets regarding rush-hour traffic jams. To speed up our experiment we exclude all traffic events between 7h-9h and 16h30-18h30.

Results

Table 6.9: Actual cause types of traffic events compared with system cause types

		Classified system cause type										
		rush-hour	accident	other accident	event	technical	non-technical	construction	weather	breakdown	other	unknown
Actual cause type	rush-hour (17)		15			2						
	accident (59)	1	34	2					2		20	
	event (1)										1	
	technical (3)					1					2	
	non-technical (1)										1	
	construction (14)		3				1		2		8	
	weather (0)											
	breakdown (18)		6						1		11	
	other (3)		3									
	unknown (0)											

Two accident events were classified by our system as accident, but investigating the tweets that were used to come to this decision revealed that they were actually about a different accident. 34 out of 58 (57%) accident events were correctly identified by our system. In 12 of the 34 detected accident events (35%) our system was able to classify the cause type before the actual start of the event, thanks to the full-vild+ linkage strategy.

When performing this experiment we found the first cause type that we were forced to put in the ‘other’ category: a traffic event caused by people on the opposite side of the road that are staring at the accident that caused the original traffic jam on the other side. We had three different traffic events of this type. Each time our system matched the tweets to the original traffic jam, causing the event to be classified as accident. These type of events are extremely difficult to identify correctly with our developed methodology. Vild terms of two directions of the same road are very similar and therefore our system cannot distinguish between different directions.

The ‘event’-event was caused by a traffic jam nearby the theme park ‘de Efteling’. Our vild linkage strategy did match some tweets of people on their way to the theme park that were stuck in this jam. Our system is not able to identify the *event* cause type in this situation, because terms related to Efteling, including Efteling itself, are simply not presented in the minimalistic cause type dictionary. An enhancement of this cause type dictionary can significantly increase the performance of the cause type classifier.

6.5.2 Conclusion

Our system was not able to identify the traffic cause type for 37% of our tested events. It classified 33% correctly and made the wrong decision for 30% of the tested events. These numbers are far from great, but expected when using a very simple cause type classifier based on a minimalistic self-constructed dictionary. The most important reasons for incorrect cause type classification are the following:

1. Tweets linked to detected events are about *other* traffic events. They are classified as ‘related’ by our linkage strategy, most of the time, because of a matching `secnd_name` vild property.
2. Tweets that are considered as related are of non real-time nature. It are tweets of traffic events that will happen in the future or happened in the past. It is difficult to filter these kind of tweets.
3. The system always tries to decide. Even if there exist only one related tweet that matches with a particular cause type, it uses this cause type as its final decision. In cases where our system decided the cause type based on more than 10 tweets, it was almost always right. So, in order to make safer decisions a threshold could be built into the system: only decide on a sufficient amount of related tweets. As a drawback the amount of ‘unknown’ cause type will also increase. Another possibility would be to provide a confidence level for the decided cause type(s).

In chapter 8 possible improvements for the cause type classifier are presented.

Part III

Conclusions and future work

Chapter 7

Conclusions

In this chapter we answer our research question: ‘To what extent can social media support traffic information during uncommon disturbances on the road?’. We discuss our project contributions and reflect on our developed methodology.

7.1 Conclusions

In Part I we focused on the detection of traffic events using Twitter as social media platform. We choose several areas where we knew particular events would take place and roads should get crowded. In order to estimate a base level to be able to detect traffic disturbances we collected tweets containing geographical coordinates within a short range to our chosen areas. We monitored the areas for several days and searched for irregular tweet behaviour. When an abnormal amount of tweets is composed within a certain amount of time you know that something is going on. The power of this kind of event detection[2][3] is that it is relatively easy to set up and can detect all different kinds of events.

We were able to detect the events that took place at our monitored areas. However, although the amount of tweets composed before, during and after the events were significantly above the normal level of activity, they did not provide us enough detailed information about the traffic congestion around the areas. We needed more data input than we received in order to compete and compare to open traffic data. We expect similar results when using other social media platforms and therefore conclude that social media is not able to operate as an appropriate alternative to open traffic data.

In Part II we presented a methodology that extracts traffic information from social media content and links this information to detected traffic disturbances. In our developed system we use predefined datasets from the NDW that contain all kinds of traffic events. We assume that most of these traffic events could also have been detected using raw open traffic data[8]. Our system is able to extract cause type information and provide images using social media content for a significant amount of *unexpected* traffic events. These cause types include technical road issues, accidents and car and truck breakdowns. Non real-time traffic tweets are difficult for our system to deal with. They can easily be incorrectly linked to traffic events, because they are hard to distinguish from real-time tweets.

7.2 Discussion/Reflection

Studies[7][1] claim to be able to detect traffic events using a keyword based approach. In countries where the Intelligent transportation systems (ITS) are less developed than the Netherlands we believe this could be a low-cost solution for traffic event detection. However, this methodology is expected to be far less detailed and reliable than precise real-time collected traffic data.

Our Twitter datasets and developed system show that it is hard to find plenty of interesting traffic material that is not composed by ‘bot users’ for all kind of traffic events. But, as soon as our linkage strategy links a tweet to a detected traffic event it has a high chance of being related to that event. When such a related tweet contains a nice picture or a possible cause type it definitely is a valuable addition to a detected traffic event. Because of the low availability of traffic material our cause type classifier has to make decisions based on just a few tweets. This means that it can be easily misled when a wrong, not-related tweet is linked to a traffic event.

Information about *expected* traffic events, like rush-hour jams and construction zones, is far less presented in our system. People obviously tweet more easily when they see an impressive accident or are stuck in a traffic jam in the middle of the night than when they do when they are in a common traffic disturbance. Most of the expected traffic events can also be identified using common sense, historical data and open traffic data¹. The lack of this information in our system should therefore not be considered as a major shortcoming.

It should be possible to apply our work in other countries under the following conditions:

- The ITS must be sufficiently developed. Since our method uses a set of pre-defined events, a technology that is able to detect traffic disturbances must be implemented in the country of interest.
- Mobile internet must be sufficiently developed. This means it is stable and has an adequate coverage in order to extract cause types for traffic disturbances near real time.
- Twitter must be available and be at ‘a certain level of popularity’.

Our vild linkage strategies make use of VILD tables: tables describing traffic measuring points by providing names and road numbers. When similar tables exist in other countries this definitely can support linking the right tweets to the right events. However, linking should also be possible based on a set of places and roads.

Furthermore our work could also be applied to other social media platforms, as long as they show some similarities with Twitter like short messages, a keyword search possibility/API and a real-time nature.

