

Enhancing Real-Time Twitter Filtering and Classification using a Semi-Automatic Dynamic Machine Learning setup approach

Master's Thesis

Nick de Jong

Enhancing Real-Time Twitter Filtering and Classification using a Semi-Automatic Dynamic Machine Learning setup approach

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE
TRACK SOFTWARE TECHNOLOGY

by

Nick de Jong
born in Rotterdam, 1988



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

The logo for CrowdSense, consisting of the word "crowdsense" in a white, lowercase, monospace-style font on a dark blue rectangular background.

CrowdSense
Wilhelmina van Pruysenweg 104
The Hague, the Netherlands
<http://www.twitcident.com>

Enhancing Real-Time Twitter Filtering and Classification using a Semi-Automatic Dynamic Machine Learning setup approach

Author: Nick de Jong
Student id: 1308130
Email: ndjong2@gmail.com

Abstract

Twitter contains massive amounts of user generated content that also contains a lot of valuable information for various interested parties. Twitcident has been developed to process and filter this information in real-time for interested parties by monitoring a set of predefined topics, exploiting humans as sensors. An analysis of the relevant information by an operator can result in an estimation of severity, and an operator can act accordingly. However, among all relevant and useful content that is extracted, also a lot of irrelevant noise is present. Our goal is to improve the filter in such a way that the majority of information presented by Twitcident is relevant. To this end we designed an artifact consisting of several components, developed within a dynamic framework. Its major components include a machine learning classifier operating on dynamic features, a semi-automatic setup approach and a training approach. Our prototype operates on Dutch content, but it can be adapted to operate on any language. With a partially implemented prototype of our designed artifact we achieve F_2 -scores of 0.7 up to 0.9 for our Dutch test-sets using 10-fold cross validation, which is on average a 30% improvement over the existing Twitcident filtering architecture. The artifact is robustly designed, allowing for many forms of future improvements and extensions. We also make some side-contributions, like an approximate matching algorithm for variable length strings.

Thesis Committee:

Chair: Prof. dr. ir. G.J.P.M. Houben, Faculty EEMCS (WIS), TU-Delft
University supervisor: Dr. C. Hauff, Faculty EEMCS (WIS), TU-Delft
Company supervisor: R.J.P. Stronkman MSc, CrowdSense
Committee Member: Dr. G.H. Wachsmuth, Faculty EEMCS (SERG), TU-Delft

Acknowledgments

First of all, I would like to thank my parents and my girlfriend Fleur Houben for their unlimited support, endurance, confidence, patience, financial support, feedback and re-motivation during hard times. Without them, I might not have succeeded.

I'd like to thank Geert-Jan Houben, Claudia Hauff and Gytha Rijnbeek of Delft University of Technology not only for their guidance and advice during the project, but mainly for all their patience and flexibility.

I'd like to especially thank my company supervisor Richard Stronkman for his invaluable brainstorming sessions and guidance, support, patience and maintaining a relaxed comfortable atmosphere. I'd also like to thank the rest of the CrowdSense team for their hours of annotating data. Noteworthy mentions are Rense Bakker, Niels Bergsma and Arjan Assink for their swift first line assistance with CrowdSense systems and access.

Nick de Jong
Delft, the Netherlands
August 11, 2015

Contents

Acknowledgments	iii
Contents	v
List of Figures	ix
List of Tables	x
1 Introduction	1
1.1 Background	1
1.2 Actors	2
1.3 Subject matter	2
1.3.1 Terminology	3
1.3.2 Examples	4
1.4 Research objective	7
1.5 Research questions & Outline	9
1.6 Appendices	12
2 Problem analysis	13
2.1 Twitcident	15
2.1.1 Web-App	15
2.1.2 Pipe-App	17
2.2 An introductory case analysis: Project X, Haren	18
2.2.1 Background	18
2.2.2 The case	19
2.2.3 Dataset properties	20
2.2.4 Topic analysis: Bekogeling (Bombardment)	20
2.2.5 Topic analysis: Brand (Fires)	22
2.3 Observing a real world instance: Nationale Politie Limburg	26
2.4 General observations	28
2.5 Challenges and difficulties	28
2.6 Summary: List of observations	30
3 Research	33

3.1	Methodology: Design Science Research	33
3.2	Related work	36
3.2.1	TwitterStand: News in Tweets	36
3.2.2	OMG Earthquake! Can Twitter Improve Earthquake Response?	37
3.2.3	Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors	38
3.2.4	Weak Signal Detection on Twitter Datasets	38
3.2.5	CrisisLex: A Lexicon for Collecting and Filtering Microblogged Communications in Crises	39
3.2.6	Other related work	40
3.3	Technologies	41
4	Design and Implementation of our artifact	47
4.1	Choices made towards a solution	48
4.2	Artifact decomposition	49
4.3	Part 1: Classification	52
4.3.1	List of features	53
4.3.2	Feature discussions and implementations	55
4.3.3	Classifier	59
4.3.4	Implementation specifics	60
4.4	Part 2: Setup Approach	62
4.4.1	Manual setup approach	63
4.4.2	Required inputs	65
4.4.3	Three levels of wordlists	65
4.4.4	Word expansion	66
4.4.5	Setup approach	68
4.5	Part 3: Training Approach	72
4.6	Summary	74
5	Experiments and Evaluation	77
5.1	Datasets	77
5.1.1	Data Collection	77
5.1.2	Resulting datasets	78
5.1.3	Dataset post-processing	78
5.2	Experiments	79
5.2.1	Suitable classifiers	79
5.2.2	Output format	81
5.2.3	Amount of features	82
5.2.4	Classifier performance	84
5.3	Results comparison	84
6	Conclusions and Future Work	87
6.1	Summary	87
6.2	Future work	91
6.3	Contributions	92
	Bibliography	95

A	Features	101
A.1	Feature list	101
A.2	Feature discussions and implementations	103
A.2.1	Keyword matches	103
A.2.2	Hypernyms and hyponyms	105
A.2.3	Username matches	106
A.2.4	Ignore features	106
A.2.5	News and media	106
A.2.6	Tweet and stream rates	106
A.2.7	Sentiment	107
A.2.8	Subjectivity	107
A.2.9	Tense	107
A.2.10	Retweet(s)	107
A.2.11	Reply	108
A.2.12	Mentions	108
A.2.13	Tweet length	108
A.2.14	Images	108
A.2.15	Links	108
A.2.16	Hashtags	109
A.2.17	Prior similar tweets	109
A.2.18	Contains popular term	109
A.2.19	Distance from hotspot	109
A.2.20	Seems observation	110
A.2.21	Street language and slang	110
A.2.22	Part of Speech attributes	110
A.2.23	Future or past days mentioned	110
A.2.24	Contains special keywords	110
A.2.25	Language	111
A.2.26	Interpunction	111
B	Approximate String Matching algorithms	113
B.1	Related work	113
B.2	Problem definition	114
B.3	Our solution	114
B.4	Demonstration	115
B.5	Performance considerations	116
C	Reproducibility information	117
C.1	SQL statements	117
C.1.1	Database structure, schema: Classification	117
C.1.2	Database structure, schema: DataAnnotator	118
C.1.3	Database structure, schema: TwitterBase	119
C.1.4	Initial Project X analysis	121
C.2	Implementation specifics	121
C.3	Word Expansion Algorithm	125
D	Potential data sources	127

E	Example word expansions	131
E.1	Aardbeving	131
E.2	Auto	131
E.3	Bank	133
E.4	Brand	136
E.5	Bureau	137
E.6	Regen	139
E.7	Trillen	140
E.8	Veeg	140
F	Data Collection	143
F.1	Strategy Considerations	143
F.1.1	What?	143
F.1.2	Who?	143
F.1.3	How much?	144
F.2	Data Annotation Application	144
F.3	Data Collection Algorithm	146
F.4	Output Processing Algorithm	148
G	Analysis of Prinsjesdag 2013	149
H	Setup for Event Detection	173

List of Figures

1.1	Twitcident logo (2014)	2
1.2	Actor relationships	3
1.3	Thesis outline	11
2.1	Twitcident Application: Dashboard	14
2.2	Twitcident Application: Map	15
2.3	Twitcident Application: Tweets View	16
2.4	Twitcident Application: Keywords	17
2.5	Twitcident Application: Ignore terms	17
2.6	Twitcident Application: Ignore users	17
2.7	Twitcident Pipeline	18
2.8	Project X	19
2.9	Project X: General dataset properties	21
2.10	Topic keywords for Project X: Brand (Fires)	24
2.11	A joke about ‘Haren’ (also: ‘Hair’) on ‘Fire’	25
2.12	Twitcident Instance: National Police Limburg	26
3.1	Related work: TwitterStand system architecture	36
4.1	Two-fold problem decomposition	48
4.2	Architectural overview and context of our artifact	50
4.3	Artifact decomposition	51
4.4	Artifact decomposition: Classifier System & Features highlighted	52
4.5	Classifier System: Actually a set of parallel classifiers	52
4.6	Artifact decomposition: Classifier System highlighted	60
4.7	Database table: a random subset of a feature configuration	61
4.8	Database table: a random subset of some feature inputs	62
4.9	Artifact decomposition: Setup Approach highlighted	62
4.10	Our working strategy to devise the Setup Approach	63
4.11	A truncated example word expansion of the word ‘brand’ (‘fire’)	67
4.12	Artifact decomposition: Training Approach highlighted	72
4.13	Artifact decomposition	75
4.14	Artifact overview: context	76

5.1	Classifier comparison for numerical output	80
5.2	Classifier comparison for nominal output	80
5.3	Classifier comparison between top three	80
5.4	Comparison of nominal class output format	81
5.5	Comparison of amount of enabled features for numerical classification . .	83
5.6	Comparison of amount of enabled features for nominal classification . . .	83
5.7	Final performance measures for our artifact	84
5.8	Comparison of existing Twitcident performance for various boundary values	85
5.9	Comparison of performance between our designed artifact and existing Twitcident application	86
6.1	Artifact decomposition	88
6.2	Comparison of performance between our designed artifact and existing Twitcident application	90
B.1	Approximate substring matching: Test cases	115
B.2	Approximate substring matching: Results on test cases	116
C.1	Database table: a random subset of a feature configuration	123
C.2	Database table: a random subset of some feature inputs	123
C.3	Extended ARFF format	124
F.1	Data Annotation Application: first annotation phase	144
F.2	Data Annotation Application: second annotation phase	145

List of Tables

1.1	Example Nationale Spoorwegen: Twitcident Streams	4
1.2	Example Nationale Spoorwegen: Twitcident Topics	5
1.3	Example ProRail (1): Twitcident Streams	5
1.4	Example ProRail (1): Twitcident Topics	5
1.5	Example ProRail (2): Twitcident Topics	6
1.6	Example Gaswinning Groningen: Twitcident Topics	7
1.7	Examples of relevant and irrelevant tweets related to ‘Earthquake’-topic in Gaswinning Groningen instance	8
1.8	Examples of relevant and irrelevant tweets related to ‘Seats’ in instance Nationale Spoorwegen	8

2.1	Project X: General dataset properties	21
2.2	Topic keywords for Project X: Bekogeling (Bombardment)	22
2.3	Examples of two relevant tweets relating to Bombardment with bikes	23
2.4	Topic keywords for Project X: Brand (Fires)	24
2.5	Examples of interesting tweets matched by the keyword 'brand' (fire)	25
2.6	Twitcident Instance: National Police Limburg	26
2.7	Examples of noteworthy tweets classified as 'Burglary'	27
2.8	Examples of noteworthy tweets classified as 'Shooting'	27
3.1	Design Science Research guidelines	35
4.1	List of features, part 1	53
4.2	List of features, part 2	54
4.3	List of features, part 3	55
4.4	Illustration match value for different matching schemes	57
4.5	List of required instance-dependent feature inputs	65
4.6	Example WordEntry weight calculation for base term 'aardbeving' and combo terms 'trillen' and 'lawaaai' using $\alpha = 1.0$ and $\beta = 10$	69
5.1	Annotated datasets	78
5.2	Nominal classes	79
5.3	Subsets of enabled features	82
A.1	List of features, part 1	101
A.2	List of features, part 2	102
A.3	List of features, part 3	103
A.4	Illustration match value for different matching schemes	105
D.1	Setup sources: dictionaries	127
D.2	Setup sources: semantic sources	127
D.3	Setup sources: synonyms	128
D.4	Setup sources: encyclopedia	128
D.5	Setup sources: proverbs	128
D.6	Setup sources: miscellaneous	129

Chapter 1

Introduction

In this chapter we will introduce the context of Twitcident in relation to this thesis: What is Twitcident? What is it used for? By whom? Who else is involved? After these questions are answered, we will discuss the problem: What is the research objective? How will we approach the problem? Finally, we will relate this approach to the structure of this thesis.

1.1 Background

Since the commencement of Web 2.0, people are able to contribute to the Web through social services like Twitter¹ and Facebook². Nowadays more than 215 million active users post more than 500 million tweets a day³. These numbers have significantly been growing over the past few years: in 2010, 190 million active users only posted 65 million tweets a day [56].

Such massive amounts of user generated content contain a lot of valuable information for various interested parties. During crisis situations such as large fires, storms or other types of incidents, people nowadays quickly report and discuss about their observations, experiences, and opinions in various Social Web streams. This enormous flow of information contains vital information which cannot be processed adequately by default means. These very fast notifications with text and often media (images, links, videos) compose a huge pile of up-to-date but unstructured data.

A lot of extraction and filtering tools have been developed over the past years with various aims. Among them is Twitcident⁴, initially developed by Stronkman in 2011 [1, 2, 56, 57]. This technology is able to monitor a specific set of topics, events or areas in real-time, providing the monitoring client with potentially valuable information. Twitcident sifts through and analyzes social big data in real time for local authorities, police, emergency services and vital infrastructure operators. It basically exploits humans as sensors, filters the signal coming from all those sensors and attempts to extract the relevant meaningful information. An analysis of the relevant

¹<http://www.twitter.com/>

²<http://facebook.com/>

³<http://www.sec.gov/Archives/edgar/data/1418091/000119312513390321/d564001ds1.htm>

⁴<http://www.twitcident.com/>



Figure 1.1: Twitcident logo (2014)

information by an operator can result in an estimation of severity, and an operator can act accordingly.

However, among all relevant and useful content that is extracted, also a lot of irrelevant noise is present. We will thoroughly address this issue and attempt to solve it.

1.2 Actors

This thesis work is relevant to multiple actors:

CrowdSense CrowdSense is the company that develops and maintains Twitcident, and has a partnership regarding Twitcident with *Delft University of Technology* and *TNO*. Their primary interest regarding this thesis is a contribution to Twitcident that significantly improves the data output quality, and optimizing the configuration process (explained in Chapter 2). However, as a significant part of their clients comprise of public services and (governmental) authorities, this also serves as a societal contribution.

Delft University of Technology Delft University of Technology is the university facilitating this thesis work. They will support the scientific process and contribution.

Twitcident Client(s) Twitcident Client(s) are companies and/or authorities that purchased a Twitcident license from *CrowdSense*, using Twitcident to monitor their target areas, vital infrastructures or events. With the output quality of Twitcident improving, they can optimize their (business) processes with less effort.

Twitcident Operator(s) Twitcident Operator(s) are those people within a *Twitcident Client* that actively operate the Twitcident application, monitor its output and respond appropriately to it. With the output quality of Twitcident improving, their tasks become easier, more accessible and more efficient.

1.3 Subject matter

Within this work, it is assumed the reader is familiar with Social Media and especially Twitter, understanding the concepts of Tweets, Re-Tweets, Replies, Mentions and Hash-tags⁵. Especially for chapter 4, we also assume a background in Computer

⁵If not, an introduction to Twitter can be found at <http://mashable.com/guidebook/twitter/>

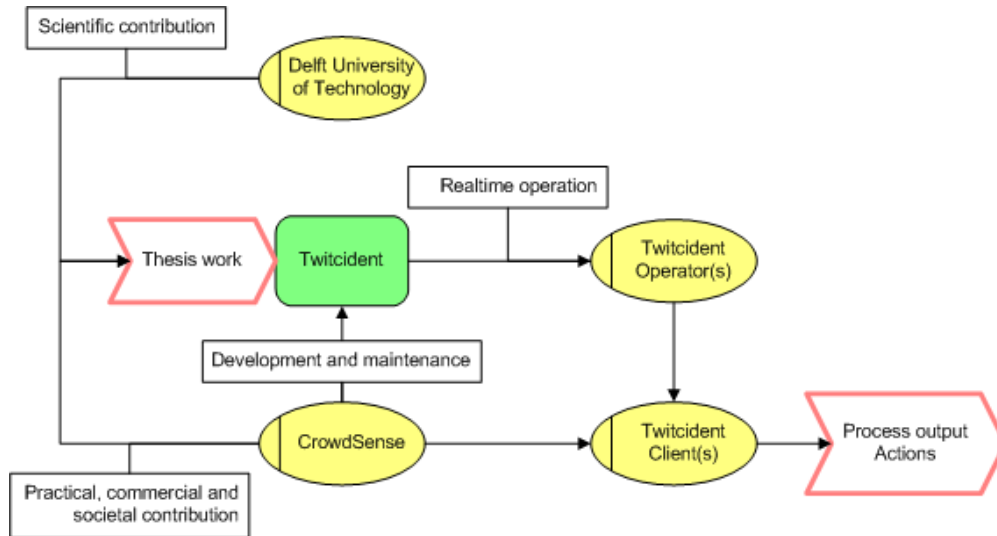


Figure 1.2: Actor relationships

Science. We will be making use of technologies like Machine Learning and Classifiers, of which familiarity is advised.

This work will be building upon earlier thesis work by Stronkman [56]. However, between his work and this work, the current state and setup of Twitcident has evolved remarkably. Stronkman describes the system layout as consisting of three chained black boxes, respectively Incident Detection, Tracking and Location. As the application has been put into commercial use for Twitcident Clients, and as a consequence focus shifted away from regular public users, this structure has been replaced by a more selective data collection approach based on individual clients needs. Emergency broadcast services are no longer the main detection source, as individual clients already possess this information and often have published it themselves. Rather, they are more interested in information that they have not obtained yet through default means, and employ Twitcident as an additional detection channel using twittering people as sensors. In section 2.1 the current architecture of the current *Twitcident 1.0* will be explained thoroughly.

1.3.1 Terminology

In this thesis we will be making extensive use of three Twitcident specific terms:

Twitcident Instance A Twitcident Instance is a specific preconfigured instance of the Twitcident application for a client. Sometimes also called a monitor; these terms are interchangeable. Most Twitcident clients possess a single Twitcident Instance.

Twitcident Stream A Twitcident Stream is a set of keywords, usernames or geo-locations on the basis of which tweets are requested from the Twitter API, and are used for diverging *Data Collection*. A Twitcident Instance can have one or

more Twitcident Streams. Multiple streams indicate a categorization of incoming data, for example, by catchment subarea.

Twitcident Topic A Twitcident Topic is a specific topic a client is interested in, and are used for converging *Data Filtering and Classification*. Each incoming tweet from a Twitcident Stream can be classified as being relevant to a Twitcident Topic or not. A Twitcident Instance usually has four to ten topics. Note that theoretically a single tweet is allowed to be classified as belonging to multiple topics.

1.3.2 Examples

To illustrate the terminology defined above, we consider a few random real-world Twitcident Instances.