¹The NDW distributes construction-zone datasets that are available long before the events actually take place.

7.3 Contributions

In the scientific field there exist keyword based approaches [7][1] that use Twitter to detect and classify traffic events. These researches use people as *social sensors* in order to detect disturbances. Our developed methodology differs from these approaches in a way that it uses more precise and reliable open traffic data to do the detection part and use social media content to enrich detected events with cause types and pictures.

A demonstration system has been developed that is able to link tweets to traffic events using different linkage strategies. From the related linked tweets it tries to derive the cause type of a traffic event using a dictionary approach.

In the practical field the automatically extracted traffic information can be useful for news sites. Especially the cases when the system finds relevant tweets that contain interesting pictures can be of great value in this application. Furthermore our system can be useful for road users. When people are unexpectedly stuck in traffic they are often curious about *what* is happening. That is the question that we aimed to solve in our developed system. The advantage of reasoning from detected traffic events using reliable measuring equipment is that we cannot provide information for disturbances that do not exist. At time of writing there exist many mobile traffic applications that display user mentioned traffic disturbances on a map. I have experienced in practice that these applications display many incorrect events. It is confusing and misleading when a Google map shows a green road together with a stop icon that says 'road closure', without any notion of time or details about the closure.

Chapter 8

Future Work

In this chapter future work to improve the system is presented.

8.1 Bot detection algorithm

Our simple bot detection algorithm is able to classify Twitter users as ‘bot users’ with a high precision. Unfortunately there also exists a relatively high amount of *false negatives*: bot users that are not detected by the algorithm. We had to manually set the bot status for these users to make sure that they did not disturb our experiments and unintentionally influence our system. Improving the bot detection algorithm means an improve of the overall system, because bot users pollute our data in all phases of the methodology. It is likely that in the future new traffic bot users will be created. An improvement of the detection algorithm can reduce or even eliminate the amount of manual work that needs to be done to identify them.

One potential improvement of the algorithm is to investigate the distribution of tweets. When this distribution show suspicious patterns this can indicate the presence of a bot. They are often silent for a certain amount of time and than spread several tweets within a very short amount of time.

8.2 Collection

To collect more potential interesting tweets more keywords could be added to the system. We now use 12. The Twitter streaming API allow you to add up to 400 keywords. For example roads that are known to cause lots of traffic disturbances could be added. Furthermore collecting tweets using multiple Twitter accounts might help to work around the Twitter API limitations.

As discussed in chapter 7 our methodology should not be restricted to Twitter as only working social media platform. Therefore adding different social media platforms, such as Instagram, might add value to our system.

8.3 Linkage

To increase the recall of related tweets we can enrich the set of VILD terms. Our method now uses a small extension to the original set of VILD terms: terms that

contain ‘van’, like ‘van Brienenoordbrug’, are also put in the list of match candidates without the preposition ‘van’. There probably exists more terms that are often referred to by different synonyms or acronyms. To match tweets by these different formats they must be added to the list of match candidates.

The positive match percentage that is used as heuristic to determine related tweets works quite well in practice. However, it can not distinguish non real-time tweets from real-time tweets. Non real-time tweets can easily ‘fool’ the heuristic and potentially causes the system to extract the wrong cause type for a detected event. To develop a more stable classifier a filter must be created that is able to eliminate non real-time tweets. This is not easy.

The *secnd_name* property of the VILD points, that at this moment can be used for matching, is not always leading to related tweets. We had cases where this property linked traffic tweets to wrong traffic events, because the *secnd_name* property refers to intersecting roads. It is wise to investigate if this property actually increases the recall of related tweets. Perhaps it is better to remove this property, just as we did with the *loc_des* (location description) property. This property had a very bad impact on the precision of our linkage strategies.

8.4 Classification

The developed cause type classification step is primitive. It determines cause types for traffic events by scanning the related tweets of the linkage step for terms available in a self-constructed dictionary. The rule could certainly be improved. Extending the dictionary might help, but this can also lead to wrong decisions. Some terms only indicate a particular cause type when they coexist with other terms. This *dependency feature*, together with other features, might be used in a supervised machine learning environment to extract cause types from a set of tweets.

Besides traffic cause types, other type of information could also be extracted from the retrieved data. Sometimes tweets contain very detailed descriptions of accidents that happened. This information can be useful for other road users and (local) news sites. Some tweets also contain information about road and lane closures. This could also be interesting information for our system.

Bibliography

- [1] N. Wanichayapong;W. Pruthipunyaskul;W. Pattara-Atikom;P. Chaovalit. Social-based traffic information extraction and classification. In *ITS Telecommunications (ITST), 2011 11th International Conference*, pages 107–112, 2011.
- [2] Tatsuya Fujisaka, Ryong Lee, and Kazutoshi Sumiya. Discovery of user behavior patterns from geo-tagged micro-blogs. In *Proceedings of the 4th International Conference on Ubiquitous Information Management and Communication, ICUIMC '10*, pages 36:1–36:10, New York, NY, USA, 2010. ACM.
- [3] Ryong Lee and Kazutoshi Sumiya. Measuring geographical regularities of crowd behaviors for Twitter-based geo-social event detection. In *Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Location Based Social Networks, LBSN '10*, pages 1–10, New York, NY, USA, November 2010. ACM.
- [4] Fred Morstatter, Jürgen Pfeffer, Huan Liu, and Kathleen M. Carley. Is the sample good enough? comparing data from twitter’s streaming api with twitter’s firehose. *CoRR*, abs/1306.5204, 2013.
- [5] Alexandra Olteanu, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. CrisisLex: A Lexicon for Collecting and Filtering Microblogged Communications in Crises. In *Proc. of the 8th International Conference on Weblogs and Social Media*, pages 376–385, June 2014.
- [6] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: Real-time event detection by social sensors. In *Proceedings of the 19th International Conference on World Wide Web, WWW '10*, pages 851–860, New York, NY, USA, 2010. ACM.
- [7] Yutaka; Yanagihara Tadashi; Chandrasiri Naiwala P.; Nawa Kazunari Sakaki, Takeshi; Matsuo. Real-time event extraction for driving information from social sensors. In *Proceedings of the 2012 IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems, 2012*.
- [8] Jasper Vries. Provinciale toepassingen voor wegverkeersgegevens: Gebeurtenis-detectie op basis van ndw verkeersdata. Master’s thesis, Delft University of Technology, the Netherlands, 2012.

- [9] Lance Whitney. Google maps adds waze traffic reports, August 2013.

Appendix A

NDW Dataset examples

A.1 MST

Code A.1: An MeasurementSiteTable (MST) example dataset containing measuring locations. The complete xml file consists of over 3.5 million lines

```
1 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"><soap:Body><
  d2LogicalModel xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.
  org/2001/XMLSchema-instance" xmlns="http://datex2.eu/schema/2/2_0" modelBaseVersion="2"
  >
2 <exchange>
3   <supplierIdentification>
4     <country>nl</country>
5     <nationalIdentifier>NDW-CNS</nationalIdentifier>
6   </supplierIdentification>
7 </exchange>
8 <payloadPublication xsi:type="MeasurementSiteTablePublication" lang="nl">
9   <publicationTime>2014-10-31T12:00:00Z</publicationTime>
10  <publicationCreator>
11    <country>nl</country>
12    <nationalIdentifier>NDW-CNS</nationalIdentifier>
13  </publicationCreator>
14  <headerInformation>
15    <confidentiality>noRestriction</confidentiality>
16    <informationStatus>real</informationStatus>
17  </headerInformation>
18  <measurementSiteTable id="NDW01_MT" version="657">
19    <measurementSiteRecord id="PZH01_MST_0007_00" version="1">
20      <measurementSiteRecordVersionTime>2014-02-16T22:00:00Z</
        measurementSiteRecordVersionTime>
21      <computationMethod>arithmeticAverageOfSamplesInATimePeriod</computationMethod>
22      <measurementEquipmentTypeUsed>
23        <values>
24          <value lang="nl">lus</value>
25        </values>
26      </measurementEquipmentTypeUsed>
27      <measurementSiteName>
28        <values>
29          <value lang="nl">N206 km 9.742</value>
30        </values>
31      </measurementSiteName>
32      <measurementSiteNumberOfLanes>2</measurementSiteNumberOfLanes>
```

```

33 <measurementSpecificCharacteristics index="1">
34 <measurementSpecificCharacteristics>
35 <accuracy>95.0</accuracy>
36 <period>60.0</period>
37 <specificLane>lane1</specificLane>
38 <specificMeasurementValueType>trafficFlow</specificMeasurementValueType>
39 <specificVehicleCharacteristics>
40 <lengthCharacteristic>
41 <comparisonOperator>lessThan</comparisonOperator>
42 <vehicleLength>5.60</vehicleLength>
43 </lengthCharacteristic>
44 </specificVehicleCharacteristics>
45 </measurementSpecificCharacteristics>
46 </measurementSpecificCharacteristics>
47 <measurementSpecificCharacteristics index="2">
48 <measurementSpecificCharacteristics>
49 <accuracy>95.0</accuracy>
50 <period>60.0</period>
51 <specificLane>lane1</specificLane>
52 <specificMeasurementValueType>trafficFlow</specificMeasurementValueType>
53 <specificVehicleCharacteristics>
54 <lengthCharacteristic>
55 <comparisonOperator>greaterThanOrEqualTo</comparisonOperator>
56 <vehicleLength>5.60</vehicleLength>
57 </lengthCharacteristic>
58 <lengthCharacteristic>
59 <comparisonOperator>lessThanOrEqualTo</comparisonOperator>
60 <vehicleLength>12.20</vehicleLength>
61 </lengthCharacteristic>
62 </specificVehicleCharacteristics>
63 </measurementSpecificCharacteristics>
64 </measurementSpecificCharacteristics>
65 <measurementSiteLocation xsi:type="Point">
66 <locationForDisplay>
67 <latitude>52.13606</latitude>
68 <longitude>4.48862</longitude>
69 </locationForDisplay>
70 <supplementaryPositionalDescription>
71 <affectedCarriagewayAndLanes>
72 <carriageway>mainCarriageway</carriageway>
73 </affectedCarriagewayAndLanes>
74 </supplementaryPositionalDescription>
75 <alertCPoint xsi:type="AlertCMethod4Point">
76 <alertCLocationCountryCode>8</alertCLocationCountryCode>
77 <alertCLocationTableNumber>5.7</alertCLocationTableNumber>
78 <alertCLocationTableVersion>A</alertCLocationTableVersion>
79 <alertCDirection>
80 <alertCDirectionCoded>positive</alertCDirectionCoded>
81 </alertCDirection>
82 <alertCMethod4PrimaryPointLocation>
83 <alertCLocation>
84 <specificLocation>10824</specificLocation>
85 </alertCLocation>
86 <offsetDistance>
87 <offsetDistance>842</offsetDistance>
88 </offsetDistance>
89 </alertCMethod4PrimaryPointLocation>

```

```

90     </alertCPoint>
91     </measurementSiteLocation>
92     </measurementSiteRecord>
93     </measurementSiteTable>
94     </payloadPublication>
95 </d2LogicalModel></soap:Body></soap:Envelope>

```

A.2 TFTS

Code A.2: An Traffic Speed Traffic Flow (TFTS) dataset example containint real time traffic speeds and flow values

```

1 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"><soap:Body><
  d2LogicalModel xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.
  org/2001/XMLSchema-instance" xmlns="http://datex2.eu/schema/2/2_0" modelBaseVersion="2"
  xsi:schemaLocation="http://datex2.eu/schema/2/2_0 http://www.ndw.nu/DATEXII/
  DATEXII Schema_2_2_1.xsd">
2 <exchange xmlns="http://datex2.eu/schema/2/2_0">
3 <supplierIdentification>
4 <country>nl</country>
5 <nationalIdentifier>NDW-CNS</nationalIdentifier>
6 </supplierIdentification>
7 </exchange>
8 <payloadPublication xmlns="http://datex2.eu/schema/2/2_0" xsi:type="MeasuredDataPublication"
  lang="nl">
9 <publicationTime>2014-10-31T17:48:55.000Z</publicationTime>
10 <publicationCreator>
11 <country>nl</country>
12 <nationalIdentifier>NDW-CNS</nationalIdentifier>
13 </publicationCreator>
14 <measurementSiteTableReference id="NDW01_MT" version="657" targetClass="
  MeasurementSiteTable"/>
15 <headerInformation>
16 <confidentiality>noRestriction</confidentiality>
17 <informationStatus>real</informationStatus>
18 </headerInformation>
19 <siteMeasurements>
20 <measurementSiteReference id="GEO01_Z_RWSTI650" version="1" targetClass="
  MeasurementSiteRecord"/>
21 <measurementTimeDefault>2014-10-31T17:47:00Z</measurementTimeDefault>
22 <measuredValue index="1" xsi:type="_SiteMeasurementsIndexMeasuredValue">
23 <measuredValue xsi:type="MeasuredValue">
24 <basicData xsi:type="TrafficFlow">
25 <vehicleFlow numberOfInputValuesUsed="0">
26 <vehicleFlowRate>0</vehicleFlowRate>
27 </vehicleFlow>
28 </basicData>
29 </measuredValue>
30 </measuredValue>
31 <measuredValue index="2" xsi:type="_SiteMeasurementsIndexMeasuredValue">
32 <measuredValue xsi:type="MeasuredValue">
33 <basicData xsi:type="TrafficFlow">
34 <vehicleFlow numberOfInputValuesUsed="6">
35 <vehicleFlowRate>360</vehicleFlowRate>
36 </vehicleFlow>
37 </basicData>