1.3.2.1 Nationale Spoorwegen

Nationale Spoorwegen (NS) is the primary Dutch railway operator for public transport by train. Although they have systems to identify whether trains are having trouble, and actively measure the occupancy rate of their trains, they only measure what they should measure and thus can miss things.

In table 1.1 the Twitcident Streams are listed that have been configured for their Twitcident Instance. As we can see this is a geographical partitioning by the Dutch provinces, with the major metropolitan provinces split into subareas by main tracks. These streams are collected by railway station keywords as well as geographical GPS bounding boxes. In table 1.2 the Twitcident Topics are listed.

Groningen
Friesland
...
Limburg
Zeeland
Utrecht: Richting West
Utrecht: Richting Oost
Noord Holland: Amsterdam-Den Helder
Noord Holland: Amsterdam-Haarlem
Noord Holland: Amsterdam-Schiphol
Zuid Holland: Rotterdam-Gouda
Zuid Holland: Rotterdam-Den Haag
Zuid Holland: Den Haag-Leiden

Table 1.1: Example Nationale Spoorwegen: Twitcident Streams

Category	Topic (NL)	Topic (EN)	Description
Impact	Zitplaatsen	Seating	Availability of seats, usually indicates overcrowding or trains that are too short.
	Vervangend vervoer	Alternate transport	If trains are unavailable due to (track-)maintenance, NS is responsible for alternate transport.
	Aansluiting gemist	Missed connections	Everything related to missed connections, potentially indicates too tight scheduling or usual delays.
	Vertraging	Delays	General delays, an additional sensing channel next to other means.
Calamiteiten (Calamities)	Met persoon	With person	Everything related to collisions with people (i.e. jumpers).
	Met object	With object	Everything related to collisions with objects (i.e. vehicles).
	Storing	Failures	General failures, an additional sensing channel next to other means.

Table 1.2: Example Nationale Spoorwegen: Twitcident Topics

Amsterdam
Schiphol
Rotterdam
Utrecht
Schiphol
...
Leeuwarden
Nijmegen

Table 1.3: Example ProRail (1): Twitcident Streams

Category	Topic (NL)	Topic (EN)	Description
Weer (Weather)	Regen	Rain	
	Sneeuw	Snow	
	IJzel	Frost	
	Hagel	Hail	
	Mist	Mist	
	Wind	Wind	
	Onweer	Thunderstorm	
Reizigers (Travelers)	Gestrand	Stranded	During heavy weather situations travelers get stranded more often. ProRail has an interest in this matter due to being responsible for evacuating them, and this being a major political issue.

Table 1.4: Example ProRail (1): Twitcident Topics

1.3.2.2 ProRail (1)

ProRail is a government task organization that maintains and extends the Dutch railway network.

The first of two Twitcident Instances of ProRail monitors observed weather. Although there are weather forecasts, it remains to be seen whether these predictions are correct. People are very keen on observing specific weather conditions and this information can be invaluable to take preemptive action, for example if rail switches are frozen due to snow.

In table 1.3 their Twitcident Streams are listed. This is a partitioning by the major train stations and adjacent trajectories. These streams are collected by city keywords as well as geographical GPS bounding boxes. In table 1.4 the Twitcident Topics are listed.

1.3.2.3 Prorail (2)

The other Twitcident Instance of ProRail is used for Asset Management. People are able to notice defects and litter around train stations, for which ProRail is responsible.

As this concerns the railway infrastructure that is also relevant to NS, the streams are roughly the same as those depicted in table 1.1: this is a partitioning by the major train stations and adjacent trajectories. These streams are collected by city keywords as well as geographical GPS bounding boxes. In table 1.5 the Twitcident Topics are listed.

Category	Topic (NL)	Topic (EN)	Description
Overwegen (Railway crossings)	Defect	Defect	
	Gevaar	Danger	
	Bellen	Ringers	For example: the bells that keep on ringing at crossings
Geluid (Noise)	Overlast	Disturbance	Everything related to noise disturbance caused by trains
Bovenleiding (Caternaries)	Defect	Defect	
	Vonken	Sparks	
Overig (Misc.)	Open hekken	Open gates	Open gates along tracks or around stations
	Verdacht	Suspicious	For example, people walking along tracks
Stations (Railway stations)	Borden	Signs	
	Omroep	Broadcast	
	Voorziening	Facilities	
	Veiligheid	Safety	
Spoor (Rail)	Beleving	Sentiment	Mainly focused on sentiment related to rail maintenance
Vandalisme (Vandalism)	Koperdief	Copper thief	
	Rotzooi	Garbage	
	Muntjes	Coins	

Table 1.5: Example ProRail (2): Twitcident Topics

Category	Topic (NL)	Topic (EN)
Sentiment	Nieuws	News
	Sentiment	Sentiment
Aardbevingen (Earthquakes)	Aardbeving	Earthquake
	Slachtoffers	Casualties
	Schade woning	Property damage
	Schade omgeving	Environmental damage
	Uitval voorzieningen	Public provision failures
	Brand	Fires

Table 1.6: Example Gaswinning Groningen: Twitcident Topics

1.3.2.4 Gaswinning Groningen

In Groningen, one of the Dutch provinces, shale gas is being extracted to the dismay of many, mainly because it increases the probability earthquakes occur. Therefore, a monitor has been setup to detect earthquakes, identify the resulting damage, and track sentiment around earthquakes.

This Twitcident Instance is configured with only one Twitcident Stream: it is composed of city keywords within Groningen as well as a geographical GPS bounding box for this area. In table 1.6 the Twitcident Topics are listed.

1.4 Research objective

Our goal is to improve the filter in such a way that the majority of information presented by Twitcident is relevant. In order to achieve this, we must reduce the noise among presented tweets, and detect whether tweets are relevant in the current situation. Let us consider an example from a topic *Aardbeving* (*Earthquake*) which serves to detect earthquakes. In table 1.7 some examples from our database containing the word ‘aardbeving’ are listed. Most of these are definitely irrelevant, and only a few are relevant. To illustrate the difference, we have explicitly searched for a few relevant tweets.

Another list of examples for a totally different instance would be the topic *Zitplaatsen* (*Seats*) in Nationale Spoorwegen, as shown in table 1.8. We narrowed the examples within this topic down to one of the keywords: *vol* (*full*), in conjunction with the streaming keywords (trains, stations, ...). All of these items would be presented in the current state of Twitcident. We can clearly observe that the context of the keyword *vol* (*full*) is very relevant. To take this one step further: a word can also have multiple meanings. For example, an instance might be searching for the keyword *dood* (*death*) with the intent of locating casualties and disturbances. Now if someone tweets *Ik verveel me dood* (*I’m bored to death*) this item would be presented, while it clearly is irrelevant.

Furthermore, the configuration of a Twitcident Instance is currently done completely manual: all input keywords, which can be a lot, are defined by hand by Crowd-

⁶An explanation of when tweets are irrelevant follows in Chapter 2, but in this (and most) cases, tweets from news media are irrelevant.

⁷This one would be relevant to the sentiment topic, but not relevant to the earthquake detection topic.

Tweet text	Relevant?
Lichte aardbeving in Noord-Groningen http://t.co/PtiJgQe2bI <i>Little earthquake in Groningen North http://t.co/PtiJgQe2bI</i>	No ⁶
Leek wel even een hele kleine aardbeving .. <i>That seemed like a very small earthquake ..</i>	Yes
Dus, kans op zware aardbeving in #groningen. Ook een manier om je los te maken van Nederland <i>So.. risk of severe earthquakes in #groningen. Another way to cut loose from Netherlands</i>	No ⁷
Pfieww onweer vannacht leek wel een aardbeving elke keer als die donder en bliksem tegelijk kwamen. <i>Pfieww thunderstorm tonight seemed like an earthquake every time that thunder and lightning came together.</i>	No
Wow. Leek net wel een aardbeving te zijn in Limmen. Hele huis trilde en bromde. Buren hoorden t ook, maar voelden niets. <i>Wow. There just seemed to be an earthquake in Limmen. Whole house shook and rumbled. Neighbors heard it too, but felt nothing.</i>	Yes
Ik rammel van de honger <i>Literally: 'I'm shaking of hunger' meaning 'I'm starving'</i>	No
En daar was er weer een aardbeving in Appingedam. 00.55 uur. Stond even behoorlijk te schudden. <i>And there was an earthquake in Appingedam. 00:55 pm. Was shaking severely for a moment.</i>	Yes
Binnenkort aardbeving verwacht in groningen wegens extreem gewicht maar we maken het wel weer gezellig samen #familiediner <i>Soon an earthquake is expected in Groningen due to extreme weight but we will make it cozy together #familydinner</i>	No

Table 1.7: Examples of relevant and irrelevant tweets related to 'Earthquake'-topic in Gaswinning Groningen instance

Tweet text	Relevant?
Overvolle afvalcontainers in Lunetten en Zuilen (pvda.nl) http://t.co/Oq0sNjWATj #Utrecht <i>Overflowing wastebins in Lunetten and Zuilen (pvda.nl) http://t.co/Oq0sNjWATj #Utrecht</i>	No
Onderweg naar schiphol. Trein vol met Feyenoord supporters.. <i>On my way to Schiphol. Train full of Feyenoord supporters..</i>	No
Die intercity richting RDam CS is altijd vet vol, tot Utrecht. Daar stapt de halve kudde uit. <i>Intercity from RDam CS is always overly full, up to Utrecht. Then half the herd gets off.</i>	Yes
Liefde voor t personeel etos station den bosch :) Toch nog een zegeltje gehad dus mn spaarkaart vol #yeah <i>Love for the staff of etos station den bosch :) Still received a stamp so my savings card got filled #yeah</i>	No
zo ziet intercity van Adam naar Hsum eruit. Het is iedere dag te vol, maar dit...? http://t.co/6NAv8MZZsx <i>This is what the intercity Adam to Hsum looks like. Every day it is too full, but this...?</i> http://t.co/6NAv8MZZsx	Yes

Table 1.8: Examples of relevant and irrelevant tweets related to 'Seats' in instance Nationale Spoorwegen

Sense employees, consuming a lot of time and inspiration. We will further discuss this issue in section 2.1, but our goal is also to optimize and streamline this setup process.

1.5 Research questions & Outline

Now that we have illustrated why we are actually in need of a better filtering mechanism, we can state our main research question:

How can we improve the results of real-time social media filtering with respect to Twitcident?

In order to do this, we first have to get a good grasp of the problem, explore where the noise comes from and what constraints are imposed by our specific problem. Therefore, in Chapter 2 we will specify the problem in detail and answer the following questions:

*What characterizes Twitcident and what implications does that have?
What properties or aspects of tweets (could) cause noise?
What are the specific problem properties and constraints imposed by the needs that Twitcident tends to fulfill?*

If we have these questions answered, we will have the problem defined and we might conclude that the problem is non-trivial. We should perform rigorous research and check for related work and solutions. We will also explore and state what methodologies are suitable to solve our problem. In Chapter 3 we will answer the following questions:

*What research methodology are we going to apply to solve our problem?
What related work has been performed? How does this work relate to our problem? What are the relevant contributions that could be applicable to our problem?
Which scientific methodologies and technologies are suitable to (partially) solve our problem?*

When we have done our research, we can design and compose our artifact. However, we will not be able to apply all suitable technologies and need to make some choices. Furthermore, we may need to scope our problem and introduce some constraints or assumptions under which we will design our artifact. Therefore, in Chapter 4 we will answer the following questions:

*What choices have been made towards a solution? What constraints or which assumptions have we imposed while designing our artifact?
What artifact have we designed to solve our problem?
How does each of the components of this artifact work and why have we designed it like that?*

After our artifact is designed, we need to evaluate it. We first need to obtain and collect datasets as ground truth. As our artifact undoubtedly will have variables to be filled in, we need to experiment with it to find a good configuration. When we have

found a suitable configuration, we need to compare the performance of our artifact to the existing Twitcident application in order to validate if it indeed improves the results, which then answers our main research question. Finally, we want to know what we can learn from the output and performance of our artifact. Therefore, in Chapter 5 we will answer the following questions:

How can we obtain good representative ground truth datasets?

What is a good performing configuration of our artifact? How well does it perform?

Does our artifact actually improve the results with respect to the existing Twitcident application?

What can we learn from the artifacts performance and output?

Note that the combination of Chapter 4 and a positive answer to third question also answers our main research question.

We will conclude with Chapter 6. We will summarize our thesis work, and reflect on each chapter with conclusions. We will continue with a thorough discussion of future work to follow up this thesis work. Finally, we will list and discuss our contributions: both scientific as well as societal. This final chapter will answer the following questions:

What work have we performed and what can we conclude?

What future research and work could be done to continue or improve our artifact?

What contributions did we make with this thesis work?

Although our artifact is designed to in the context of Twitcident, its application is not limited to it but can be generalized to any real-time social media filtering system under the same constraints extracted in Chapter 2.

Figure 1.3 depicts a graphical representation of this thesis outline and the research questions with their corresponding chapters.

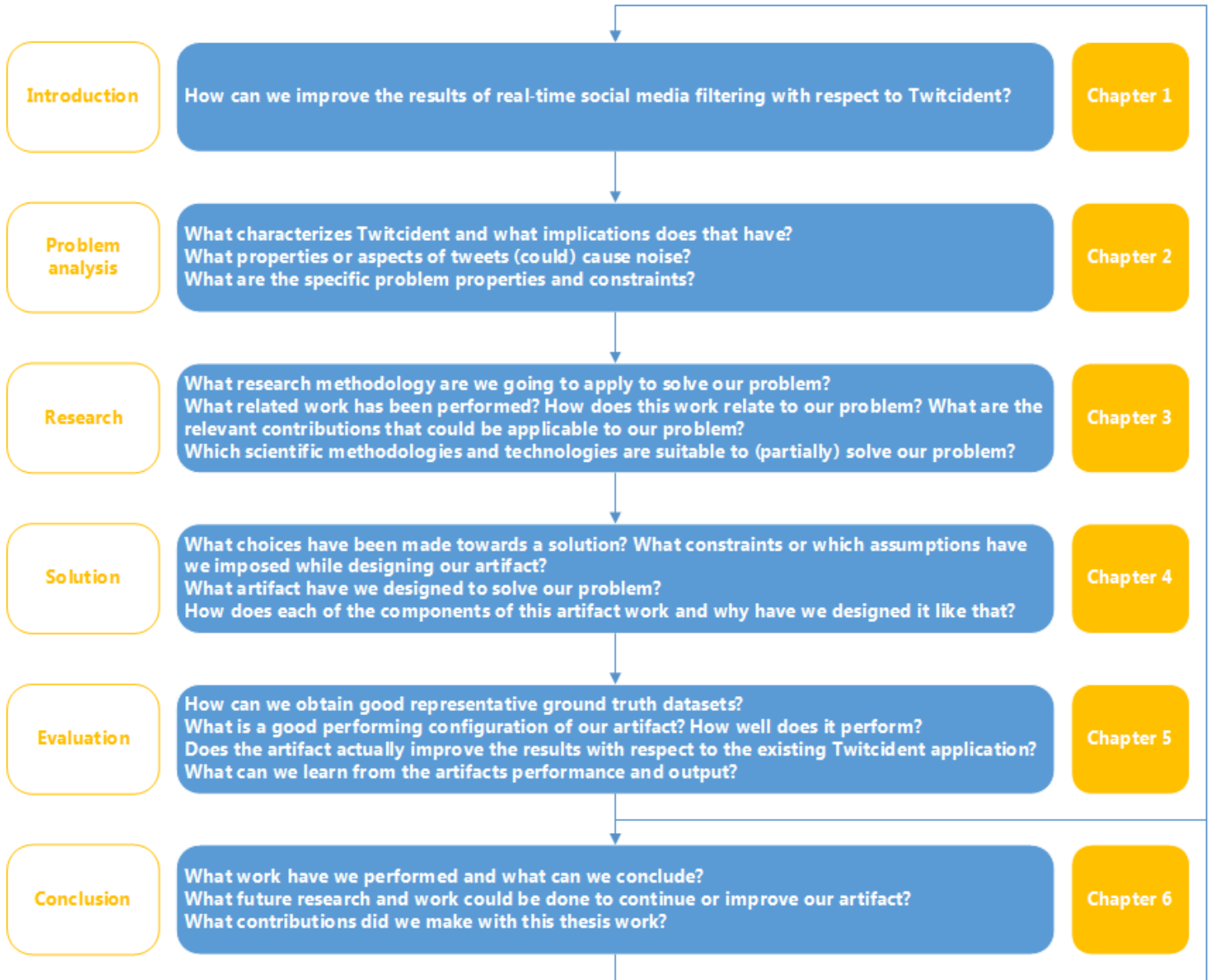


Figure 1.3: Thesis outline

1.6 Appendices

This thesis also contains a few appendices, of which some are contributions on their own. We will briefly mention each appendix and its context. They are chronologically ordered by their first real reference within this thesis.

Appendix A The features are first introduced in Chapter 4, but are not all elaborated upon. This appendix contains the full feature list, including more in-depth discussions of each feature. This can be regarded a separate contribution, and could be used as a basis for further research on extending, optimizing and fine-tuning the set of features.

Appendix B This appendix contains a variable length approximate string matching algorithm we developed during our thesis work. Although this was not a requirement of this thesis work, it is a separate additional contribution and contains relevant implementation details that should not be ignored. It is referenced as part of one of the features.

Appendix C This appendix contains reproducibility information and output listings that were too large or detailed to put within the thesis content itself, in order to be less distracting. It is referenced a few times within Chapter 4.

Appendix D This appendix contains an analysis of potential Dutch data sources we could interact with as introduced in Section 4.4.2. They serve as more comprehensive listings for completeness.

Appendix E This appendix contains a few complete word expansions, as introduced and explained in Section 4.4.4. They serve as more comprehensive listings for completeness.

Appendix F This appendix elaborates on our method to obtain training and test data. It is an essential part of our work, but is too comprehensive to present within the main thesis flow and is therefore discussed separately.

Appendix G This appendix attaches an analysis of the Dutch “Prinsjesdag” tweets, with some conclusions, that spinned of during this thesis work.

Appendix H This appendix follows up appendix G and present ideas about event detection that arose during the Prinsjesdag analysis. Event detection is another topic very relevant to Twitcident, and relates to feature 55 (Has popular term). This can be regarded as a separate additional contribution.

Chapter 2

Problem analysis

In Chapter 1 we introduced the problem, the need for a solution and mildly explored what the objective of this thesis is, without going into too much depth. To continue, we first have to get a good grasp of the problem, explore where the noise comes from and what constraints are imposed by our specific problem.

Because we will be aiming our research towards the needs defined by Twitcident [56], the first question we need to answer to define our problem is:

What characterizes Twitcident and what implications does that have?

We will look into this question in Section 2.1, and discuss the present state of Twitcident.

Our main goal is to reduce noise within tweets, so we will ask ourselves

What properties or aspects of tweets (could) cause noise?

We will perform an analysis and approach this using an observation-style approach in Sections 2.2–2.4, making note of what we observe, so we can use them as a foundation for our designed artifact in Chapter 4. These observations will later be referred to. We will tend to consider something a (relevant) observation if it either seems to be a property that may influence the relevancy of a tweet, or if it affects the design of our artifact, for example, if it imposes a potential constraint on our solution. All observations referenced are listed in Section 2.6 at the end of this Chapter.