```

```

38     </measuredValue>
39 </measuredValue>
40 <measuredValue index="3" xsi:type="_SiteMeasurementsIndexMeasuredValue">
41   <measuredValue xsi:type="MeasuredValue">
42     <basicData xsi:type="TrafficFlow">
43       <vehicleFlow numberOfInputValuesUsed="1">
44         <vehicleFlowRate>60</vehicleFlowRate>
45       </vehicleFlow>
46     </basicData>
47   </measuredValue>
48 </measuredValue>
49 <measuredValue index="4" xsi:type="_SiteMeasurementsIndexMeasuredValue">
50   <measuredValue xsi:type="MeasuredValue">
51     <basicData xsi:type="TrafficFlow">
52       <vehicleFlow numberOfInputValuesUsed="0">
53         <vehicleFlowRate>0</vehicleFlowRate>
54       </vehicleFlow>
55     </basicData>
56   </measuredValue>
57 </measuredValue>
58 <measuredValue index="5" xsi:type="_SiteMeasurementsIndexMeasuredValue">
59   <measuredValue xsi:type="MeasuredValue">
60     <basicData xsi:type="TrafficFlow">
61       <vehicleFlow numberOfInputValuesUsed="0">
62         <vehicleFlowRate>0</vehicleFlowRate>
63       </vehicleFlow>
64     </basicData>
65   </measuredValue>
66 </measuredValue>
67 <measuredValue index="6" xsi:type="_SiteMeasurementsIndexMeasuredValue">
68   <measuredValue xsi:type="MeasuredValue">
69     <basicData xsi:type="TrafficFlow">
70       <vehicleFlow numberOfInputValuesUsed="7">
71         <vehicleFlowRate>420</vehicleFlowRate>
72       </vehicleFlow>
73     </basicData>
74   </measuredValue>
75 </measuredValue>
76 <measuredValue index="7" xsi:type="_SiteMeasurementsIndexMeasuredValue">
77   <measuredValue xsi:type="MeasuredValue">
78     <basicData xsi:type="TrafficSpeed">
79       <averageVehicleSpeed numberOfIncompleteInputs="0" numberOfInputValuesUsed="0"
80         supplierCalculatedDataQuality="100.0">
81         <speed>0</speed>
82       </averageVehicleSpeed>
83     </basicData>
84   </measuredValue>
85 </siteMeasurements>
86 </payloadPublication>
87 </d2LogicalModel></soap:Body></soap:Envelope>

```

A.3 TT

Code A.3: An traveltime (TT) dataset example containing traveltimes between measuring points

```

1 <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"><soap:Body><
  d2LogicalModel xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.
  org/2001/XMLSchema-instance" xmlns="http://datex2.eu/schema/2/2_0" modelBaseVersion="2"
  xsi:schemaLocation="http://datex2.eu/schema/2/2_0 http://www.ndw.nu/DATEXII/
  DATEXII Schema_2_2_1.xsd">
2 <exchange xmlns="http://datex2.eu/schema/2/2_0">
3   <supplierIdentification>
4     <country>nl</country>
5     <nationalIdentifier>NDW-CNS</nationalIdentifier>
6   </supplierIdentification>
7 </exchange>
8 <payloadPublication xmlns="http://datex2.eu/schema/2/2_0" xsi:type="MeasuredDataPublication"
  lang="nl">
9   <publicationTime>2014-10-31T18:00:55.000Z</publicationTime>
10  <publicationCreator>
11    <country>nl</country>
12    <nationalIdentifier>NDW-CNS</nationalIdentifier>
13  </publicationCreator>
14  <measurementSiteTableReference id="NDW01_MT" version="657" targetClass="
  MeasurementSiteTable"/>
15  <headerInformation>
16    <confidentiality>noRestriction</confidentiality>
17    <informationStatus>real</informationStatus>
18  </headerInformation>
19  <siteMeasurements>
20    <measurementSiteReference id="PNB01_BEMOB_68" version="8" targetClass="
  MeasurementSiteRecord"/>
21    <measurementTimeDefault>2014-10-31T17:59:00Z</measurementTimeDefault>
22    <measuredValue index="1" xsi:type="_SiteMeasurementsIndexMeasuredValue">
23      <measuredValue xsi:type="MeasuredValue">
24        <basicData xsi:type="TravelTimeData">
25          <measurementOrCalculationTime>2014-10-31T18:00:49Z</
  measurementOrCalculationTime>
26          <travelTimeType>best</travelTimeType>
27          <travelTime>
28            <duration>168</duration>
29          </travelTime>
30        </basicData>
31      </measuredValue>
32    </measuredValue>
33  </siteMeasurements><siteMeasurements>
34  <measurementSiteReference id="GEO03_D4T-POV_T_HAVER1_ID_8578" version="1"
  targetClass="MeasurementSiteRecord"/>
35  <measurementTimeDefault>2014-10-31T17:59:00Z</measurementTimeDefault>
36  <measuredValue index="1">
37    <measuredValue>
38      <basicData xsi:type="TravelTimeData">
39        <travelTimeType>best</travelTimeType>
40        <travelTime numberOfInputValuesUsed="22" standardDeviation="16">
41          <duration>137</duration>
42        </travelTime>
43      </basicData>
44    </measuredValue>

```

```

45     </measuredValue>
46 </siteMeasurements>
47 <siteMeasurements>
48     <measurementSiteReference id="PNB01_BEMOB_70" version="8" targetClass="
MeasurementSiteRecord"/>
49     <measurementTimeDefault>2014-10-31T17:59:00Z</measurementTimeDefault>
50     <measuredValue index="1" xsi:type="_SiteMeasurementsIndexMeasuredValue">
51         <measuredValue xsi:type="MeasuredValue">
52             <basicData xsi:type="TravelTimeData">
53                 <measurementOrCalculationTime>2014-10-31T18:00:49Z</
measurementOrCalculationTime>
54                 <travelTimeType>best</travelTimeType>
55                 <travelTime>
56                     <duration>130</duration>
57                 </travelTime>
58             </basicData>
59         </measuredValue>
60     </measuredValue>
61 </siteMeasurements>
62 <siteMeasurements>
63     <measurementSiteReference id="GEO03_D4T-RWS_T_0317_ID_324" version="3"
targetClass="MeasurementSiteRecord"/>
64     <measurementTimeDefault>2014-10-31T17:59:00Z</measurementTimeDefault>
65     <measuredValue index="1">
66         <measuredValue>
67             <basicData xsi:type="TravelTimeData">
68                 <travelTimeType>best</travelTimeType>
69                 <travelTime numberOfInputValuesUsed="33" standardDeviation="26">
70                     <duration>40</duration>
71                 </travelTime>
72             </basicData>
73         </measuredValue>
74     </measuredValue>
75 </siteMeasurements>
76     <measurementSiteReference id="RWS01_MONIBAS_0021hrr0308ra0" version="1"
targetClass="MeasurementSiteRecord"/>
77     <measurementTimeDefault>2014-10-31T17:59:00Z</measurementTimeDefault>
78     <measuredValue index="1" xsi:type="_SiteMeasurementsIndexMeasuredValue">
79         <measuredValue xsi:type="MeasuredValue">
80             <basicData xsi:type="TravelTimeData">
81                 <travelTimeType>best</travelTimeType>
82                 <travelTime numberOfIncompleteInputs="0">
83                     <duration>19</duration>
84                 </travelTime>
85             </basicData>
86         </measuredValue>
87     </measuredValue>
88 </siteMeasurements>
89 </payloadPublication>
90 </d2LogicalModel></soap:Body></soap:Envelope>

```


A.4 VILD

Code A.4: A snapshot of a VILD table

1	7616;P1.3;Afrit;A16;;Zwijndrecht ;;0;22;335;326;323;328;;-1;1;1;1;1;;2564;3065;0;7617;7615;0;1;1;0;;;0;367;16;427
2	7617;P1.3;Afrit;A16;;Hendrik-Ido-Ambacht ;;0;23;314;297;294;311;;-1;1;1;1;1;;2501;3065;0;21695;7616;0;1;1;0;;;0;367;16;427
3	7618;P1.14;Verbindingsweg;A16;;A16 vanuit Breda;A15 richting Gorinchem;7619;;280;275;-1;-1;s ;-1;0;1;0;0;;2535;3065;0;7621;21696;0;1;0;0;;;0;367;16;427
4	7619;P1.1;Knooppunt;A16;;Ridderkerk-Zuid;A15 ;7619;;287;267;267;287;;-1;1;1;1;1;;2535;3065;7453;7620;7621;0;1;1;0;;;0;367;16;427
5	7620;P1.14;Verbindingsweg;A16;;A16 vanuit Rotterdam;A15 richting Gorinchem ;7619;;-1;-1;249;259;t;-1;0;0;0;1;;2535;3065;0;7622;7619;0;0;1;0;;;0;367;16;427
6	7621;P1.14;Verbindingsweg;A16;;A16 vanuit Breda;A15 richting Europoort;7619;;279;272;-1;-1;y ;-1;0;1;0;0;;2535;3065;0;7619;7618;0;1;0;0;;;0;367;16;427
7	7622;P1.1;Knooppunt;A16;Ring Rotterdam;Ridderkerk-Noord;A15/A38 ;7622;;267;246;246;267;;-1;1;1;1;1;;2535;3065;7450;7623;7620;0;1;1;0;;;0;367;16;427
8	7623;P1.14;Verbindingsweg;A16;Ring Rotterdam;A16 vanuit Rotterdam;A15 richting Europoort ;7622;;-1;-1;245;254;p;-1;0;0;0;1;;2535;3065;0;7624;7622;0;0;1;0;;;0;367;16;427
9	7624;P3.47;Viaduct;A16;Ring Rotterdam;Groeninx van Zoelen ;;0;235;235;235;235;;-1;0;0;0;0;;2536;3065;0;7625;7623;0;1;1;0;;;0;367;16;427
10	7625;P1.3;Afrit;A16;Ring Rotterdam;Feijenoord ;;0;24;230;215;216;225;;-1;1;1;1;1;;2536;3065;26224;7626;7624;0;1;1;0;Rotterdam ;;0;367;16;427
11	7626;P3.47;Viaduct;A16;Ring Rotterdam;John F Kennedyweg ;;0;215;215;215;215;;-1;0;0;0;0;;2536;3065;0;7627;7625;0;1;1;0;;;0;366;16;425
12	7627;P3.2;Brug;A16;Ring Rotterdam;Van Brienenoordbrug;Nieuwe Maas ;0;;211;207;207;211;;-1;0;0;0;0;;2536;3065;0;7628;7626;0;1;1;0;;;2;366;16;425
13	7628;P1.3;Afrit;A16;Ring Rotterdam;Rotterdam-Centrum;N210 ;0;25;200;191;190;201;;-1;1;1;1;1;;2536;3065;26230;7629;7627;0;1;1;0;;;0;366;16;425
14	7629;P1.3;Afrit;A16;Ring Rotterdam;Kralingen ;;0;26;187;187;184;184;;-1;1;1;1;1;;2536;3065;26300;7630;7628;0;1;1;0;Rotterdam ;;0;366;16;425
15	7630;P1.3;Afrit;A16;Ring Rotterdam;Prins Alexander ;;0;27;161;161;160;160;;-1;0;1;1;0;;2536;3065;26245;7631;7629;0;1;1;0;Rotterdam ;;0;366;16;425
16	7631;P1.14;Verbindingsweg;A16;Ring Rotterdam;A16 vanuit Breda;A20 richting Gouda ;7633;;160;150;-1;-1;f;-1;0;1;0;0;;2536;3065;0;7632;7630;0;1;0;0;;;0;366;16;425
17	7632;P1.14;Verbindingsweg;A16;Ring Rotterdam;A16 vanuit Breda;A20 richting Hoek van Holland ;7633;;156;153;-1;-1;v;-1;0;1;0;0;;2536;3065;0;7633;7631;0;1;0;0;;;0;366;16;425
18	7633;P1.1;Knooppunt;A16;Ring Rotterdam;Terbregseplein;A20 ;7633;;166;140;156;156;;-1;1;1;1;1;;2536;3065;7682;0;7632;0;1;1;0;;;0;366;16;425