Finally, based on what we have observed in previous sections and grasped the problem on a tweet level basis, we can determine broader additional aspects of the problem that make it particularly different and which define the differentiating uniqueness of our problem. Therefore, in Section 2.5 we will answer the following question:

What are the specific problem properties and constraints imposed by the needs that Twitcident tends to fulfill?

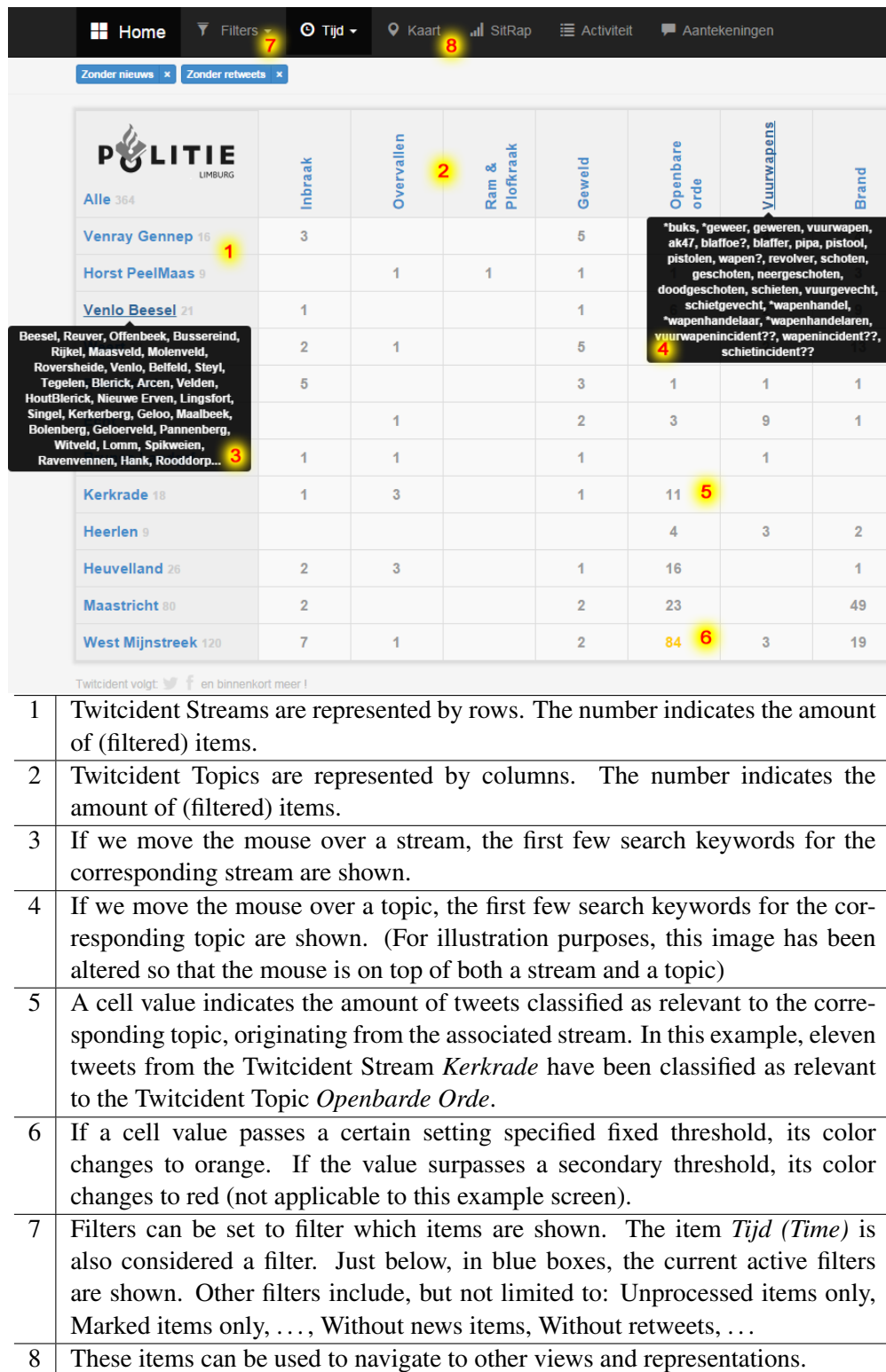


Figure 2.1: Twitcident Application: Dashboard

2.1 Twitcident

In this section we will present a demonstration of the present state of Twitcident, to feature a more concrete context to which we will develop our solution. We will be walking through the Twitcident User Interface, discuss some features and also make our first observations here. The Twitcident project is currently split up into two applications:

- Web-App. This web-application serves as the front-end application for the end-user (Twitcident Operator), and is used to present the data in different ways.
- Pipe-App. This web-application serves as the back-end application for maintenance, and also performs the retrieval and classification of data.

2.1.1 Web-App

For this demonstration we have selected *Nationale Politie Limburg (National Police)* as our Twitcident Instance for practical reasons. When the user opens the Web-App through a secure portal and provides his login credentials, the main screen is presented (Figure 2.1) which is called the dashboard. We can see that the interface is primary composed of a matrix layout, where rows represent the Twitcident Streams, and columns represent the Twitcident Topics.

If one clicks the *Map* tab, a heat map is shown where tweets are originating from (Figure 2.2). This map automatically re-calibrates when the user zooms in, as to set the desired level of granularity for locating hotspots. Note that only a fraction of tweets have an attached geo-location, and only those tweets can be shown.

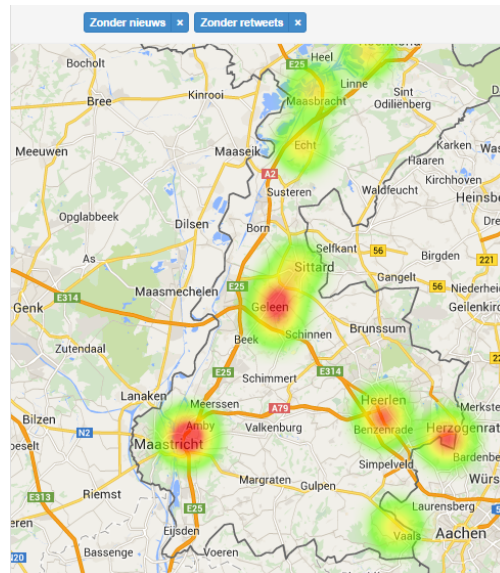
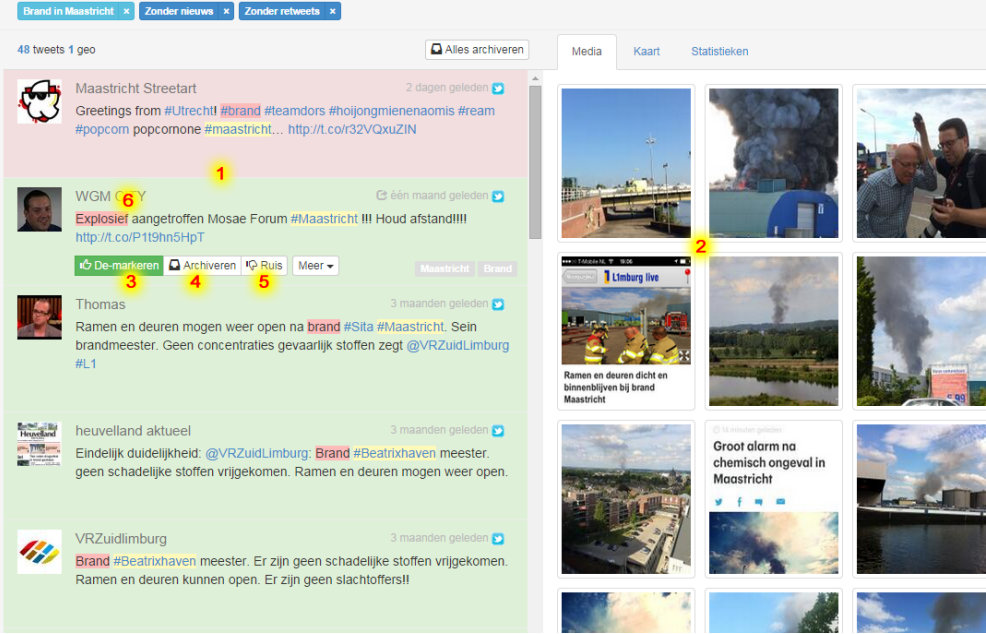


Figure 2.2: Twitcident Application: Map

When the user clicks on a stream name, topic name or matrix cell on the dashboard, Twitcident navigates to the Tweet View (Figure 2.3). Here, we will also draw our first



1	All filtered tweets are shown to the left. Tweets can be selected and various actions can be performed upon them.
2	All media contained within the filtered tweets is displayed on the Media tab.
3	Tweets can be marked. Marked tweets will receive a green background-color. Interpretation of a tweet being <i>marked</i> is left open: it can indicate a relevant item, or that it needs follow-up. This is left to the Twitcident Operator(s).
4	Tweets can be archived. Archived tweets are not shown by default.
5	Tweets can be flagged as noise. Noisy tweets are not shown by default.
6	Within the tweet text, the keywords that caused the tweet to be matched to this topic are highlighted: red for regular keywords, yellow if the keyword was encountered as a hashtag.

Figure 2.3: Twitcident Application: Tweets View

observations, that will be relevant to our solution presented in Chapter 4. Note that the Observations referenced are located in Section 2.6. In the Twitcident user interface, a feedback mechanism is present: Tweets can be flagged as marked (relevant?) or noise (Observation 1). We also observe that Twitcident Topic classification is currently solely done by matching on predefined static keywords (Observation 2).

If the Twitcident Operator navigates to the *Instellingen (Settings)* pane, he can edit keywords for each Twitcident Topic (Figure 2.4). We note that two wildcards “?” and “*” are currently supported. Furthermore, if a “keyword” consists of multiple words, all words have to be present to be matched where the order in which they occur is not taken into consideration (Observation 3).

Additionally, for each Twitcident Topic *ignore terms* can be specified (Figure 2.5). We note that these are only used in conjunction with regular keywords (Observation 4). Therefore, if a tweet would be classified according to a keyword, but a corresponding ignore term is present, it won’t be classified as such.

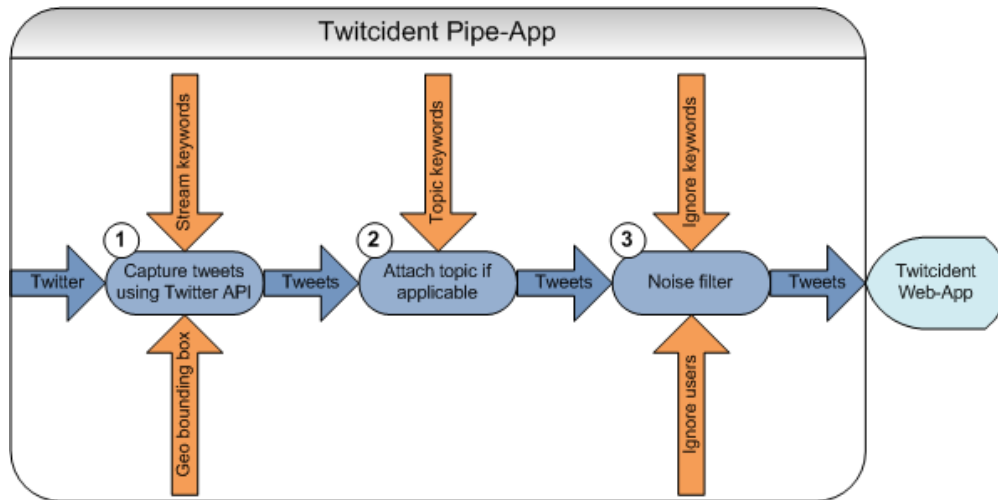


Figure 2.7: Twitcident Pipeline

Figure 2.7 illustrates its ‘pipeline’ flow. Incoming tweets are captured by the Twitcident Stream keywords and an optional geographical bounding boxes (Observation 7). In step 2, each tweet is matched against the topic keywords and if a match is found, the topic is attached to the tweet. Finally, in step 3 topics may be detached based on ignore terms or ignore users, and data is presented (through a middleware database).

2.2 An introductory case analysis: Project X, Haren

In the next few sections, we perform case studies and will make notice of relevant observations related to the data itself. For this we will use real world running Twitcident Instances, and prefer those that are recurring throughout the thesis. We will analyze some instances individually, but will also analyze generic properties empirically over a large corpus. Our main aim is to observe what indicates relevant content: what properties of a tweet potentially distinguishes relevant content from irrelevant content and vice versa? How can we define relevant? A secondary goal is to identify what exactly makes the problem difficult, and why our problem could differentiate from existing solutions.

For our initial analysis, we used the case “Project X, Haren”. Project X Haren was an event that started out as a public invitation to a birthday party by a girl on Facebook, but ended up as a gathering of thousands of youths causing riots on 21 September 2012 in the town of *Haren, Groningen*.

2.2.1 Background

On 6 September 2012, a 15 year old girl from Haren sent 78 friends a public invitation to her 16th birthday via the social network site Facebook. She deliberately chose the option ‘public’, so that her friends could bring other friends. This way, the girl hoped to obtain a head count. One of her friends’ friends misused the invitation and invited 500 people himself. Through the notorious snowball effect thousands of people

were added to that number in a matter of days. In just two days, 16.000 people were invited. The girl then deleted the event after consulting her parents. However, others took command and quickly Twitter and Facebook were filled with terms like Project X Merthe (the name of the girl), Project X Stationsweg (the street where the girl lived) and Project X Haren.

On 18 September, the municipality Haren and the local police force agreed to act strongly if the party would escalate. By then, the rumours had spread to the national media. 55.000 people had been invited, 6.000 of them accepted. Local entrepreneurs took advantage of the situation and organised afterparties in Groningen. Between 19 and 21 September, the expected attendance grew from 8.000 to 30.000. On the 21st alone, around 400.000 Tweets were sent regarding Project X Haren.¹



Figure 2.8: Project X

2.2.2 The case

This makes up for an interesting case for us to analyze, as it is a typical event that local authorities would like to monitor using a Social Media analysis tool like Twitcident. TNO, a research institute and partner of CrowdSense/Twitcident, analyzed this event afterwards, primarily by using a modified Twitcident application. They analyzed and annotated these tweets, manually classifying them as either:

Relevant Related to Project X Haren, and containing potentially valuable information.

Irrelevant Related to Project X Haren, but not containing potentially valueable information, for example sarcasm or jokes about it.

Noise Not related to Project X, for example when 'Project X' or 'Haren' are used in a different unrelated context.

Furthermore, they classified the annotating user as one of the following:

Eye witness (hard core), Eye witness (followers), Eye witness (potential),
 Eye witness, Present (inflammatory), Present, Not present (inflammatory),
 Not present, News media, Entertainment media, Authorities, Crisis pro-
 fessional, Entrepreneur, Artist, Doubtful

¹Source: http://en.wikipedia.org/wiki/Project_X_Haren

This case differs from our other case studies, as the data was collected after the event, instead of real-time. However, its setup does not differ significantly from other instances. Furthermore, this is our only annotated case where part of the data is manually analyzed (by TNO). Therefore, we consider this case as our preliminary case and entrypoint for our own further analysis.

2.2.3 Dataset properties

The provided dataset contains 740421 collected tweets from an interval between 17-09-2012 11:37AM and 26-09-2012 06:23AM, acquired by searching for keywords like ‘projectx’, ‘projectxharen’, ‘haren’, ... We have summarized this dataset by Table 2.1 and Figure 2.9, as well as which topics are configured with this instance.

Up front, we should notice this dataset that was provided for this research contains two biases. First of all, due to the database structure a tweet can only be assigned one topic. This means that if a tweet could be positively classified among multiple Twitcident Topics, only the latest (by program flow) classification topic is stored. Furthermore, the configuration was setup afterwards following the actual event, and therefore the monitor was setup with prior knowledge of what one would be looking for.

On the other hand, we observe no pattern in the distribution of which tweets are annotated: some topics have an annotation ratio as high as 10-11% of all topic classified tweets, while others have a ratio as low as 0.66%. Even 45 unclassified tweets have been annotated. Therefore, we assume annotation has been done in a rather unstructured way. Furthermore, we were given information that the annotations were not done by a single person, but by a group of people, with potentially different views on the data. Overall 0.76% of our dataset has been annotated.

However, yet with all this prior knowledge and biased configuration, only just 19% of the tweets is judged relevant! This illustrates both the need for a more effective solution as well as that we are dealing with low signal detection, as we will subsequently notice among our next datasets (Observation 9). We also notice that the proportion of irrelevant tweets with respect to the amount of noise is pretty high (Observation 10).

Although this dataset seems to be unreliable due to containing a few biases, it can very well serve as an initial entry-point to discover specific properties of our problem, primarily on the tweet content level itself. Therefore, we will proceed by analyzing and discussing a selection of topics within the next few subsections.

2.2.4 Topic analysis: Bekogeling (Bombardment)

We will first take a look at the topic *Bekogeling (Bombardment)*, as this topic discriminates itself with the highest noise ratio among all topics, and the lowest relevant ratio. To analyze a topic, we review the keywords associated with the topic, and the amount of relevant and irrelevant matches. Table 2.2 lists all keywords, ordered by the amount of relevant matches.