Code A.5: A snapshot of a converted VILD row into JSON

```
1 {
2   "_id": "7620_VILD_5.7.A",
3   "_rev": "1-f529f91408687bc6850329bc627ca4f9",
4   "neg_out": 1,
5   "hstart_neg": 249,
6   "aw_ref": 427,
7   "hend_neg": 259,
8   "pos_in": 0,
9   "top_sign": "",
10  "type_code": 0,
11  "pres_neg": 1,
12  "secnd_name": "A15 richting Gorinchem",
13  "first_name": "A16 vanuit Rotterdam",
14  "lin_ref": 3065,
15  "hend_pos": -1,
16  "mw_ref": 367,
17  "version": {
18    "date": "1-04-2014",
19    "number": "5.7.A",
20    "datems": 1396303200000
21  },
22  "neg_in": 0,
23  "rw_nr": 16,
24  "loc_nr": 7620,
25  "type": "vild",
26  "pos_off": 7622,
27  "roadname": "",
28  "far_away": 0,
29  "area_ref": 2535,
30  "roadnumber": "A16",
31  "inter_ref": 0,
32  "junct_ref": 7619,
33  "pos_out": 0,
34  "neg_off": 7619,
35  "city_distr": "",
36  "hecto_char": "t",
37  "loc_des": "Verbindingsweg",
38  "hecto_dir": -1,
39  "pres_pos": 0,
40  "loc_type": "P1.14",
41  "urban_code": 0,
42  "exit_nr": "",
43  "hstart_pos": -1,
44  "dir": ""
45 }
```

A.5 Eventinfo

Code A.6: An eventinfo (EI) example dataset containing one event/situation

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
3   <SOAP-ENV:Header />
4   <SOAP-ENV:Body>
5     <d2LogicalModel modelBaseVersion="2.0" xmlns="http://datex2.eu/schema/2_0/2_0">
6       <exchange>
7         <supplierIdentification>
8           <country>nl</country>
9           <nationalIdentifier>NDWNL</nationalIdentifier>
10        </supplierIdentification>
11        <subscription>
12          <operatingMode>operatingMode1</operatingMode>
13          <subscriptionStartTime>2014-01-15T20:46:56Z</subscriptionStartTime>
14          <subscriptionState>active</subscriptionState>
15          <updateMethod>snapshot</updateMethod>
16          <target>
17            <address></address>
18            <protocol>HTTP</protocol>
19          </target>
20        </subscription>
21        <filterReference>
22          <keyFilterReference>CBS filter</keyFilterReference>
23        </filterReference>
24      </exchange>
25      <payloadPublication lang="nl" xsi:type="SituationPublication" xmlns:xsi="http://www.w3.org
26        /2001/XMLSchema-instance">
27        <publicationTime>2014-01-15T20:46:56Z</publicationTime>
28        <publicationCreator>
29          <country>nl</country>
30          <nationalIdentifier>NDWNL</nationalIdentifier>
31        </publicationCreator>
32        <situation id="NLRWS_NLSIT001575095">
33          <situationVersion>1</situationVersion>
34          <situationVersionTime>2014-01-15T20:13:46Z</situationVersionTime>
35          <headerInformation>
36            <confidentiality>noRestriction</confidentiality>
37            <informationStatus>real</informationStatus>
38          </headerInformation>
39          <situationRecord xsi:type="RoadOrCarriagewayOrLaneManagement" id="
40            NLRWS_NLSIT001575095_1">
41            <situationRecordCreationTime>2014-01-15T20:13:00Z</situationRecordCreationTime>
42            <situationRecordVersion>1</situationRecordVersion>
43            <situationRecordVersionTime>2014-01-15T20:13:45Z</situationRecordVersionTime>
44            <probabilityOfOccurrence>certain</probabilityOfOccurrence>
45            <source>
46              <sourceIdentification>NLTIC</sourceIdentification>
47              <sourceName>
48                <value lang="nl"></value>
49              </sourceName>
50            </source>
51            <validity>
52              <validityStatus>definedByValidityTimeSpec</validityStatus>
53              <validityTimeSpecification>

```

```

52     <overallStartTime>2014-01-15T20:10:00Z</overallStartTime>
53     <overallEndTime>2014-01-16T04:00:00Z</overallEndTime>
54     <validPeriod>
55         <startOfPeriod>2014-01-15T20:10:00Z</startOfPeriod>
56         <endOfPeriod>2014-01-16T04:00:00Z</endOfPeriod>
57     </validPeriod>
58 </validityTimeSpecification>
59 </validity>
60 <cause xsi:type="NonManagedCause">
61     <causeDescription>
62         <value lang="nl">Wegwerkzaamheden</value>
63     </causeDescription>
64     <causeType>other</causeType>
65 </cause>
66 <generalPublicComment>
67     <comment>
68         <value lang="nl">Het verkeer wordt geadviseerd een andere route te kiezen</value>
69     </comment>
70 </generalPublicComment>
71 <groupOfLocations xsi:type="Itinerary">
72     <locationContainedInItinerary index="0">
73         <location xsi:type="Linear">
74             <locationForDisplay>
75                 <latitude>52.441444</latitude>
76                 <longitude>4.668316</longitude>
77             </locationForDisplay>
78             <supplementaryPositionalDescription />
79             <alertCLinear xsi:type="AlertCMethod4Linear">
80                 <alertCLocationCountryCode>8</alertCLocationCountryCode>
81                 <alertCLocationTableNumber>5.6</alertCLocationTableNumber>
82                 <alertCLocationTableVersion>A</alertCLocationTableVersion>
83                 <alertCDirection>
84                     <alertCDirectionCoded>negative</alertCDirectionCoded>
85                 </alertCDirection>
86                 <alertCMethod4PrimaryPointLocation>
87                     <alertCLocation>
88                         <specificLocation>10591</specificLocation>
89                     </alertCLocation>
90                     <offsetDistance>
91                         <offsetDistance>0</offsetDistance>
92                     </offsetDistance>
93                 </alertCMethod4PrimaryPointLocation>
94                 <alertCMethod4SecondaryPointLocation>
95                     <alertCLocation>
96                         <specificLocation>10594</specificLocation>
97                     </alertCLocation>
98                     <offsetDistance>
99                         <offsetDistance>100</offsetDistance>
100                    </offsetDistance>
101                </alertCMethod4SecondaryPointLocation>
102            </alertCLinear>
103        </location>
104    </locationContainedInItinerary>
105    <locationContainedInItinerary index="1">
106        <location xsi:type="Point">
107            <pointByCoordinates>
108                <pointCoordinates>

```