The intend of this topic is to identify if and where objects are thrown as bombardment, for example towards police or other crowds. To resolve this, we observe that the list of keywords is mainly composed of throwable objects like *hekken (fences)*, *fietsen (bikes)* and *blikken (cans)*. These are actually hyponyms of ‘a throwable’ or ‘bombardables’ (Observation 11). However, these terms can be used in a lot of other

Topic	#Tweets	#Annotated	Noise	Irrelevant	Relevant
Bekogeling (Bombardment)	6820 (0.92%)	383 (5.62%)	249 (65%)	118 (31%)	16 (4%)
Brand (Fire)	4044 (0.55%)	111 (2.74%)	23 (21%)	57 (51%)	31 (28%)
Communicatie (Communication)	1534 (0.21%)	168 (10.95%)	6 (4%)	138 (82%)	24 (14%)
Drugs (Drugs)	2303 (0.31%)	241 (10.46%)	13 (5%)	140 (58%)	88 (37%)
Drukte (Overcrowding)	11080 (1.50%)	827 (7.46%)	10 (1%)	647 (78%)	170 (21%)
Geweld (Violence)	96447 (13.00%)	1527 (1.58%)	201 (13%)	1134 (74%)	192 (13%)
Hulpdiensten (Public Services)	2294 (0.31%)	33 (1.44%)	1 (3%)	25 (76%)	7 (21%)
Opvallend (Striking)	1833 (0.25%)	55 (3.00%)	1 (2%)	46 (84%)	8 (15%)
Slachtoffers (Casualties)	5380 (0.73%)	108 (2.01%)	6 (6%)	85 (79%)	17 (16%)
Vernieling (Havoc)	16584 (2.24%)	110 (0.66%)	33 (30%)	62 (56%)	15 (14%)
Vervoer (Transport)	23500 (3.17%)	1892 (8.05%)	32 (2%)	1352 (71%)	508 (27%)
Vuurwerk (Fireworks)	2219 (0.30%)	164 (7.39%)	20 (12%)	115 (70%)	29 (18%)
Unclassified	566383 (76.49%)	45 (0.01%)	2 (5%)	32 (71%)	11 (24%)
Total	740421	5664 (0.76%)	597 (11%)	3951 (70%)	1116 (19%)

Table 2.1: Project X: General dataset properties

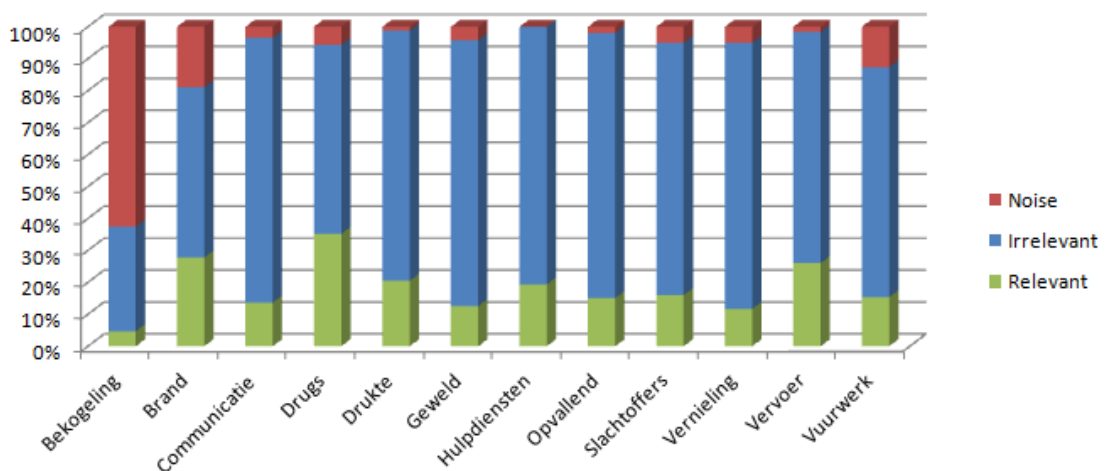


Figure 2.9: Project X: General dataset properties

Keyword	Count	Noise	Relevant	Irrelevant
hekken	76	18	5	53
dranghek	42	11	4	27
dranghekken	40	11	4	25
fiets	164	144	4	16
blik	14	6	2	6
blikken	3	1	2	0
fles	30	9	2	19
flessen	20	3	2	15
verkeersbord	23	11	2	10
verkeersborden	18	8	2	8
fietsen	48	43	1	4
steen	22	17	1	4
bierblik	1	0	0	1
bierblikje	1	0	0	1
bierblikjes	1	0	0	1
blikje	5	1	0	4
blikjes	3	0	0	3
glas	6	4	0	2
glazen	4		0	1
stenen	10	1	0	9
stoeptegels	1	1	0	0
No matches for: baksteen, bakstenen, bierblikken, paaltjes, staaf, staven, stoelen, stoeptegels, terrasmeubilair, trottoirband, trottoirbanden, trottoirtegels, trottoirtegels, verkeersmeubilair				

Table 2.2: Topic keywords for Project X: Bekogeling (Bombardment)

different contexts rather than only in the context of bombardment (Observation 12): most occurrences with ‘bike’ won’t be in the context of bombardment, as people might also be traveling to Haren ‘by bike’. However, extracting the context of a term can be difficult as illustrated by the most relevant examples, listed in Table 2.3: whereas the first case the context could be identified through the word *gooien* (*throwing*), it is much harder with the second example.

When we look at these tweets, we note that images can often provide additional relevant information like the offending person (Observation 13). We also note that another object that is being thrown is mentioned: *bloempotten* (*flowerpots*). However, this keyword is not present in the topic keywords. We note that it is hard to be comprehensive and manually think of all cases in advance (Observation 14).

Furthermore, we observe that some terms do not have matches, despite being potentially relevant. For example *bakstenen* (*bricks*) are often used in bombardments against police, but due to not occurring within our dataset, the relevancy of the term cannot be measured (Observation 15).

2.2.5 Topic analysis: Brand (Fires)

The second topic we will take a look at is *Brand* (*Fires*), as this topic has one of the highest relevance ratios, but still has a large proportion of noisy and irrelevant tweets. Furthermore, this is a typical topic that is used commonly among various public service

Zit nu in #haren gaat helemaal los hier echt te gek mensen gooien met fietsen bloempotten alles #gezellige zweer http://t.co/x9g6nSrS I am in #haren now totally going crazy here really crazy people are throwing bikes flowerpots everything #cozy atmosphere http://t.co/x9g6nSrS
Daar gaat weer een fiets #projectX #Haren #pa http://t.co/DUvhJsIN There goes another bike #projectX #Haren #pa http://t.co/DUvhJsIN



Table 2.3: Examples of two relevant tweets relating to Bombardment with bikes

related instances, and was the leading topic in preceding work [56]. Table 2.4 and Figure 2.10 lists all keywords, ordered by the amount of relevant matches.

First of all, we notice the main topic keyword is also the most frequent keyword, containing most relevant but also most irrelevant tweets. However, we also notice a few synonyms to it, some of which are slang, like *fik*, *fikkie* and *hens* (Observations 16 and 17), and the main topic keyword and its synonyms capture the biggest shares within a topic.

We also note that most keywords are observatory keywords that could identify fires: *rook* (*smoke*), *brandalarm* (*fire alarm*), *gas*, ... (Observation 18).

We see that the Twitcident topic keywords also include common misspellings: both *wordt niet goed* and *word niet goed* are defined, both *gaslucht* and *gaslugt*, both *misselijk* and *misselyk*, etc. (Observation 19)

Furthermore, for some keywords, we observe that both the singular and plural form are included (Observation 20). For English and some Dutch words this is easy (*brandstapel* and *brandstapels*, but Dutch also contains a lot of irregular plural forms², for example: *schip* vs *schepen*, *gelid* vs *gelederen*, *aanbod* vs *aanbiedingen*, ... In general, Dutch is a more complex language than English (Observation 21).

This also reminds us of the tense in which a tweet is written: in present or past tense. Most clients are only interested in current observations to act upon, and therefore past tense would be irrelevant. However, past tense may be relevant to other instances, for example police tracking information around a case (Observation 22).

²[http://nl.wikipedia.org/wiki/Meervoud_\(Nederlands\)#Onregelmatige_meervoudsvorming](http://nl.wikipedia.org/wiki/Meervoud_(Nederlands)#Onregelmatige_meervoudsvorming)

Keyword	Count	Noise	Relevant	Irrelevant
brand	38	8	11	19
fik	22	0	9	13
vuur	10	1	4	5
autobrand	4	0	2	2
fikkie	3	0	2	1
vuurtje	5	0	2	3
autobranden	1	0	1	0
rook	6	2	1	3
brandalarm	4	4	0	0
brandweer	3	0	0	3
brievibus	2	1	0	1
gas	10	4	0	6
hens	1	1	0	0
rookkolom	1	0	0	1
verbrand	3	2	0	1
verbranden	1	1	0	0
word niet goed	1	0	0	1

No matches for: brandstapel, brandstapels, brandweerauto, brandweerwagen, brandwond, brandwonden, brievenbussen, dampoe, dokken, explosie, frisse lucht, gaslucht, gaslugt, geen zuurstof, in de hens, krijg geen lucht, misselijk, misselyk, neergaan, onwel geworden, postbussen, rookontwikkeling, rookwolk, rookwolken, ruik gas, stank, stinkt hier, stookcontainer, uitgebrand, verbranding, vlam, vlammen, vreemde geur, vreugdevuren, vreugdevuur, vuurtje stoken, vuurton, vuurzee, vuutje, wolk, wordt niet goed

Table 2.4: Topic keywords for Project X: Brand (Fires)

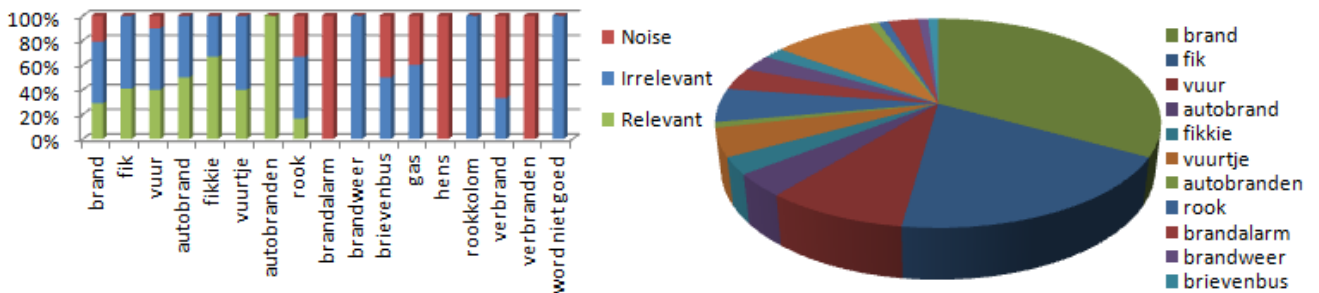


Figure 2.10: Topic keywords for Project X: Brand (Fires)

1	Net Haren voorbij gereden. Nog geen rook te zien xD <i>Just drove past Haren. No smoke to be seen yet xD</i>
2	Het loopt volledig uit de hand #autobrand #projectxharen ME gaat zo ingrijpen #wegwezen <i>It's totally getting out of hand #carfire #projectxharen Riot Police will act soon #getoutofhere</i>
3	Staat Haren al in brand? Of is het er gezellig? <i>Is Haren already on fire? Or is it friendly?</i>
4	Lekker hang ik met me haren in een kaars nu zijn ze dus verbrand :''''''''(<i>Great my hair got into candlefire en now they are burned :''''''''(</i>
5	ze hebben gewoon brand gemaakt op t gras hahaha #ProjectXHaren <i>they just made fire on a grassplot hahaha #ProjectXHaren</i>
6	Haren gestyld en 10.000 x verbrand.. <i>Hair styled and 10000 times burned..</i>
7	Ok'e, eerste dingen gaan nu is brand hier bij #projectxharen <i>OK, first things happening now there is fire here at #projectxharen</i>
8	Eerste brand gesticht. #projectxharen Dat wordt nog wat. http://t.co/4VpSjppz <i>First arson committed. #projectxharen That's gonna be something. http://t.co/4VpSjppz</i>
9	Mijn haren stijlen =vingers verbranden </3 <i>Styling my hairs =burning my fingers </3</i>
10	Na beroving van de supermarkt ontstaat er nu ook brand in haren: http://t.co/ljQk8XMD <i>After a supermarket robbery some fires are arising in haren: http://t.co/ljQk8XMD</i>

Table 2.5: Examples of interesting tweets matched by the keyword 'brand' (fire)



Figure 2.11: A joke about 'Haren' (also: 'Hair') on 'Fire'

Table 2.5 lists some interesting tweets matched by the keyword 'brand'. Tweets 1 and 3 illustrate tweets containing a joke or question, but are irrelevant as they contain no useful information. Tweets 4, 6 and 9 are irrelevant as the keyword 'haren' also has another meaning: 'hair' (Observation 23). Tweets 5, 7 and 8 are relevant, while Tweet 10 seems to be relevant at first. However, taking a closer look, the image linked is displayed in Figure 2.11: this tweet was obviously a joke. We also observe that some tweets contain emoticons, which *may* indicate a joke (Observations 24, 25).

2.3 Observing a real world instance: Nationale Politie Limburg

In this section we will discuss and analyze a real world running Twitcident instance, and prefer an instance that is recurring throughout the thesis: National Police Limburg.

National Police of the district Limburg is a typical public safety instance, interested in what is currently happening in their district so they can act upon it if required. Figure 2.12 and Table 2.6 display the monitor setup and keyword configuration.


 Alle 1425	Inbraak	Overvallen	Ram & Plofkraak	Geweld	Openbare orde	Vuurwapens	Brand
Venray Gennepe 124	2			35	19	23	45
Horst PeelMaas 32		1	1	1	7	1	19
Venlo Beesel 105	31	5		2	11	5	33
Weert 109	4	1		8	6	4	84
Roermond 28	9	2		6	2	6	3
Echt 27	1	1		4	2	11	8
Brunss Landgrf 17		2		1	1	1	12
Kerkrade 179	3			1	163		6
Heerlen 30		18		3	3	3	3

Figure 2.12: Twitcident Instance: National Police Limburg

Topic	Keywords
Inbraak (Burglary)	inbraak, inbraken, inbreker, inbrekers, ingebroken, breekt in, breek in, breken in, inbreken, woninginbraak, huisbraak, inbraakgolf, rijden weg auto, rijden weg scooter, weggereden auto, weggereden scooter, rijdt weg auto, rijdt weg scooter, reed weg auto, ...
Overvallen (Robbery)	berooft??, beroving, overval, overvallen, roofoverval, straatoverval, roofovervaller?, straatovervaller?, overvaller?, plundering??, geplunder?, plunder?, roverij, tasjesdief, tasjesdieven
Ram & Plofkraak (Ram- & Explosive raids)	plof? pinautomaat, plof? ATM, plof? geldautomaat, plof? pin, plof? ing, plof? abn, plof? rabobank, geploft? pinautomaat, geploft? ATM, geploft? geldautomaat, geploft? pin, geploft? ing, geploft? abn, geploft? rabobank, ontploft??? pinautomaat, ontploft??? ...
Geweld (Violence)	bedreig?, elkaar geslagen, bedreiging??, bek verbouw??, ga je nakken, ga je timmeren, geef? Klap???, hoofd afhakken, hoofd verbouwen, ik boek je, ik steek je, kom? Vechten, koudmaken, doodmaken, krijg? Tikken, krijg?? Wat verdien??, maak? ??m kapot, ...
Openbare orde (Public order)	wanorde, illegaal feest, illegale afterparty, actievoerder?, demonstratie, demonstrant??, demonstreren, gedemonstreer?, betoging??, manifestatie?, mileuactie, protest, protestactie, spandoek, spandoeken, voert actie?, voeren actie, actievoerder?, actie gevoerd, ...
Vuurwapens (Firearms)	*buks, *geweer, geweren, vuurwapen, ak47, blaffoe?, blaffer, pipa, pistool, pistolen, wapen?, revolver, schoten, geschoten, neergeschoten, doodgeschoten, schieten, vuurgevecht, schietgevecht, *wapenhandel, *wapenhandelaar, *wapenhandelaren, ...
Brand (Fire)	brandstichting??, brand, brandje, branden, fikkie, rookontwikkeling, brandalarm, explosie*, fik, fikkie?, pyromaan, piromaan, pyro, piro, ontploffing, rookmelder, uitgebran?, veenbrand??, bosbrand??, verbran??, rookwolk, vlam*, vuutje, vuurtje, gaslek, gaslekkage, ...

Table 2.6: Twitcident Instance: National Police Limburg

If we look at some tweets that have been classified as burglary (Table 2.7), we observe that Tweet 1 is matched based on the keyword combination ‘breek in’ but

these words are not correlated and therefore not relevant. Some word combinations are related as belonging to the same verb in which the order may be relevant (Observation 26). Furthermore, Tweet 2 and 3 are tweets from news media which are not relevant to this instance as they do not provide new real-time information (Observation 27).

Table 2.8 contains some interesting examples of irrelevant tweets related to the topic ‘schieten’ (‘Shooting’). In Tweet 1, we observe that compound forms of terms can have a significantly different meaning, and therefore introduce noise (Observation 29). Tweet 2 is obviously related to soccer, and is another example in which our term occurs in a different context than the intended context (Observation 12). Tweets 3–6 each contain our primary term within an saying or expression, which reminds us that sayings, expressions and pronouns are likely to yield noise (Observation 30).

Sometimes, a term can also be used as an intensification of another term, yielding the intensifying term to cause noise (Observation 31). Consider for example the search term ‘dood’ (‘death’) and its usages as an intensifier: ‘dood lachen’, ‘dood vervelen’, ‘doodsbang’, ‘doodeng’, ‘doodzonde’, ‘dood schrikken’, ...

Although we have made observations related to a specific Twitcident Instance in this section, we can state that these observations are not instance-dependent, and also present themselves among other instances like *Nationale Spoorwegen* and *Gaswinning Groningen*. We chose not to elaborate on those instances, since there are few to none new observations to be made with respect to the presented instance. However, there may be some very distinctive aspects to an instance with respect to defining relevancy. For example, most instances are not interested in tweets from news media, while others actually primarily focus on Tweets by news media.

1	“@blvckyeetus_: @_Shannelle K was toen kk dun nu kan ik niet eens in een boom klimmen ??” - nu breek ik sws me tand :s
2	14 jarige inbreker aangehouden http://t.co/XXbOrc958v # limburg # provincie # maastricht
3	Drastische stijging auto-inbraken in Roermond http://t.co/8tRlngCagI

Table 2.7: Examples of noteworthy tweets classified as ‘Burglary’

1	Gelukkig is de Sint al aangekomen in Groningen...hij mag wel <i>opschieten</i> want om 14.00 uur verwachten wij hem in Achterveld
2	Kostic laat na om Groningen op voorsprong te <i>schieten</i> . Hij raakt vanuit een moeilijke hoek de buitenkant van de paal. #grozwo
3	Niets op Twitter, niets op tv, niets op Facebook of whatsapp, geen geld en niets te roken. Wie <i>schiet</i> mij neer?
4	<i>Schiet</i> mij maar lek
5	Jongens file <i>schiet</i> me neer
6	Kromowidjojo laat sportgala <i>schieten</i> voor zwemtweestrijd http://t.comUD6s5To6k #grunnen #info #groningen

Table 2.8: Examples of noteworthy tweets classified as ‘Shooting’

2.4 General observations

We compiled an aggregated database of some Twitcident Instances to create one large common dataset. This aggregated dataset include 6.951.693 tweets and 3.003.954 unique corresponding users, from an interval August 1st, 2013 to June 30th, 2014. From this dataset we can deduct several instance-independent statistics based on some additional fields Twitter provides.

If we take a look at the geo-availability, we observe that only 3.55% has attached their geo-location (Observation 32), while 4.23% of the Dutch language tagged tweets has an attached geo-location. We also observe that 10.3% of the tweets is a reply, according to Twitter (Observation 33).

Furthermore, 40.2% of the tweets has the 'lang' field of the tweet set to 'nl' (which is heuristically determined by Twitter), while the actual percentage of Dutch tweets is significantly higher (Observation 34). However, language detection is relevant, since a keyword can occur in multiple languages with different meanings. Consider for example the keyword 'brand', which means 'fire', but can be used in the context of 'a popular brand', 'brand new' or 'brand-manager' in English.

2.5 Challenges and difficulties

In the previous sections we analyzed the problem on a tweet level basis, and with that answered the question "What properties or aspects of tweets (could) cause noise?". The final question that we would answer in this Chapter was "What are the specific problem properties and constraints imposed by the needs that Twitcident tends to fulfill?".

In this section, we will list some additional aspects of the problem that make it particularly different and which define the differentiating uniqueness of our problem. These are all derived from the needs that Twitcident tends to fulfill and its target domain (Twitter), and were encountered during our search for a solution.

Real-time The application operates real-time, and therefore data needs to be classified real-time, tweet by tweet. We can hardly calculate over datasets or a corpus (groups of tweets), can hardly search for content (content just streams in) and need to judge each tweets relevancy individually. This means it will be hard to make use of Information Retrieval techniques. Also, if we want to involve historical data or other processed information, we have to maintain a state. Furthermore, processing speed is important, as our artifact should be able to process faster than the input streams in. This also makes it harder to invest time on the setup-phase of a system, like training a potential machine learning system.

Limited setup time Often, especially for events, a Twitcident Instance only has limited setup time and in some cases is required to be operational within just hours, limiting the time available for configuration or potential machine learning.

Difficult domain We serve a specific, difficult domain with a difficult goal. Currently, even for Twitcident Operators themselves it is hard to manually judge whether a tweet is relevant or not, and they are considered the domain experts/oracles.

Even more difficult is the determination of what the Twitcident Client actually wants: it is hard to know a priori what we want or need to present. We are looking for events and tweets that may be relevant to them, but we do not specifically know what to search for in advance; which relates to the real-time aspect. This makes it also harder to pre-train a machine learning system as we do not possess training corpora.

Instance variety Each Twitcident Client has its own specific interests and needs for their Twitcident Instance, and generally monitor a specific domain with no or little correlation to other Twitcident Instances. For example, news articles may be considered noise to one instance while it is the main source of information for another. Or sentiment analysis, which may be irrelevant to one while it has a special Twitcident Topic in another. Furthermore, each Twitcident Instance has its own set of Twitcident Topics, making it harder to establish training corpora.

Low signal detection Only an extremely small fraction of the tweets is relevant (Observation 9), and they can hardly be found if one were to be looking for them: this actually pretty much is the problem itself. This makes it very hard to collect training data for machine learning systems, and also due to the low relevant/irrelevant ratio. Furthermore, recall is valued much higher than precision since we do want to miss a potential relevant tweet.

Geo availability Only a limited amount of tweets contain a geo-location (Observation 32) which make it unreliable to merely collect tweets based on location, since we do not want to miss potential relevant tweets without a geo location. Therefore, we need to rely on other ways to collect our data stream.

Informal content Social Media contain informal content and do not apply punctuation or other formalisms very well (for example: for some tweets it is very hard to distinguish sentence boundaries), which make it harder to apply linguistic analysis. Also, slang and street language should be taken into account.

Error-prone content Social Media content regularly contains typos and spelling mistakes, due to informality as well as the pace with which they want or need to be presented.

Short content Tweets have a limited short length, which increases the difficulty of analyzing and judging them. For example, linguistic analysis is much harder due to a lack of context. When specific words have multiple meanings, it is harder to extract the actual intended meaning. Furthermore, sarcasm and sentiment are much harder to extract, again due the limited context.

Language Twitcident primarily serves Dutch organizations and therefore operates on the Dutch language. Most scientific state of the art research is done for English, and therefore most data sources, datasets and corpora are English. For Dutch or other languages, this availability is significantly lower.

2.6 Summary: List of observations

At the start of this chapter we casted three research questions to analyze our problem:

1. What characterizes Twitcident and what implications does that have?
2. What properties or aspects of tweets (could) cause noise?
3. What are the specific problem properties and constraints imposed by the needs that Twitcident tends to fulfill?

In the previous we have addressed the third question and identified what aspects make our problem unique, challenging and hard to solve. Furthermore, throughout this chapter we have observed several phenomena and properties of tweets that we expect to be relevant to our problem. We conclude this chapter with an overview of all observations answering the other two research questions:

Observation 1. In the Twitcident user interface, a feedback mechanism is present: Tweets can be flagged as marked (relevant?) or noise.

Observation 2. Twitcident Topic classification is currently solely done by matching on predefined static keywords.

Observation 3. The matching mechanism of Twitcident Topic classification currently supports wildcards. Searching for combination of words is possible, although the order is considered insignificant.

Observation 4. Ignore terms are always used in conjunction with regular keywords, and therefore serve as a post-classification filter.

Observation 5. A tweet can be irrelevant based on the posting user account.

Observation 6. Users are able to, and used to, alter (add, remove, modify) keyword, ignore term and ignore user lists. Therefore, although these lists consist of static predefined words, it can be dynamically altered during application usage with real-time impact.

Observation 7. Tweets are captured from a stream by keywords and optional geographical bounding boxes.

Observation 8. Each Twitcident Instance is currently manually set up and preconfigured by manually defining settings, streaming keywords, topic keywords, ignore terms and ignore users; by CrowdSense employees.

Observation 9. The problem involves low signal detection: Only an extremely small fraction of a very large dataset is relevant.

Observation 10. The proportion of irrelevant tweets with respect to the amount of noise is high: 70% *irrelevant* vs 11% *noise*, for the Project X case. This phenomenon can also be observed in all other Twitcident instances.

Observation 11. Some relevant keyword lists are composed of hyponyms of the main topic keyword.

Observation 12. The context in which a word is used is relevant. For example, when searching for bombardments with thrown ‘bike(s)’, simply searching for the keyword ‘bike’ introduces a lot of noise. One should look for the object in the context of ‘throwing’, ‘tossing’, ‘bombarding’, . . .

Observation 13. Attached images can often improve the relevance of a tweet, and provide additional information.

Observation 14. It is hard to be comprehensive when constructing a list of (potentially) relevant search keywords, and manually think of all scenarios in advance.

Observation 15. Some terms may be relevant, despite not occurring in collected datasets, and therefore the relevancy to the intended context of a search term can not always be measured.

Observation 16. Regarding keywords: *synonyms* often are relevant and will decrease the chance of missing relevant tweets.

Observation 17. Within tweets, some people use quite a bit of *street language or slang*.

Observation 18. Human *observations* are relevant: see, hear, smell, notice, etc. . .

Observation 19. (Key)words in tweets are easily *misspelled* or contain *typos*, but should be captured nonetheless.

Observation 20. (Key)words can be written in *singular* or *plural* form, both of which could be relevant.

Observation 21. The problem involves *language specific* characteristics. Twitcident mainly serves Dutch-focused instances.

Observation 22. Tweets can be written in present *tense*, past tense or no tense. Mostly present tense is relevant, but past tense may be relevant to some instances too.

Observation 23. A word can have different *meanings*, based on its context. Often only one of these definitions is intended to be captured, while the others introduce noise.

Observation 24. While some issues are serious, there are people that make *jokes* about it which introduce noise. Some of these jokes can be very hard to identify from the textual content alone. The hardest jokes constitute *sarcasm*.

Observation 25. Tweets can contain textual *emoticon* representations, which might (but not necessarily) indicate a joke.

Observation 26. For some keyword combinations, the *order* in which they occur may be relevant.

Observation 27. Tweets from *news media* may not be relevant. These often contain an URL to the news article.

Observation 28. Tweets that contain a *question* often do not provide relevant information to most instances. However, a tweet can contain *multiple sentences*, of which at least one may provide relevant information.

Observation 29. *Compound* forms of terms may introduce noise.

Observation 30. Terms can occur within *sayings, expressions and pronouns*, which are very likely to yield noise.

Observation 31. Keywords may be used as *intensifications* to other words, which could render them noisy.

Observation 32. Only a fraction (3.55% in our aggregated sample) of the tweets have a *geo-location* attached. This observation is also supported by Olteanu et al. [45].

Observation 33. In our aggregated sample, 10.3% of the tweets is a *reply* according to Twitter.

Observation 34. According to Twitter's heuristic *language* determination 40.2% of the tweets in our aggregated sample is Dutch, while the actual fraction is a lot higher. Nonetheless, identifying the right target language is relevant.

Chapter 3

Research

In Chapter 2 we explored where the noise comes from and what constraints are imposed by our specific problem. With this in mind, we can now perform rigorous research and start the search for a fitting solution.

First, we want to elaborate on the research methodology that we applied. Therefore, we start this chapter by answering the following question in Section 3.1:

What research methodology are we going to apply to solve our problem?

Next, we should check existing literature for related work and existing solutions to parts of our problem and answer the next research question:

What related work has been performed? How does this work relate to our problem? What are the relevant contributions that could be applicable to our problem?

We will also explore and state what methodologies are suitable to solve our problem and address the final question for this chapter in Section 3.3:

What scientific methodologies and technologies are suitable to (partially) solve our problem?

3.1 Methodology: Design Science Research

Design Science Research is a methodology based on the understanding of a problem and focusing on utility, rather than behavior. Design-Science strives to expand the borders of the human and organizational capacities by creating new, innovative and meaningful artifacts [25]. It is essential that the artifact is new and has utility. After all, the artifact must be a solution for a yet unsolved problem, or a more effective and innovative solution for an already solved problem. March and Smith [38] identify two processes: Building and Evaluating.

The process wherein the artifact is being created includes a search process wherein the problem is analyzed and the artifact is being constructed. The creation of the artifact should be based on existing theories. These theories are being applied on the research and tested, adapted and extended by the experience, creativity and problem-solving capacities of the researcher. The artifact itself should be rigorously defined, formally presented, coherent and internally consistent.

The evaluation of the artifact then provides feedback information and a better understanding of the problem in order to improve both the quality of the product and the design process. This build-and-evaluate loop is usually iterated a number of times before the final design artifact is generated [25].

Artifacts created by means of Design Science Research are hardly ever full fledged information systems. Rather, the artifacts are mostly just ideas and technical possibilities by which implementation, design and the use of information systems can be efficiently achieved [16].

Design Science Research addresses those problems characterized by [25]:

- unstable requirements and constraints based upon ill-defined environmental contexts,
- complex interactions among sub-components of the problem and its solution,
- a critical dependence upon human cognitive abilities (e.g., creativity) to produce effective solutions

In Chapter 2 we have seen that our problem satisfies these characterizations, and therefore will adopt this methodology. Hevner et al. [25] created seven guidelines reflecting the most fundamental principles of the Design Science Research process. Table 3.1 lists these guidelines, along with a short discussion how our research design meets them.

Guideline	Discussion
1. <i>Design as an Artifact</i> : Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.	In Chapter 4 we provide our solution for the problem. Here we describe the artifact we produced to solve the problem in detail.
2. <i>Problem Relevance</i> : The objective of design-science research is to develop technology-based solutions to important and relevant business problems.	In Chapter 1 we introduced our problem. Our artifact will contribute to the quality of information presented by Twitcident, which aids Twitcident Clients in their operational processes and services.
3. <i>Design Evaluation</i> : The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.	In Chapter 5 we demonstrate the performance of our artifact and evaluate it. We describe rigorously how we obtained and processed our datasets, and how we executed the experiments and evaluation.
4. <i>Research Contributions</i> : Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.	In Chapter 6 we conclude with a discussion of our work and reflect upon our contributions. We also discuss areas for improvement and future research to complete the research cycle.
5. <i>Research rigor</i> : Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.	In the remainder of this chapter we discuss relevant literature and related work. Furthermore, our research relies on the Design Science research methodology and we employ existing knowledge (related work) as a foundation for our artifact and its evaluation.
6. <i>Design as a search process</i> : The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.	In Chapter 3 we search for utilizable technologies. In Chapter 5 we will search for an effective configuration of our artifact. Based on the results and insights of our evaluation, we will also deduct new goals and areas for improvement and future research.
7. <i>Communication of research</i> : Communication of Design-science research must be presented effectively both Research to technology-oriented as well as management-oriented audiences.	Within this thesis it is assumed the reader is familiar with social-media, especially Twitter. For the discussion of our solution we also assume a background in Computer Science. However, if our presented information would be relevant for more management-oriented audiences, these findings will be communicated appropriately (Chapters 1, 2 and 6).

Table 3.1: Design Science Research guidelines

3.2 Related work

In this section we will discuss related work. We will discuss the papers most resembling our problem in detail in the following sections, where for each paper we will answer the following questions using a consistent format: What? Why? How? Conclusions? Relation to our work? What technologies or conclusions our useful with respect to our problem?

After the most resembling publications have been discussed, we will briefly discuss other relevant related work in Section 3.2.6. Other research areas of interest are then discussed in Section 3.3.

3.2.1 TwitterStand: News in Tweets

In [50] Sankaranarayanan et al. introduce a news processing system based on tweets, called TwitterStand. Their aim is to capture tweets that correspond to late breaking news, without relying on other sources. Their primary difficulties are noise within tweet sets, and that contributors/reporters are not known in advance while there may be many of them. They address issues like noise removal, determining tweet clusters of interest and determining relevant locations associated with the tweet clusters; all constrained by that methods must be online (not all data available at the start, but gradually arriving).

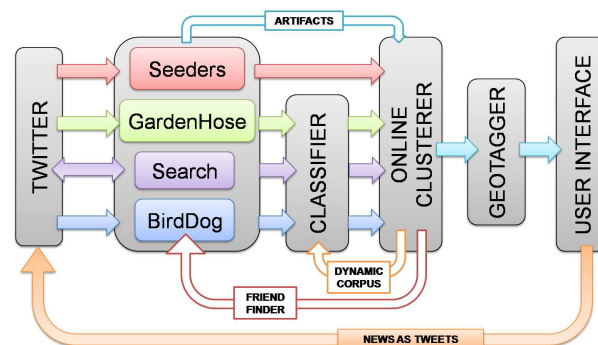


Figure 3.1: Related work: TwitterStand system architecture

Figure 3.1 depicts the system architecture of TwitterStand. First, they gather tweets from several sources. Seeders are a handpicked set of 2000 news accounts known to publish news. GardenHose publishes a lot of tweets about diverse subjects. BirdDog enables the authors to follow a set of up to 200,000 selected users: for this set they select the followers of the Seeders set, based on the assumption that these are most likely to tweet about the news.

Next, they classify tweets as either junk or news as to remove almost all noise (at the cost of potentially losing news). They use a Naive Bayes classifier that is pre-trained on an annotated corpus. This classifier operates on the set of words a tweet is composed of, although more specifics are missing.

After classification, they use an online clustering algorithm (leader-follower clustering) to detect topics and cluster tweets according to these topics, taking both the

feature vector (weighted keywords using TF-IDF) as well as the tweet time into account.

They deal with noise based on the assumption that if you define quality clusters, they will be relatively noise-free. Therefore, they only allow tweets from reputable sources (Seeders) to form clusters.

They proceed with attaching geo-locations to each clustered tweet, based on two methods. The first is trivially done by looking up a possibly attached geo-location field of a tweet. The second method is toponym recognition and resolution of tweet texts. The latter technique is partly performed with the use of two Natural Language Processing (NLP) techniques: Part-of-Speech (PoS) tagging and Named Entity Resolution (NER).

They conclude their paper by remarking that their system can easily be generalized and applied to other concepts.

TwitterStand's aim is similar to Twitcident's: gathering news from tweets real-time, although Twitcident has both a broader and narrower scope: broader in the sense it can address any topic that not necessarily is news, but narrower in the sense that it addresses a single topic of interest. The goals of this paper are also very similar to ours, especially noise removal. Furthermore, they also focus on the same real-time problem, limiting the scope to online methods, making their contribution relevant.

However, we note that nearly all parts of the system depend on the initial Seeders set: the users to follow depend on this set, and the clustering quality completely relies on this set. This set is pretty appropriate for news related content since it is a set of news-accounts, but our application primarily aims at exploiting the crowd outside of media. If none of the news-accounts tweets about a subject, it will not receive a cluster and therefore not presented. Furthermore, their classifier is pre-trained on an available news dataset, which we cannot apply due to our varying topics.

On the other hand, we can identify several techniques we could employ in our solution: Classifiers (Naive Bayes being an instance of it), Clustering, Part-of-Speech tagging and Geo-tagging are all viable (sub-)technologies we could use in our exploration.

3.2.2 OMG Earthquake! Can Twitter Improve Earthquake Response?

In [18] Earle et al. explore the use of Twitter in detecting earthquakes as early as possible, as they observed that in some case the first tweet about it was posted in 19 seconds. They search for all geo-coded tweets containing the word 'earthquake' and plot the tweet frequency against time. They do note that the limitation to only geo-coded tweets reduces the amount of tweets by 50%. After evaluating this approach on various occasions of earthquakes, they observe mixed results. They observed their approach had a lot of noise on some occasions due to the word 'earthquake' being used in different contexts, a very limited availability of tweets in some areas and in some cases missing most relevant tweets due to another phrase being used more often ('Did you feel that?!').

First of all, this paper hits the mark with their topic, since earthquake detection is one of our example instances. They limit their scope to geo-coded tweets, and state this reduces the captured amount by 50%. By Observation 32 we have seen that in our cases this reduction is even higher, up to as high a reduction as 96.45%, which

is confirmed by some cases in their evaluation making this approach not viable for us. However, we do receive confirmation of earlier observations regarding multiple contexts and other forms of phrasing.

3.2.3 Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors

Sakaki et al. [49] construct a system for real-time event detection, and use earthquake detection as an application. To identify earthquake related tweets, they setup a Support Vector Machine (SVM) binary classifier based on three types of features: statistical features (number of words, position of keyword), keyword features and word context features (the words directly preceding and succeeding a keyword). They then construct a temporal model to detect earthquakes by monitoring the tweet distribution. Observing that tweets tend to form an exponential distribution in case of an event, they statistically determine the threshold to trigger an alarm for the specified event. They also construct a spatial model using Kalman and particle filtering to estimate the event location. After evaluation, they report that their system detected 96% of the earthquakes in Japan and that they are reported faster than traditional reporting methods.

This work is very similar to ours, in that it aims at real-time identifying events related to a given topic. Also, their illustrative case ‘earthquake detection’ is a real-world Twitcident instance too. However, their work primarily aims at *detecting* a given event based on a *volume* of collected data, but does not care about the *quality* of the collected data itself. Our aim is to improve the quality of the collected data. Furthermore, their work only applies to large-scale events a lot of people will Twitter about (as was one of their assumptions) whereas we should also support low signal events.

3.2.4 Weak Signal Detection on Twitter Datasets

In [54] Song researches the detection of weak signals within Twitter data. In contrast to most research done on event detection that relies on volumes of data to cluster, this work is about collecting all (little) information about a topic. Given a topic, its configuration and a pre-labeled dataset his system classifies tweets as ‘relevant’, i.e. belonging to the given topic, or ‘irrelevant’. This is done by a two phase process: a ‘simple filter’ succeeded by an ‘advanced filter’. The first one performs normalization, keyword filtering and tweet structure filtering, while the second filter consists of a classifier with multiple types of features: Twitter function related, syntactic, semantic and sentiment features. The advanced filter needs to be trained with the use of a pre-labeled dataset for each topic.

Song evaluates multiple facets of his work. First he evaluates the performance of his system with all features active, and achieves satisfactory results. Second, he evaluates the impact of each feature-category on the performance by evaluating all subsets of the four types, and concludes that the syntactic features set a baseline that is improved with each category of features added. Finally, Song compares his approach to the traditional approach.

This work very much resembles our work, in that we are trying to extract a weak signal from a given topic, with a few differences: our problem is more specific (more

constrained), and therefore his solution is not directly applicable to our problem. For example, the real-time aspect has not been taken into account in his work.

Although the results presented are very satisfactory, these were obtained from a single relatively simple target domain, and may not be as good on our difficult domain instances. However, due to the few differences, we can implement parts of his contribution in our work. We can take the defined features into account, as well as the results about using multiple types of features. Furthermore, we can also consider his method of feature selection.

3.2.5 CrisisLex: A Lexicon for Collecting and Filtering Microblogged Communications in Crises

In [45] Olteanu et al. aim at locating tweets that contain crisis-relevant information during mass emergency situations. Their goals are to improve query methods, and return more relevant results than is possible using conventional manually-edited keywords or location-based searches. They approach the problem from a perspective to improve on the recall without significant loss of precision.