```
109         <latitude>52.441444</latitude>
110         <longitude>4.668316</longitude>
111     </pointCoordinates>
112 </pointByCoordinates>
113 </location>
114 </locationContainedInItinerary>
115 <locationContainedInItinerary index="2">
116     <location xsi:type="Point">
117         <pointByCoordinates>
118             <pointCoordinates>
119                 <latitude>52.486115</latitude>
120                 <longitude>4.692282</longitude>
121             </pointCoordinates>
122         </pointByCoordinates>
123     </location>
124 </locationContainedInItinerary>
125 </groupOfLocations>
126 <operatorActionStatus>approved</operatorActionStatus>
127 <complianceOption>mandatory</complianceOption>
128 <roadOrCarriagewayOrLaneManagementType>roadClosed</
roadOrCarriagewayOrLaneManagementType>
129 </situationRecord>
130 </situation>
131 </payloadPublication>
132 </d2LogicalModel>
133 </SOAP-ENV:Body>
134 </SOAP-ENV:Envelope>
```

Appendix B

DBPedia Spotlight example results

Code B.1: Results of the DBPedia spotter on an example tweet

```
1 {
2   "annotation": {
3     "@text": "@fileinformatie_ Morning! Afrit #28 naar #A1 #
4     hoevelaken -&gt; A'dam is een drama. A1 bij 41,5 gaat nu weer
5     rijden.. langzaam.. @ANWBverkeer",
6     "surfaceForm": [
7       {
8         "@name": "fileinformatie",
9         "@offset": "1"
10      },
11     {
12       "@name": "A1",
13       "@offset": "42"
14     },
15     {
16       "@name": "hoevelaken",
17       "@offset": "46"
18     },
19     {
20       "@name": "A1",
21       "@offset": "83"
22     }
23   ]
24 }
```

Code B.2: Partially results of the DBPedia candidates service on an example tweet

```

1 {
2   "annotation": {
3     "@text": "@fileinformatie_ Morning! Afrit #28 naar #A1 #
4     hoevelaken -&gt; A'dam is een drama. A1 bij 41,5 gaat nu weer
5     rijden.. langzaam.. @ANWBverkeer",
6     "surfaceForm": [
7       ..
8       {
9         "@name": "hoevelaken",
10        "@offset": "46",
11        "resource": {
12          "@label": "Knooppunt Hoevelaken",
13          "@uri": "Knooppunt_Hoevelaken",
14          "@contextualScore": "0.9999320305275015",
15          "@percentageOfSecondRank": "0.004447682312902582",
16          "@support": "24",
17          "@priorScore": "1.3082359174535842E-6",
18          "@finalScore": "0.9955714958208094",
19          "@types": "Schema:Place, DBpedia:Place, DBpedia:
20          ArchitecturalStructure, DBpedia:Infrastructure, DBpedia:
21          RouteOfTransportation, DBpedia:RoadJunction"
22        }
23      },
24      {
25        "@name": "A1",
26        "@offset": "83",
27        "resource": {
28          "@label": "Rijksweg 1",
29          "@uri": "Rijksweg_1",
30          "@contextualScore": "0.9969469089192675",
31          "@percentageOfSecondRank": "2.139943072195324E-4",
32          "@support": "285",
33          "@priorScore": "1.5535301519761313E-5",
34          "@finalScore": "0.9997315439355874",
35          "@types": "Schema:Place, DBpedia:Place, DBpedia:
36          ArchitecturalStructure, DBpedia:Infrastructure, DBpedia:
37          RouteOfTransportation, DBpedia:Road"
38        }
39      }
40    ]
41  }
42 }

```


Appendix C

Sentiment services example results

Code C.1: Example of sentiment scores and label using the text processing service

```
1 {
2   "probability": {
3     "neg": 0.5,
4     "neutral": 1,
5     "pos": 0.5
6   },
7   "label": "neutral"
8 }
```

Code C.2: Example of a sentiment score and label using the AI Applied service

```
1 {
2   "data": [
3     {
4       "language_iso": "nld",
5       "text": "File%20file%20file%20hoera%21%20Terug%20in%20
6       Nederland.",
7       "confidence_sentiment": 0.7445227630217524,
8       "sentiment_class": "neutral"
9     }
10  ],
11  "description": "OK: Call processed.",
12  "success": true
13 }
```

Appendix D

Linked event example

Code D.1: Tweets linked to an event

```
1 ..
2   "linkedTweets": {
3     "geoTweets": {},
4     "timeAndFullVildBased+": {
5       "10.0%": [
6         {
7           "twitterScreenName": "yourvipdriver",
8           "matchTerms": [
9             "afrit",
10            "a1"
11          ],
12          "matchPercentage": 10,
13          "linkageTimeDate": 1432293242112,
14          "matches": {
15            "secndNameMatch": false,
16            "roadNumberMatch": true,
17            "firstNameMatch": false,
18            "locDesMatch": true,
19            "roadNameMatch": false
20          },
21          "vildTerms": {
22            "first_name": [
23              "muidenberg",
24              "ronduit",
25              "naarden",
26              "bastion",
27              "naarden-west",
28              "a1 vanuit amersfoort",
29              "naarden-vesting"
30            ],
31            "secnd_name": [
32              "a6 richting lelystad",
```