To achieve this they construct a generalized crisis lexicon of terms that frequently occur during various crisis situations, and try to automatically identify terms using pseudo-relevance feedback mechanisms. The lexicon is constructed through several phases. During the first phase, candidate terms are selected from the complete dataset (sampled based on rough keywords and/or location; equivalent to our input Twitcident Stream). Candidate terms are all word uni-grams and bi-grams. They remove mentions, URLs, terms shorter than two characters, terms longer than 16 characters and punctuation. The resulting set is stemmed using Porter's stemmer and then filtered: only entries that occur in at least 0.5% of the dataset are kept. This is followed by a scoring phase using two statistical tests (chi-squared and point-wise mutual information). The third phase consists of several 'curation' steps: they remove names and identify the crisis-relevancy of each word. Both these phases involve CrowdSourcing. The resulting crisis-independent lexicon is obtained by selecting the top K scored tweets.

For each crisis, the lexicon is adapted to the specific crisis by employing pseudo-relevance feedback (PRF): during the first hours of a crisis, tweets are sampled using the generalized lexicon. The most frequent occurring word unigrams and bigrams not already in the lexicon are added to it. Because hashtags may be widely adopted and thus increasing recall, they also apply this process to hastags.

They experiment with varying scoring strategies and varying curation steps and notice a clear trade-off between precision and recall with the varying configurations. They also experiment with varying PRF-configurations, and notice a boost in recall while precision remains stable. Their results are satisfying, with F-scores up to 0.6 and a recall up to 0.7.

The problem described in this paper is part of our problem too: creating an accurate collection of words to specify a topic significantly affects the noise level, as we have seen in Chapter 2. A well built lexicon could be extremely useful. However, we observe a few problems of their approach if we were to apply it to our problem:

- They employ CrowdSourcing for every new instance. This requires the topic

and goal to be public, which is not suitable to every Twitcident Instance due to confidentiality. We have also seen that even domain experts have difficulties judging some tweets: only clients themselves accurately know what is relevant and what is not. Furthermore, employing CrowdSourcing makes the setup process dependent of other people which might be a problem: in urgent cases an instance should be constructed very fast at any given time.

- Olteanu et al. measure the performance of their work using two joined classes (directly and indirectly related tweets) as their positive class. However, indirectly related tweets are often considered noise with respect to our problem. Although we could modify this part, we expect the precision to drop. In our opinion, further research on this part should first be performed to consider this option.
- In our current terminology, we could state that their work focuses more upon creating and building an effective Twitcident Stream, rather than effectively classifying for a specific Twitcident Topic which is our aim.

An interesting contribution presented in this paper is the observation that the pseudo-relevant feedback (PRF) mechanisms increase recall without significant loss of precision, which is something we should really consider within our work.

3.2.6 Other related work

Stronkman's thesis work [56] forms the foundation for our work as it introduces Twitcident, followed by the publication of additional papers [1, 2]. These all describe a prior design of Twitcident that detects incidents using an emergency broadcast service, aggregates data from social media (Twitter) to it, semantically enriches this using Named Entity Recognition (NER) and then filters out noise. We should point out that the current version of Twitcident mainly focuses upon detecting relevant 'incidents' of a predefined topic, and effectively presenting the results for analysis.

Java et al. [26] explore and observe why people use Twitter, and how they adopt it. It is useful for understanding the motivations of people using Twitter. Vieweg et al. [59] analyze what Twitter may contribute to situational awareness during natural hazardous events.

Becker et al. [6] explore a method for identifying real-world events by clustering the stream of tweets, which may be useful for defining our Twitcident Stream. Walther and Kaisser [61] detect geo-spatial events through a clustering approach, focussing on geo-spatial properties by monitoring a specific area. They use a multiple types of features for clustering, including word overlap, sentiment features, subjectivity, tense and semantic categorization. Packer et al. [46] approach the problem from a different angle, applying semantic query expansion.

Li et al. [32] present TEDAS to detect news events, analyze their spatial and temporal patterns and identify the importance of the events. Interesting about their work is that they attach a location to each tweet, by first sequentially looking at the geo-tag of the tweet, location entities within the tweet text and geo-location of the user. If none exist, they try to predict the users location by analyzing the tweet history and friends of the user.

Culotta [14] explores multiple methods for detecting influenza epidemics by analyzing Twitter messages, and attempt to model influenza rates using regression models. However, they focus upon estimating the right quantities and epidemic intensities, rather than presenting accurate substantial content. Lampos et al. [30] aim to solve the same problem, by using a bootstrapped version of Lasso (Bolasso) to extract a set of words to use as features and calculating a daily flu-score.

Mathioudakis and Koudas [41] present a system that identifies emerging topics (trends) on Twitter in real-time, by detecting and analyzing ‘bursty keywords’. We could take such keywords into account and analyze its effects on the relevancy of a tweet.

Chen et al. [11] present CrowdE: a filtering system to collect user opinions about brands, for direct customer engagements. They apply a machine learning classifier trained using crowd-sourcing, and separate two levels of classification: tweets are first classified on relevance to the brand, then by opinion.

3.3 Technologies

In this section, we will list technologies that may be useful to our work.

Machine Learning During our literature research, we often encountered the use of Machine Learning classifiers. Machine learning explores the construction and study of algorithms that can learn from and make predictions on data. Such algorithms operate by building a model from example inputs in order to make data-driven predictions or decisions rather than following strictly static program instructions.

There are generally three subtypes of Machine Learning: classification, regression and clustering. Classification and regression are supervised learning problems [28]: the computer is presented with example inputs and their desired outputs (the training set), and the goal is to learn a general rule that maps inputs (the features) to outputs. For classification, the output consists of one or more nominal classes which can be chosen from. For regression, the output is a numerical value. Types of classifiers include Naive Bayes networks, Support Vector Machines (SVM), Neural Networks, tree-based classifiers, regression models and rule-based classifiers.

Clustering usually is an unsupervised problem, i.e: the outputs of the training data are not known in advance. The goal here is that a set of inputs is to be divided into groups (clusters), that are to be determined by the Machine Learner. Since in our problem the topics are defined in advance, we are not interested in clustering, but rather in the other two types.

In Chapter 5 we will observe that three classifiers perform best on our problem: REPTree, RIPPER and Random Forest. We will briefly discuss these a little further.

Reduced Error Pruning Tree (REPTree) [55] is a fast decision tree learner which builds a decision/regression tree using information gain as the splitting criterion, and prunes it using reduced error pruning. REPTree uses the regression tree

logic and creates multiple trees in different iterations. After that it selects best one from all generated trees as the *representative*. The tree is then pruned based on the mean squared error on the predictions made by the tree.

Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [13] was proposed as an optimized version of IREP [21]. It is based in association rules with reduced error pruning (REP), a common and effective technique found in decision tree algorithms. IREP is used to determine an initial rule set. This rule set is then simplified and optimized using a rule optimization heuristic, after which new rules are added that cover remaining positive examples (again using IREP). This process is repeated a number of times. At each stage of simplification, the pruning operator chosen is the one that yields the greatest reduction of error on the pruning set. Simplification ends when applying any pruning operator would increase error on the pruning set.

Random Forests [7] are an ensemble learning method that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random forests correct for decision trees' habit of overfitting to their training set. The method combines the idea of 'Bagging' and the random selection of features in order to construct a collection of decision trees with controlled variance.

Natural Language Processing Natural Language Processing (NLP) [37] is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (natural) languages. It is a research field to make computers able to process and understand natural language in the form of text or speech. The research is based on many different disciplines such as computer sciences, linguistics, mathematics, electrical engineering and even psychology.

Some areas within NLP are useful to our work. Named Entity Recognition (NER) [43] determines, given a stream of text, which items in the text map to proper names, such as people or places, and what the type of each such name is (for example, a person, a location or an organization). Part of Speech (PoS) [22, 39, 60] determines, given a sentence, the part of speech for each word, like 'noun', 'verb' or 'adverb'. Note that this is not a trivial task, since some words can be of multiple parts of speech, like 'book': it can be the object book, or used as verb in to book a flight. English as well as Dutch have many such ambiguities. Sentence breaking is involved with the separation of whole sentences, while tokenization is involved with the identification of boundaries between words. Note that names of entities can consist of multiple words but should be regarded as a single token.

Sentiment and sarcasm detection Sentiment analysis [3, 5, 27, 34, 42, 47, 58] aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document. The attitude may be his or her judgment, affective state or the intended emotional communication. Sarcasm detection [23, 33] takes this one step further, by identifying sarcasm. These

are extremely hard and hot research areas, as even the US secret service issued requests for such software ¹.

String Similarity metrics A string similarity metric is a metric that measures distance (opposite of similarity) between two text strings for approximate string matching. Most of these algorithms are closely related to sequence alignment in bio-informatics. The most widely known string metric is the Levenshtein Distance [31], returning a number equivalent to the number of substitutions and deletions needed in order to transform one input string into another, known as an edit distance. The Damerau-Levenshtein [8, 15] distance also allows transposition operations. While these algorithms focus on minimizing the edit distance, the Needleman-Wunsch [44] algorithm focuses on maximizing similarity which turns out to be equivalent. Whereas the Needleman-Wunsch algorithm translates to global sequence alignment, the Smith-Waterman [53] algorithm performs local sequence alignment. Dice's coefficient [17] is a set similarity metric, which can be applied on strings by splitting them in sets of n-grams.

Most of these metrics and algorithms calculate the similarity or distance between two strings, with a length difference also imposing a penalty. However, when we want to know if a smaller string (i.e. a keyword) occurs misspelled within a larger string (i.e. a tweet) we need to apply string approximation algorithms for variable length, or approximate substring matching. A brute-force approach would be to compute the edit distance (for example by applying Levenshtein) to the smaller string for all substrings of the larger string, and then choose the substring with the minimum distance. However, this algorithm would have asymptotic running time $O(n^3m)$. Sellers [51] improves on this by applying dynamic programming (DP) while introducing a parameter k representing the maximum amount of mismatches. Other techniques involve indexing. However, as we are not interested in the exact edit distance but rather in a lightweight fast-performing algorithm measuring similarity, we have developed our own algorithm by adapting Dice's similarity coefficient. This algorithm is presented in Appendix B.

Performance metrics Many performance metrics exist to evaluate the performance of a system. For comparing two binary classification outputs, of which one is often considered the ground truth and the other the evaluated classification, most performance metrics rely on a confusion matrix of true/false positives/negatives. Considering a binary classification $\{relevant, irrelevant\}$, the number of true positives is the amount of documents (tweets) in which the classifier correctly classifies a tweet as relevant when it actually is relevant (according to ground truth), while a true negative is an irrelevant tweet classified as irrelevant. A false positive is an irrelevant tweet that has been classified as relevant (Type I error). A false negative is a relevant tweet classified as irrelevant.

¹http://www.theregister.co.uk/2014/06/04/secret_service_wants_twitter_sarcasm_radar (requested July 14th, 2015)
https://www.fbo.gov/?s=opportunity&mode=form&id=8aaf9a50dd4558899b0df22abc31d30e&tab=core&_cview=0 (requested July 14th, 2015)

The most widely used performance metrics are accuracy, precision and recall. Let us denote the amount of true positives by TP , the amount of false positives by FP , and so on. Then these measures are defined by:

$$\begin{aligned} \text{Accuracy} &= \frac{TP + TN}{TP + FP + TN + FN} \\ \text{Precision} &= \frac{TP}{TP + FP} \\ \text{Recall} &= \frac{TP}{TP + FN} \end{aligned}$$

Accuracy basically tells us the fraction of tweets that have been correctly identified. Precision tells us what fraction of tweets presented is actually relevant, so it basically measures our noise. Recall tells us what fraction of the relevant tweets have actually been identified as relevant, so its complement is actually the amount of relevant tweets we are missing. Note that within our problem, while our aim is to reduce noise and thus increase precision, it would be far worse to lose on recall as the application would then disregard relevant information.

The most common measure that combines precision and recall into a single value is the F_β score, defined by:

$$F_\beta\text{-score} = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}}$$

The most common variant is the F_1 -score, which weighs precision and recall equally well, while a F_2 -score weighs recall twice as much as precision and a $F_{0.5}$ -score weighs precision twice as much as recall. Since it is important not to lose many relevant tweets while still reducing noise, we will primarily adopt the F_2 -measure which weighs recall more.

An alternative to the F-score is the Matthews correlation coefficient, defined by:

$$MCC = \frac{TP \cdot TN + FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}}$$

This measure has been proposed to deal with classes of very different sizes. This applies to our problem, since we have a low signal so the amount of relevant tweets (positives) is far lower than the amount of irrelevant tweets (negatives).

Another measure we will list for completeness during our evaluation is Cohen's Kappa statistic, commonly used while evaluating classifiers. Cohen's Kappa is an inter-rater agreement statistic, and basically tells us how much two raters (in this case ground truth and classifier) agree.

For regression (numerical output class), the above statistics do not apply, and these problems have their own performance measures. The most commonly used statistics are the Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Relative Absolute Error (RAE) and Relative Root Squared Error (RRSE). Let us denote the ground truth value for tweet t by y_t , the corresponding predicted (classifier) output value by \hat{y}_t and the total amount of tweets by n ,

then these measures are defined by:

$$\begin{aligned}
 MAE &= \frac{1}{n} \sum_{\forall t} |\hat{y}_t - y_t| \\
 RMSE &= \sqrt{\frac{1}{n} \sum_{\forall t} (\hat{y}_t - y_t)^2} \\
 RAE &= \frac{\sum_{\forall t} |\hat{y}_t - y_t|}{\sum_{\forall t} |\bar{y} - y_t|} && \text{with } \bar{y} = \frac{1}{n} \sum_{\forall t} y_t \\
 RRSE &= \sqrt{\frac{\sum_{\forall t} (\hat{y}_t - y_t)^2}{\sum_{\forall t} (\bar{y} - y_t)^2}} && \text{with } \bar{y} = \frac{1}{n} \sum_{\forall t} y_t
 \end{aligned}$$

The MAE is basically the average error. The RMSE is similar, but larger individual errors are weighed heavier. RAE and RRSE are respectively their similar relative counterparts, calculating the error relative to the mean. In contrast to the performance metrics for nominal classes, for these metrics a lower value is better.

Query expansion Query expansion [9, 12, 40] is the process of reformulating a seed query to improve retrieval performance in information retrieval operations. In the context of web search engines, query expansion involves evaluating a user's input and expanding the search query to match additional documents. Query expansion involves techniques such as finding synonyms and finding various morphological forms.

Chapter 4

Design and Implementation of our artifact

In Chapter 1 we introduced our case, which we analyzed and defined in depth throughout Chapter 2 resulting in a list of observations. In Chapter 3 we performed research over existing literature, methods and technologies. Now that we have our utilities ready, we can design and compose our artifact. However, we will not be able to apply all suitable technologies and need to make some choices. Furthermore, we may need to scope our problem and introduce some constraints or assumptions under which we will design our artifact. Therefore, in this Chapter we will answer the following questions:

What choices have been made towards a solution? What constraints or which assumptions have we imposed while designing our artifact?

What artifact have we designed to solve our problem?

How does each of the components of this artifact work and why have we designed it like that?

We will address the first question in Section 4.1, and by answering it, we implicitly restrict our solution space to search through and will therefore make it easier to comprehend why we chose the approach presented in subsequent sections.

We will present an overview of our designed artifact in Section 4.2 and make a decomposition of it. In subsequent Sections 4.3, 4.4 and 4.5 each major decomposed part of the system is discussed and explained. We will conclude this chapter with Section 4.6. Throughout this chapter we will tend to relate each choice or decision to an observation. We will not discuss every line of code of our implementation, but will elaborate on relevant specifics and choices.

4.1 Choices made towards a solution

Here we will list some ‘design’ choices that follow from observations (Chapter 2) and challenges/difficulties (Section 2.5). They either explain why our artifact is designed as it is, or scope the problem to a more specific version. Whereas the previous Chapters were just collecting information, from this point onwards, we will be aiming at a solution and making choices based on the collected information. This will also restrict yet focus our search space for a solution. After this section, we will design and present our artifact based on these choices.

Classifier System We have observed that our problem is complex due to many details, and even domain experts have difficulties identifying what they are looking for, or whether a tweet is relevant. Therefore, we will not rely on a static algorithm but use the same calculated properties (features) within a Machine Learning environment. We will be creating a system with a classifier component as its heart. In our research we have seen that this is the most appropriate solution for problems similar to ours. We also consider them to be most capable of handling the problems diversity and complexity. Furthermore, this choice does not really restrict our solution space since there are many types of classifiers and methods of training them.

Two-fold problem We consider the main problem to be split into two major subproblems (Figure 4.1). First of all, the domain expert configuring the instance must think of a lot of scenarios and relationships manually, which makes missing certain scenarios more likely. The properties extracted rely a lot on instance-specific configuration and keywords. We observed that the composition of the keyword lists significantly affects the amount of irrelevant tweets, and is prone to human errors. Therefore, we want to guide the user through a process of setting up the classifier component. Furthermore, we have observed that besides keyword lists, many more factors and properties can affect the relevancy of a tweet, which makes mere keyword matching insufficient. Therefore, we also want to improve the actual classifier system and make it more robust, which makes are problem two-fold with a) Developing a robust classifier, and b) Configuring the classifier as automated and easy as possible.

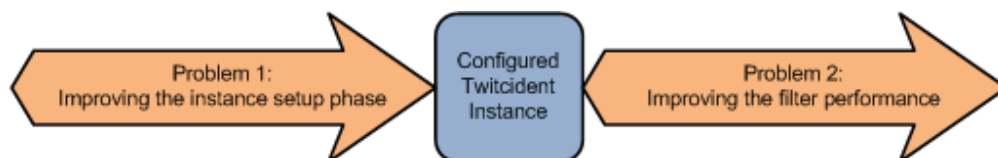


Figure 4.1: Two-fold problem decomposition

Instance variety We observed a lot of variety among instances. However, we want to design an artifact generic and instance-independent enough to be configured by a domain expert, so that our solution is not limited to a specific case. This will largely been taken into account by the aforementioned second part: configuring the classifier.

Incoming streams Our problem involves low signal detection, with only an extremely small fraction of the tweets being relevant. Therefore, missing a single item significantly affects the performance. We also observed that only less than five percent of the tweets have a geo-location attached, so we can and will not rely on this feature (but may use it auxiliary). Furthermore, defining the incoming data stream seems to be complex enough to be considered a separate problem, and therefore we will assume an incoming data stream as precondition to our solution. We will be focusing on optimizing a filtering system and output tweets with relevancy scores to a certain topic. Our solution must correlate as little as possible to the width of the incoming data stream.

IR engines Information Retrieval [4, 36] systems with document stores are not applicable as main engine to solve our problem, since we will not be searching for data, but classifying data. This is a direct consequence of the real-time aspect of our problem. However, IR engines (for example Apache Lucene [24]) may be used as supplementary systems, for example, to improve performance if we would be looking within precalculated caches (recent tweets).

Dynamic keywords We have seen that defined keywords are key to the classification performance, and new scenarios come up all the time. Furthermore, we have observed that users are used to add/remove keywords based on their experiences. Therefore, our artifact should support dynamic word-lists with users able to add, alter and remove keywords during runtime.

Language We are not natural language or linguistic specialists, so we can't dig into language specifics too deep, but we could use existing stable implementations (if available) to support our solution. Furthermore, we want our artifact to be able to extend to multiple languages, we want to clearly identify which parts of our system our language specific. However, we will focus on Dutch as our primary language and will only evaluate this language.

4.2 Artifact decomposition

In the previous section we stated that our problem is two-fold. Therefore, our designed artifact consists of multiple core components, which are tightly coupled. Figure 4.2 illustrates an *architectural overview and context* of our artifact:

- Elliptical boxes indicate processes
- Rectangular boxes indicate data and/or input
- Unfilled boxes indicate initial input and final output
- Blue boxes indicate the current state of Twitcident, without our artifact
- Lighter green boxes indicate our artifact to be developed
- Darker green boxes indicate shared elements

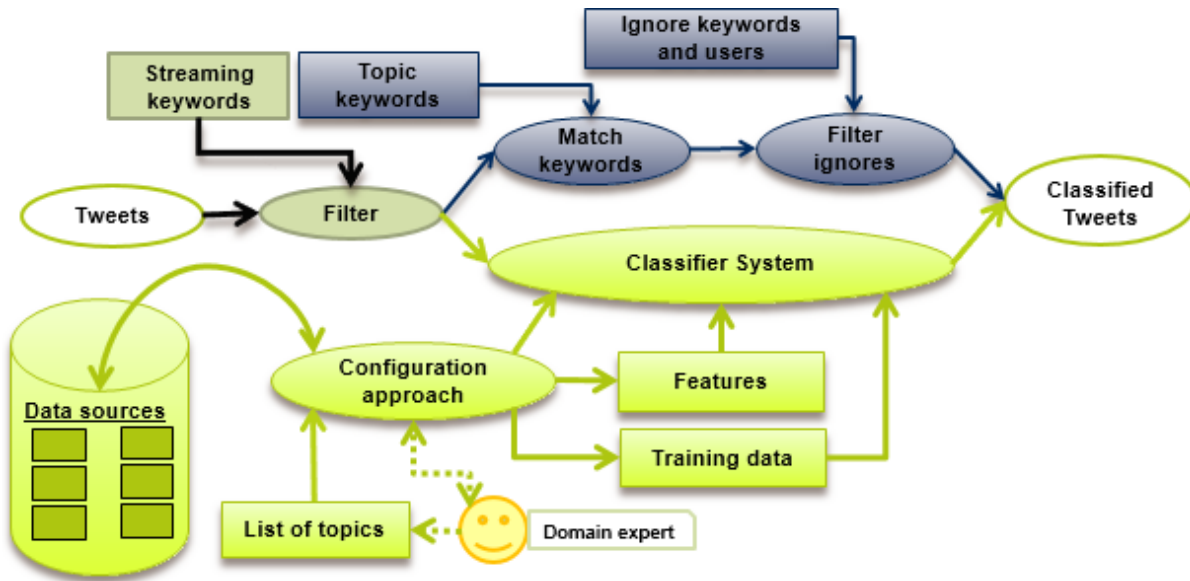


Figure 4.2: Architectural overview and context of our artifact

We can discern two separate tracks from start to end: a blue one and a lighter green one, while both share a common darker green part. The blue part indicates the existing solutions Pipe-App architecture, which we described thoroughly in Section 2.1 and more specifically in Section 2.1.2. The lighter green track represents the high-level architecture of our designed artifact, which we will discuss next. The common darker green prefixed part is the shared part, and actually represents one of the scopes made for our problem: we assume an incoming stream of data.

As our proposed track can be seen as an alternative path to the existing blue track, and the blue track represents the Twitcident Pipe-App, we could say that our aim is to replace and improve over the existing pipe-app, and we output tweets to the current Web-App, which may be modified slightly with enhanced features to configure/drive our solution.

If we take a closer look at our artifact's architecture, we can see two processes (two elliptical boxes), which represent our two-fold problem: a classifier system to improve the actual filtering process, and a configuration approach to set-up, configure and optimize our classifier.

The *Classifier System* consists of a machine learning classifier, and will be discussed in Section 4.3. A machine learning system relies on and is largely defined by its *Features*: properties and attributes of the current “document” to be classified. Examples of such properties in our case would be:

- Does the current tweet contain keyword X?
- Is the current tweet a retweet?
- What kind of user is posting our current tweet?
- How many similar tweets were posted within the last hour?

The values of these features have to be defined and calculated, which is one of the main tasks of this thesis work. We will present our list of features in and will discuss the implementation of these features. Finally, after the classifier itself has been discussed in Section 4.3.3, we will discuss the most important implementation specifics in Section 4.3.4.

After the classifier part is clear, we will move on to the *Configuration Approach*. Most features will need some sort of instance-specific input. Consider for example the feature “Does the current tweet contain keyword X?”. In this case, X is an instance specific keyword which needs to be configured. We will discuss this configuration approach in Section 4.4. To propose for example keywords, this approach relies on some *Data Sources* like vocabularies, dictionaries, thesauri, . . . , which are also depicted in Figure 4.2.

A second part of the Configuration Approach is teaching the classifier what is relevant. This can be done through presenting a corpus of tweets of which the output (relevancy) is known a priori. This process is called training the classifier. However, as this is quite a challenge in our context, we consider this a separate “Part 3” of our problem, which is discussed in Section 4.5.

Figure 4.3 depicts another representation of our artifact, from a more technical point of view. The classifier system (1 - Section 4.3) does the actual work: filtering the tweets by classifying them. However, this system is defined by configured features (1a). This configuration is done by a Setup Approach (2 - Section 4.4). Finally, the system needs to be trained in order to become operational, which is done through the Training Approach (3 - Section 4.5). Note that the Setup Approach (2) and Training Approach (3) together form the aforementioned Configuration Approach.

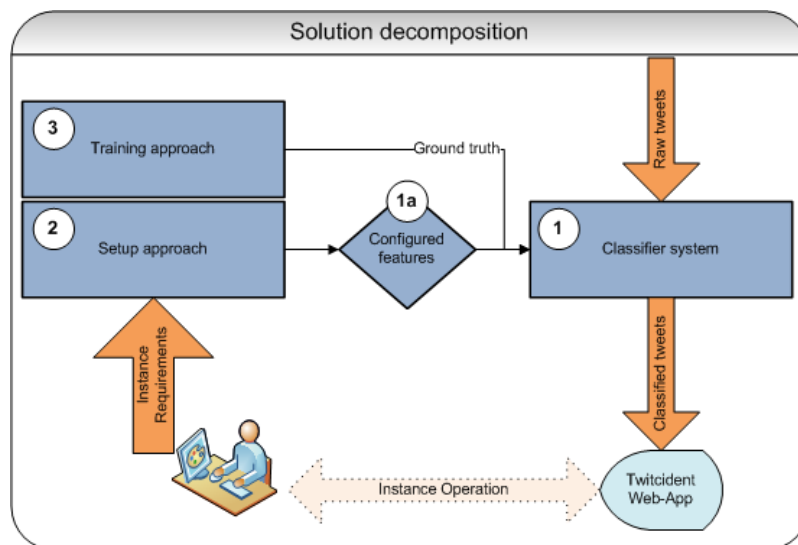


Figure 4.3: Artifact decomposition

4.3 Part 1: Classification

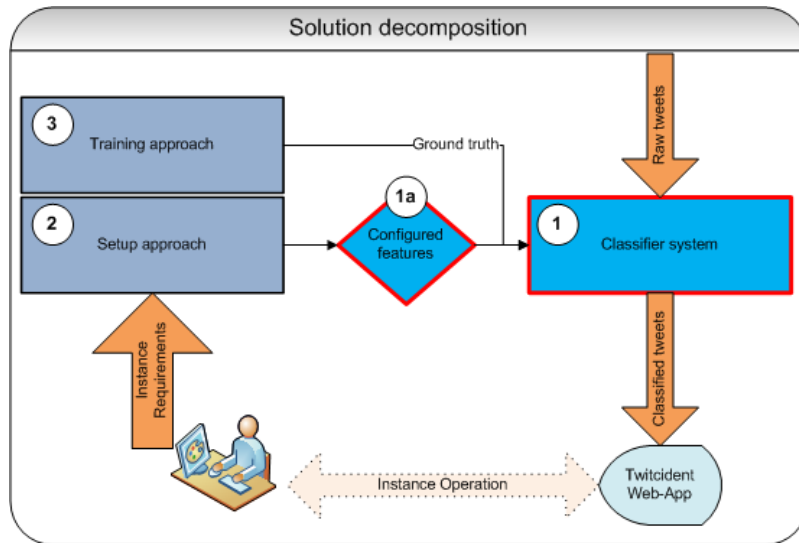


Figure 4.4: Artifact decomposition: Classifier System & Features highlighted

In this section we will discuss the Classifier System, which is highlighted in Figure 4.4. This Machine Learning component will perform the actual filtering by classifying a stream of raw input tweets resulting in a stream of classified tweets. However, as some features will be limited to a single topic, each classifier will classify an incoming tweet by attaching a relevancy score to the corresponding topic. If this relevancy score passes a certain threshold, it can be considered relevant to the given topic. As a result, the Classifier System actually consists of a set of parallel chained classifiers, as depicted in Figure 4.5.

When we refer to a classifier within the next few sections, we actually refer to a single topic classifier element that is designed to classify for a certain topic. The input document for such a classifier is a tweet, over which its features will be calculated. This vector of feature values is then fed to a Machine Learning component, that outputs a

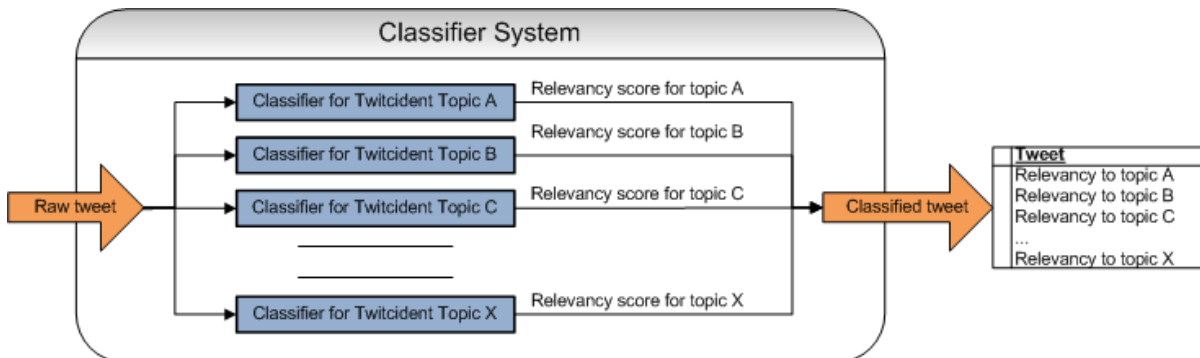


Figure 4.5: Classifier System: Actually a set of parallel classifiers

relevancy score for the Tweet regarding the topic. This relevancy score can then be interpreted by the Twitcident Web-App. The goal of this component is to improve over the current Twitcident filtering mechanism (Pipe-App): it should produce more accurate relevancy scores, as to increase the overall output relevancy.

The list of features will be presented in the next Section (4.3.1), after which they will be discussed and explained in Section A.2. After the features are clear, we will discuss the actual Machine Learning component (Classifier) in Section 4.3.3 and conclude with a rough overview of implementation specifics regarding this part of the system in Section 4.3.4.

4.3.1 List of features

In this section we will present our *initial* and *complete* list of features, whereas we will go into specifics in the next section. Note that not of all these features will actually be implemented due to time constraints, but this serves as an extensive list of potential features. Furthermore, not all features listed and implemented will be operational, as a sub-selection will be made in Chapter 5 based on evaluations.

Tables 4.1, 4.2 and 4.3 list all potential features.

#	Feature summary	Instance dependent	Feature input	Document input	Discussion
1	Tweet, hard match, 1st degr.	Yes	Keywords, 1st degr.	Tweet	Section A.2.1
2	Tweet, hard match, 2nd degr.	Yes	Keywords, 2nd degr.	Tweet	Section A.2.1
3	Tweet, hard match, 3rd degr.	Yes	Keywords, 3rd degr.	Tweet	Section A.2.1
4	Tweet, compound match, 1st degr.	Yes	Keywords, 1st degr.	Tweet	Section A.2.1
5	Tweet, compound match, 2nd degr.	Yes	Keywords, 2nd degr.	Tweet	Section A.2.1
6	Tweet, compound match, 3rd degr.	Yes	Keywords, 3rd degr.	Tweet	Section A.2.1
7	Tweet, stemmed match, 1st degr.	Yes	Keywords, 1st degr.	Tweet	Section A.2.1
8	Tweet, stemmed match, 2nd degr.	Yes	Keywords, 2nd degr.	Tweet	Section A.2.1
9	Tweet, stemmed match, 3rd degr.	Yes	Keywords, 3rd degr.	Tweet	Section A.2.1
10	Tweet, approximate match, 1st degr.	Yes	Keywords, 1st degr.	Tweet	Section A.2.1
11	Tweet, approximate match, 2nd degr.	Yes	Keywords, 2nd degr.	Tweet	Section A.2.1
12	Tweet, approximate match, 3rd degr.	Yes	Keywords, 3rd degr.	Tweet	Section A.2.1
13	Hypernym, approximate match, 1st degr.	Yes	Hypernyms, 1st degr.	Tweet	Section A.2.2
14	Hypernym, approximate match, 2nd degr.	Yes	Hypernyms, 2nd degr.	Tweet	Section A.2.2
15	Hypernym, approximate match, 3rd degr.	Yes	Hypernyms, 3rd degr.	Tweet	Section A.2.2
16	Hyponym, approximate match, 1st degr.	Yes	Hyponyms, 1st degr.	Tweet	Section A.2.2
17	Hyponym, approximate match, 2nd degr.	Yes	Hyponyms, 2nd degr.	Tweet	Section A.2.2
18	Hyponym, approximate match, 3rd degr.	Yes	Hyponyms, 3rd degr.	Tweet	Section A.2.2
19	Username, hard match, 1st degr.	Yes	Keywords, 1st degr.	User descr.	Section A.2.3
20	Username, hard match, 2nd degr.	Yes	Keywords, 2nd degr.	User descr.	Section A.2.3
21	Username, hard match, 3rd degr.	Yes	Keywords, 3rd degr.	User descr.	Section A.2.3
22	Username, compound match, 1st degr.	Yes	Keywords, 1st degr.	User descr.	Section A.2.3
23	Username, compound match, 2nd degr.	Yes	Keywords, 2nd degr.	User descr.	Section A.2.3
24	Username, compound match, 3rd degr.	Yes	Keywords, 3rd degr.	User descr.	Section A.2.3

Table 4.1: List of features, part 1

#	Feature summary	Instance dependent	Feature input	Document input	Discussion
25	Username, stemmed match, 1st degr.	Yes	Keywords, 1st degr.	User descr.	Section A.2.3
26	Username, stemmed match, 2nd degr.	Yes	Keywords, 2nd degr.	User descr.	Section A.2.3
27	Username, stemmed match, 3rd degr.	Yes	Keywords, 3rd degr.	User descr.	Section A.2.3
28	Username, approximate match, 1st degr.	Yes	Keywords, 1st degr.	User descr.	Section A.2.3
29	Username, approximate match, 2nd degr.	Yes	Keywords, 2nd degr.	User descr.	Section A.2.3
30	Username, approximate match, 3rd degr.	Yes	Keywords, 3rd degr.	User descr.	Section A.2.3
31	Ignore, Keyword	Yes	Ignore keywords	Tweet	Section A.2.4
32	Ignore, Username	Yes	Ignore users	Username	Section A.2.4
33	News and media	No	User-database	Username	Section A.2.5
34	Poster's current tweet rate	No		List of users tweets	Section A.2.6
35	Poster's average tweet rate	No		List of users tweets	Section A.2.6
36	Current stream rate	No		Processed tweet cache	Section A.2.6
37	Positive sentiment	No	<i>Research required</i>	Tweet	Section A.2.7
38	Negative sentiment	No	<i>Research required</i>	Tweet	Section A.2.7
39	Subjectivity value	No	<i>Research required</i>	Tweet	Section A.2.8
40	Present tense	No	<i>Research required</i>	Tweet	Section A.2.9
41	Past tense	No	<i>Research required</i>	Tweet	Section A.2.9
42	Is retweet	No		Retweet attribute, Tweet	Section A.2.10
43	#retweets	No		#retweets attribute	Section A.2.10
44	Is reply	No		Reply attributes	Section A.2.11
45	#pre-mentions	No		Mention attributes	Section A.2.12
46	#inner-mentions	No		Mention attributes	Section A.2.12
47	#news-mentions	No	Media usernames	Mention attributes	Section A.2.12
48	Tweet length	No		Tweet	Section A.2.13
49	Images	No		Media attributes	Section A.2.14
50	Links	No		Link attributes	Section A.2.15
51	#inner-hashtags	No		Hashtag attributes	Section A.2.16
52	#post-hashtags	No		Hashtag attributes	Section A.2.16
53	#list-hashtags	Yes	Hashtag list	Hashtag attributes	Section A.2.16
54	Prior similar tweets	No		Processed tweet cache, Tweet	Section A.2.17
55	Has popular term	No		Processed tweet cache, Tweet	Section A.2.18
56	Distance from hotspot	Yes	Hotspots	Geo attributes	Section A.2.19
57	Seems observation	No		Tweet, Geo attributes	Section A.2.20
58	Contains street language/slang	No		Tweet	Section A.2.21
59	#Nouns	No		Tweet	Section A.2.22
60	#Verbs	No		Tweet	Section A.2.22
61	#Adverbs	No		Tweet	Section A.2.22
62	Future day mentioned	No		Tweet	Section A.2.23
63	Past day mentioned	No		Tweet	Section A.2.23
64	Contains special keyword 1	??	Special keywords	Tweet	Section A.2.24

Table 4.2: List of features, part 2

#	Feature summary	Instance dependent	Feature input	Document input	Discussion
65	Contains special keyword 2	??	Special keywords	Tweet	Section A.2.24
66	User language	No		Language attribute	Section A.2.25
67	Tweet language	No		Tweet	Section A.2.25
68	Question marks	No		Tweet	Section A.2.26
69	Subsequent dots	No		Tweet	Section A.2.26
70	Interpunction-ratio	No		Tweet	Section A.2.26

Table 4.3: List of features, part 3

4.3.2 Feature discussions and implementations

In this section we will discuss some of the features that need some elaboration in our opinion. All other features are discussed in Appendix Section A.2.

4.3.2.1 Keyword matches

Features 1–30 relate to keyword matches within tweets and usernames, whereas features 1–12 specifically address keyword matches within tweets. This section will mainly address the latter subset, but does also apply to the complete set of matcher features. However, sections A.2.2 and A.2.3 will extend on this section for other subsets of the matcher features.

The main mechanism of the existing Twitcident classification mechanism relies on keyword matches (Observation 2), and we want to extend on this process. Therefore, a core subset of features is dedicated to keyword matches. However, due to Observation 6 we consider it a high priority to support dynamic keyword lists that can be altered during runtime. As Machine Learning classifiers depend on a fixed set of features over which they are trained, we have chosen for an atypical alternative construction: we define a set of fixed features that are configured with a keyword-list, rather than a single keyword. We regard this as a potential new kind of Machine Learning setup: *features do not have to be static, but may be dynamic*.