```
33         "a6"
34     ],
35     "loc_des": [
36         "parkeerplaats",
37         "knooppunt",
38         "verbindingsweg",
39         "afrit"
40     ],
41     "roadname": [],
42     "roadnumber": [
43         "a1"
44     ]
45 },
46 "is_bot": false,
47 "id": "450503270482866176_TWEET",
48 "estimatedDistanceToEvent": [],
49 "linkageTime": "2015-05-22T13:14:02"
50 },
51 ..
52 ],
53 "0.0%": []
54 },
55 "timeAndVildBased": {},
56 "timeAndFullVildBased": {}
57 }
58 ..
```

Appendix E

Experiment 2 - Results

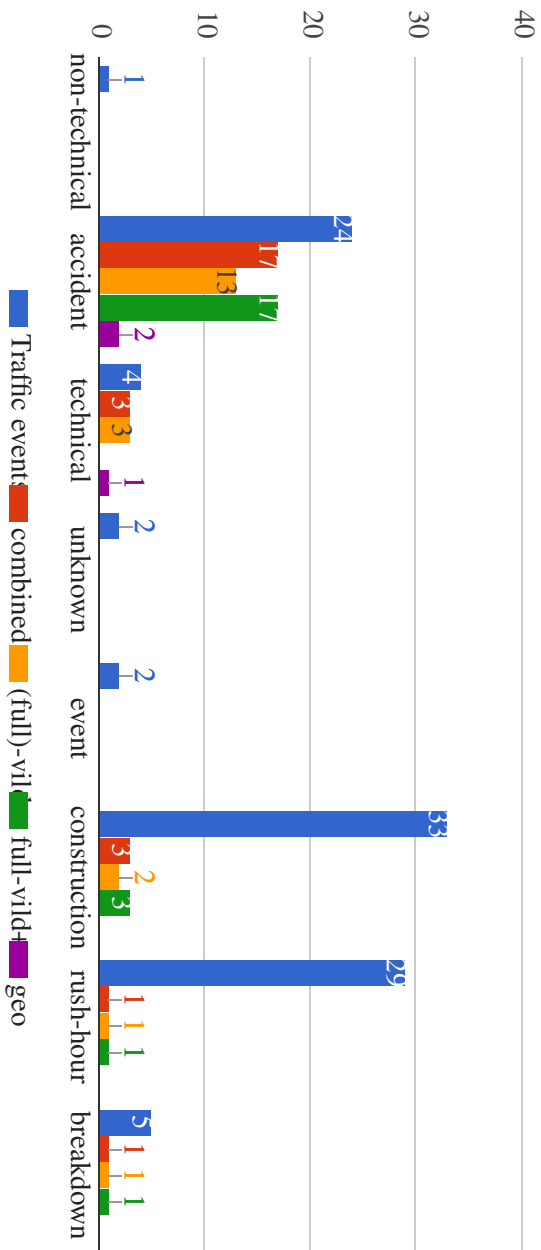


Figure E.1: Number of events with at least one 'related-real' tweet for different cause types and linkage strategies

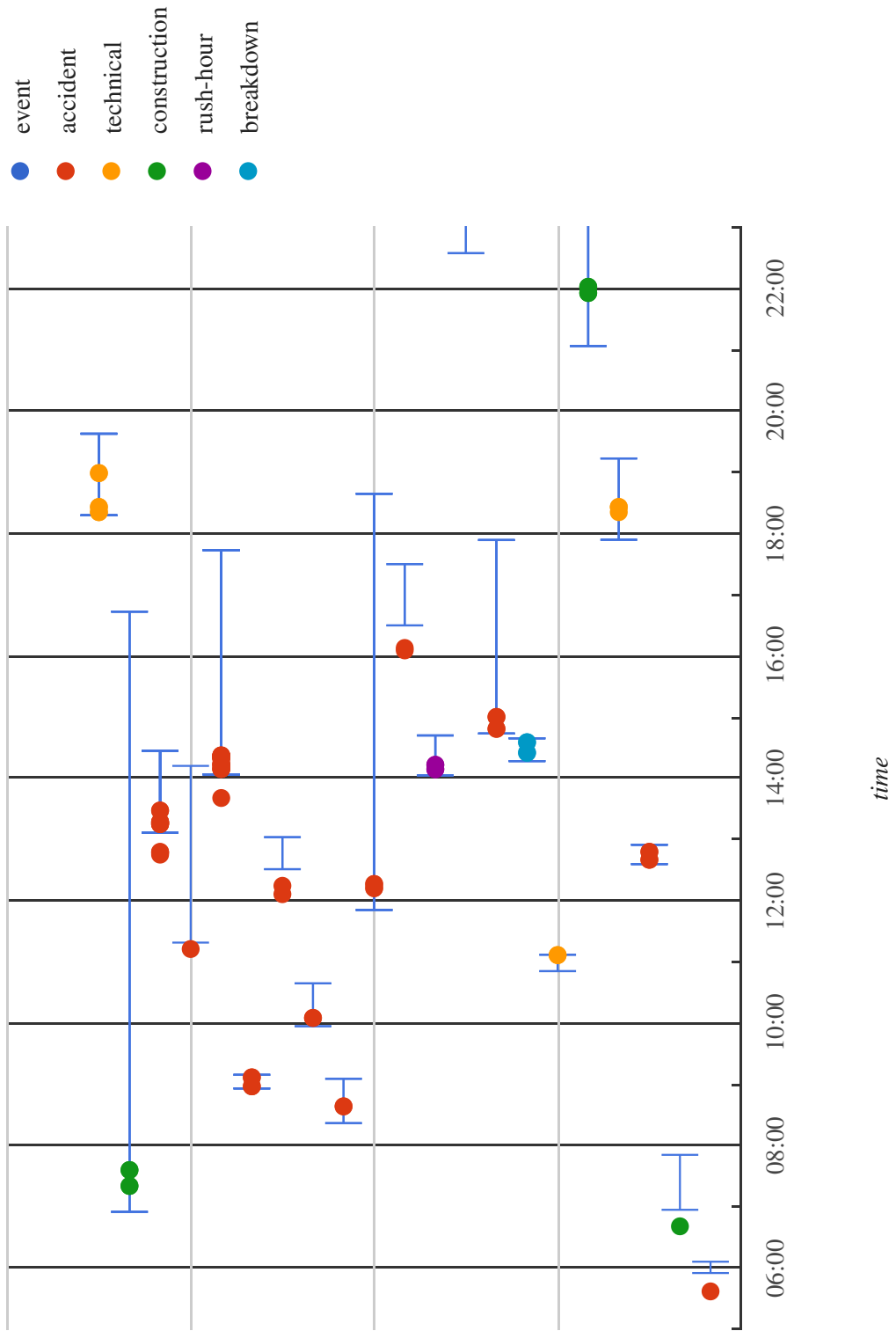


Figure E.2: Spread of tweets related to timespan of traffic events

Appendix F

Glossary

In this appendix we give an overview of frequently used terms and abbreviations.

NDW National Data Warehouse for Traffic information (<http://www.ndw.nu>)

VILD ‘VerkeersInformatieLocatieDatabase (traffic information location database)’. VILD tables describe the more than 25.000 traffic measuring points that exists in the Netherlands. The traffic information consists of properties such as *road name* and *road number*. Each VILD point in the table contains references to at most two other VILD points.

EI Eventinfo; a dataset from the NDW that contains status information about the availability of the roads: road constructions, accidents, traffic jams are examples of updates that occur in Eventinfo datasets.

Traffic event A traffic event is a traffic disturbance: a situation at a certain time and place that causes traffic delay.

ITS Intelligent Transportation System

Bot user A twitter user that is not considered as a real (road) user. Twitter accounts of bot users are specially created to tweet about traffic congestion or news all the time. It does not necessarily mean that the user is truly a bot, sometimes there is still a ‘real user’ behind the profile. We’ve chosen for this name for convenience.