This way, the keyword list is extensible and adjustable during runtime, without changing the machine feature and its meaning: this meaning can be interpreted as “relevancy of this tweet with respect to keywords in keyword list KL ”. The main drawback is that all keyword matches are treated equally. For example, if a keyword list contains both “throwing+a+bike” and “shooting+at+police”, and this features value tells the classifier that there is a match, the classifier will not know which particular entry was matched. However, we value the *dynamic extensibility* over this due to Observations 6 and 14, and will capture and resolve the consequences.

Each feature is configured with a keyword list $KL = \{KE_1, KE_2, \dots\}$ with $|KL| \geq 0$, where each keyword-entry KE_i consists of:

- A set of positive keywords $P_{KE_i} = \{PK_1, PK_2, \dots\}$ with $|P| \geq 1$. Each of the individual keywords PK_j must be present to result in a match for this entry.
- A set of negative keywords $N_{KE_i} = \{NK_1, NK_2, \dots\}$ with $|N| \geq 0$. If at least one of the individual keywords NK_k is present, this keyword entry KE_i will never result in a match.

- A weight $W_{KE_i} \in \mathbb{Z} \setminus 0$ for this keyword entry, that will be added to the final output value for this feature for each match.

The set of negative keywords NK serves to restrict the space captured by PK , and define its context (see Observation 12). An example of a very short keyword list KL could be:

$$KL = \begin{cases} KE_1 = \begin{cases} PK = \{\text{'aardbeving'}, \text{'in'}\} \\ NK = \emptyset \\ W = 5 \end{cases} \\ KE_2 = \begin{cases} PK = \{\text{'aardbeving'}\} \\ NK = \{\text{'in'}\} \\ W = 1 \end{cases} \end{cases}$$

In this example, the additional word ‘in’ could indicate that the tweet contains additional information like a place, which would increase the relevancy of the match. Therefore, a match including ‘in’ will increase the feature output value more than a match without an occurrence of ‘in’. Note that in this example, both entries are mutually exclusive, but this does not necessarily be the case.

The final score S of this numerical feature is defined as the sum of keyword entry match weights multiplied by the amount of matches for each entry:

$$S = \sum_{i=1}^{|KL|} (W_{KE_i} \cdot \#matches_{KE_i})$$

Note that if a keyword entry does not match, $\#matches_{KE_i}$ will be zero and the weight is not added to the output value. Also note, that an individual entry is thus allowed to match multiple times.

Suppose another case, in which we are looking for the word ‘bank’. In Dutch this word has multiple contexts, two of them literally translating to a ‘bench’/‘sofa’ and the other one to a ‘bank’ associated with money. On the former one, you can sit. The latter definition one is our intended context. We could now construct the following definitions to more specifically address this context:

$$KL = \begin{cases} KE_1 = \begin{cases} PK = \{\text{'bank'}\} \\ NK = \emptyset \\ W = 1 \end{cases} \\ KE_2 = \begin{cases} PK = \{\text{'bank'}\} \\ NK = \left\{ \begin{array}{l} \text{'zit'}, \quad \text{'tv'}, \quad \text{'beeldbank'}, \quad \text{'bloedbank'}, \\ \text{'boekenbank'}, \quad \text{'spermabank'}, \quad \text{'kennisbank'}, \\ \text{'achterbank'}, \text{'databank'}, \text{'zonnebank'}, \dots \end{array} \right\} \\ W = 5 \end{cases} \\ KE_3 = \begin{cases} PK = \{\text{'bank'}, \text{'zit'}, \text{'tv'}\} \\ NK = \emptyset \\ W = -1 \end{cases} \end{cases}$$

The first entry defines a match on the raw target word ‘bank’, and adds 1 to the output value for each match. The second entry also specifies negative keywords, to

eliminate all unintended meanings: if one of them does occur, this entry will not match. If none of them occur but ‘bank’ does occur, it add 5 to the output value.

Because we do not want to put *all* keywords on a single pile due to some keyword sets being more specific and relevant than others, we defined multiple identical features of this type that will just be configured with their own keyword list. We chose to create multiple degrees (levels) of keyword lists to support this. We will elaborate on this the specific meanings of each degree later on in Section 4.4.

Finally, we also defined multiple *matching schemes*:

Hard match A hard match results in a match for a keyword if it is matched exactly as a word, thus surrounded by spaces.

Compound match A compound match results in a match for a keyword if it is matched exactly at either the beginning or the end of a word, thus has at least one space on its boundaries.

Stemmed match A stemmed match results in a match for a keyword if its stemmed form exactly matches the stemmed form of one of the words of the input document (Tweet). A language specific stemmer [20, 35, 48] is required.

Approximate match To account for typos (Observation 19) and to compensate a bit for the lack of wildcards (Observation 3), we also include an approximate matcher that returns a similarity value in $\langle 0, 1 \rangle$ for each keyword. If this approximation value exceeds a certain threshold (additional feature parameter), it is counted as a match with the weight scaled by this approximation value. After experimenting with well-known string approximation algorithms, we adapted Dice’s similarity metric resulting in an algorithm that suits our problem better. We elaborate on this in Appendix B.

Table 4.4 illustrates an example of the different matching schemes on input (assume every keyword has weight 1):

“Vandaag is het weer kinderboekndag! De volgende boekjes zijn genomineerd: ...”
(*The typo in ‘kinderboekendag’ is on purpose*)

Keyword	Hard	Compound	Stemmed	Our approximation algorithm	Approximate Smith-Waterman
dag	0	1	0	1.00	1.00
boek	0	2	0	1.00	1.00
dagboek	0	0	0	0.00	0.57
kinderboekendag	0	0	0	0.84	0.90

Table 4.4: Illustration match value for different matching schemes

Although we think we have covered the core concepts with this feature, something that remains open for further research is whether order may be relevant and how to integrate that in the matching features (see Observation 26). Furthermore, we could introduce additional constraints to keyword entries, for example, a maximum distance between two individual words.

4.3.2.2 Mentions

Mentions are special constructs in a tweet identified by an '@' sign, indicating a referred user. This usually indicates retweets, replies (conversations) or notifications. We distinguish three types of mentions:

- Pre-mentions: a mentioned user at the beginning of a tweet, excluding known prefixes (for example 'RT: '). If a tweet starts with three consecutive mentions, we consider all of these pre-mentions. These usually indicate retweet or reply type mentions.
- Inner-mentions: a mentioned user that is NOT mentioned at the beginning of a tweet, but in the middle or at the end. These usually indicate notification or reference type mentions.
- News-mentions: More generalized a 'list-mention': the mention can be cross-referenced to a precompiled list (parameter input) of usernames. In our case we will use this specifically to indicate mentions of news-media, because media-tweets often mention themselves, and to identify newsmedia-retweets more accurately.

We constructed three features (45–47) which respectively output the amount of occurrences for each of the defined mention-types.

4.3.2.3 Hashtags

Hashtags are special constructs in a tweet identified by an '#' sign, to put emphasis on a word or to indicate a relation to a certain subject. We distinguish three types of mentions:

- Post-hashtags: a hashtag at the end of a tweet. If a tweet ends with multiple consecutive hashtags, we consider all of these post-hashtags. These usually indicate a relation to a certain subject, or express a conclusive feeling.
- Inner-hashtags: a hashtag occurring in the middle of a tweet. These usually indicate emphasis on the word they are applied to.
- List-hashtags: the hashtag can be cross-referenced to a precompiled list (parameter input) of keywords. Implemented as additional feature that may provide to be useful.

We constructed three features (51–53) which respectively output the amount of occurrences for each of the defined hashtag-types.

4.3.2.4 Prior similar tweets

Often in case of special events or incidents, tweets can be extremely similar and have only minor differences (if any) due to excessive retweeting and replying. Sometimes, a tweet is retweeted multiple times in a chain-linked fashion, each only appending a new 'pre-mention' at the start, but without actually providing any new content.

Therefore, we constructed a feature (54) that looks up the amount of similar prior tweets. This is implemented by maintaining a processed tweet cache (bounded by a maximum size), and attach this value to every processed tweet. This cache is the same as mentioned in Section A.2.6. By traversing this cache from most recent to less recent, we can stop traversing this list when we find the first similar tweet containing such an attached precalculated value, and incrementing it by one. This feature outputs the amount of prior similar tweets given a fixed time window (parameter).

4.3.2.5 Distance from hotspot

By Observation 32 we have seen that geo-information attached to a tweet can provide additional information, especially since most Twitcident Instances monitor a specific area. Therefore, Feature 56 outputs the straight line distance to the closest defined hotspot: this feature requires a list of hotspots as configured input. If no geo-information is attached this feature outputs a distance so large it cannot be relevant to indicate this. We consider a value of 100 kilometers sufficient. For normalization purposes, we also cap the maximum distance to this value; i.e. if someone tweets 250 kilometers away from the closest hotspot, this feature will still output 100 kilometers.

4.3.2.6 Part of Speech attributes

Part of Speech tagging [22, 39, 60] is a mechanism that provide additional useful information on how a tweet is composed. It identifies the part of speech of a word, i.e. whether it is a noun, verb, adverb, ... We constructed three features (59–61) that count the amount of nouns, verbs and adverbs respectively. The ratio of nouns to verbs may tell something about the usefulness of a tweet, while the amount of adverbs tells us how descriptive a tweet is. These features are powered by the Apache OpenNLP library¹ for the actual Part Of Speech tagging.

4.3.2.7 Future or past days mentioned

Tweets posted by news media often mention a specific day, often a day in the past. For example, on a Thursday, news media often mention ‘yesterday’, ‘Wednesday’ or ‘Tuesday’. Such a past day mentioned provides an additional indication that it may be news related content. A future day mentioned on the other hand could indicate an upcoming event, which may be relevant. Therefore, we constructed two features (62–63) that detect days of week mentioned. When such a day is within the next four days with respect to the tweet timestamp, we consider it a future day; a past day otherwise.

4.3.3 Classifier

In Figure 4.6 the actual Classifier System is highlighted. This can be any type of Machine Learning classifier that, based on a feature vector of precalculated values of a data-instance (Tweet), outputs a target value: a nominal class or a numeric value. In our case, this can be a nominal set like {‘Irrelevant’, ‘Relevant’} or a value in $< 0.000, 1.000 >$ indicating the relevancy of the tweet; a value closer to one would be

¹<https://opennlp.apache.org/>

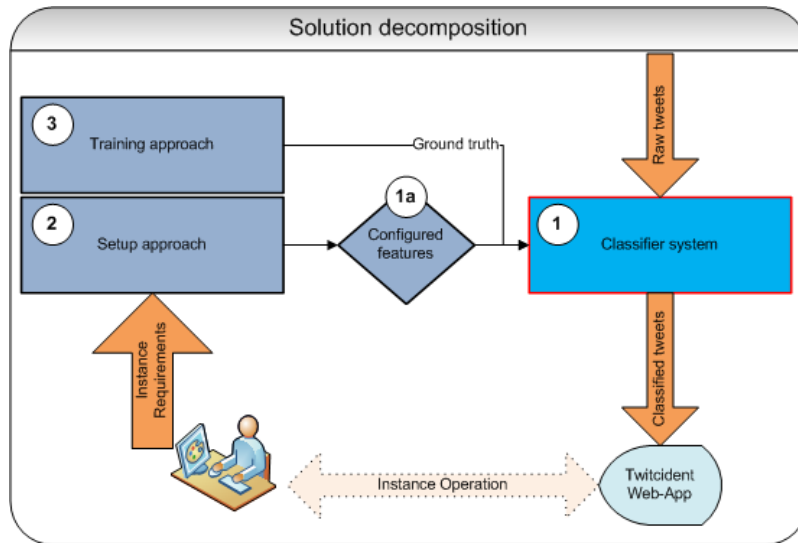


Figure 4.6: Artifact decomposition: Classifier System highlighted

more relevant. However, an output value Y twice that of X does not necessarily mean that Y is twice as relevant, and therefore we should interpret the result on an ordinal scale. The benefits of such a numeric ordinal scale are that:

- tweets can also be “a bit relevant” and “highly relevant”,
- these can be ordered,
- and that the application could feature a user specified threshold slider to give the operator freedom in choosing when a tweet is actually relevant

Therefore, we will experiment with both output setups in our evaluation (Chapter 5).

For our experimental setup and evaluation we have chosen for the Open Source Machine Learning framework WEKA. This Java package was primarily designed to develop and test new Classifier schemes, but is nowadays also used in production environments. It features a high number of existing Classifier implementations, including the option to install new ones by using packages, and also features testing and evaluation schemes built-in, which allow for more effective and standardized evaluation.

4.3.4 Implementation specifics

We have implemented the feature calculation and classifier system in Java, because the original Twitcident was implemented in Java, the large support for libraries like WEKA and the fact that the processes developed for this part run as consoled daemons. More details about this implementation can be found in Appendix Section C.2.

This part of the system is fully database driven, meaning that about everything is configurable through the MySQL databases, including settings, parameters, feature definitions, feature inputs like word-lists, . . . We considered this the most robust form of implementation, with other system parts being able to easily communicate with this

part of the system while still being completely separate modules. It also allows for multiple programming languages to communicate through a shared interface.

The full database specification can be found in Appendix C.1.1. A noteworthy part is the specification and configuration of features, which our dynamically loaded, as depicted in Figure 4.7.

id_instance	id_feature	feature_order	feature_enabled	feature_class	feature_param1	feature_param2	featureinput1_class	featureinput1_param1	featureinput1_param2	featureinput1_class	featureinput2_param1	featureinput2_param2
GG_ABVNG	13_MATCH_TWT_APPR_1	13	1	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_1	NicksAlteredNGramDiceCoefficient	3	0.6
GG_ABVNG	14_MATCH_TWT_APPR_2	14	1	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_2	NicksAlteredNGramDiceCoefficient	3	0.6
GG_ABVNG	15_MATCH_TWT_APPR_3	15	1	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_3	NicksAlteredNGramDiceCoefficient	3	0.6
GG_ABVNG	16_MATCH_TWT_APPR_4	16	0	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_4	NicksAlteredNGramDiceCoefficient	3	0.6
GG_ABVNG	17_MATCH_HPR_APPR_1	17	0	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_HYPER_1	NicksAlteredNGramDiceCoefficient	3	0.6
GG_ABVNG	18_MATCH_HPR_APPR_2	18	0	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_HYPER_2	NicksAlteredNGramDiceCoefficient	3	0.6
GG_ABVNG	19_MATCH_HPR_APPR_3	19	0	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_HYPER_3	NicksAlteredNGramDiceCoefficient	3	0.6
GG_ABVNG	20_MATCH_HPR_APPR_4	20	0	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_HYPER_4	NicksAlteredNGramDiceCoefficient	3	0.6
GG_ABVNG	21_MATCH_HPO_APPR_1	21	0	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_HYPO_1	NicksAlteredNGramDiceCoefficient	3	0.6
GG_ABVNG	22_MATCH_HPO_APPR_2	22	0	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_HYPO_2	NicksAlteredNGramDiceCoefficient	3	0.6
GG_ABVNG	23_MATCH_HPO_APPR_3	23	0	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_HYPO_3	NicksAlteredNGramDiceCoefficient	3	0.6
GG_ABVNG	24_MATCH_HPO_APPR_4	24	0	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_HYPO_4	NicksAlteredNGramDiceCoefficient	3	0.6
GG_ABVNG	41_IGNORE_TERM	41	1	IgnoreMatchFeature	TEXT		FeatureInputs.SimpleList	IGNORE_TERMS				
GG_ABVNG	42_IGNORE_USER	42	1	IgnoreMatchFeature	USER_NAME		FeatureInputs.SimpleList	IGNORE_USERS				
GG_ABVNG	47_SENTIMENT_POS	47	0	SentimentFeature	POSITIVE							
GG_ABVNG	48_SENTIMENT_NEG	48	0	SentimentFeature	NEGATIVE							
GG_ABVNG	52_IS_RETWEET	52	0	MatchesRegexFeature	TEXT	^RT.*						
GG_ABVNG	55_MENTION_PRE	55	1	MentionsFeature	PRE_MENTION							
GG_ABVNG	56_MENTION_INNER	56	1	MentionsFeature	INNER_MENTION							
GG_ABVNG	MENTION_POST	56	0	MentionsFeature	POST_MENTION							
GG_ABVNG	57_MENTION_NEWS	57	1	MentionsFeature	LIST_MENTION		FeatureInputs.SimpleList	USERLIST_MENTIONS				
GG_ABVNG	61_HASHTAG_INNER	61	1	HashtagsFeature	INNER_HASHTAG							
GG_ABVNG	62_HASHTAG_POST	62	1	HashtagsFeature	POST_HASHTAG							
GG_ABVNG	63_HASHTAG_LIST	63	1	HashtagsFeature	LIST_HASHTAG		FeatureInputs.SimpleList	HASHTAGS_1				
GG_ABVNG	64_SIMILAR_TWEETS	64	0	RecentSimilarFeature	0.51	25						
GG_ABVNG	66_HOTSPOT_DIST	66	1	HotspotDistanceFeature	1000000		FeatureInputs.HotspotList	HOTSPOTS				
GG_ABVNG	71_NOUNS	71	1	POSTagFrequencyFeature	NOUN							
GG_ABVNG	72_VERBS	72	1	POSTagFrequencyFeature	VERB							
GG_ABVNG	73_ADVERBS	73	1	POSTagFrequencyFeature	ADVERB							
GG_ABVNG	74_MENTDAY_FUTR	74	1	RelativeDayMentionedFeatu	1,2,3							
GG_ABVNG	75_MENTDAY_PAST	75	1	RelativeDayMentionedFeatu	0,6,5,4							
GG_ABVNG	76_SPECIALKW_1	76	1	CompoundMatchFeature	TEXT		FeatureInputs.WordList	SPECIALKW_POS				
GG_ABVNG	77_SPECIALKW_2	77	1	CompoundMatchFeature	TEXT		FeatureInputs.WordList	SPECIALKW_NEG				
GG_ABVNG	80_QUESTION_MARKS	80	1	RegexOccurrenceFeature	TEXT	\?						
GG_ABVNG	81_SUBSEQ_DOTS	81	1	RegexOccurrenceFeature	TEXT	[.]{2}						
GG_ABVNG	82_INTERPUNC_RATIO	82	1	InterpunctionRatioFeature								

Figure 4.7: Database table: a random subset of a feature configuration

Every feature is instantiated by the class given by *feature_class* that is passed four parameters: two optional variable field parameters and two optional *FeatureInput* classes. Each such *FeatureInput* is also given two variable field parameters. The interpretation of all variable field parameters depend on the feature class or the input class. All *FeatureInput* classes load their data from a table as depicted in Figure 4.8.

The *FeatureCalculator* unit outputs its results in three specifiable ways: directly as a return value for further real-time processing, as a CSV export file and/or as an ARFF file. ARFF is the preferred dataset input format for WEKA. However, in the original ARFF specification, it is very hard (impossible without pre-filtering) to attach metadata like ID and tweet text to each feature vector without affecting the classifier. In Appendix Section C.2 we explain how we resolved this issue.

id_instance	id_featureinput	entry1	entry2	meta1	meta2
GG_ABVNG	HASHTAGS_1	aardbeving			
GG_ABVNG	HASHTAGS_1	trillen			
GG_ABVNG	HOTSPOTS	53.18605	6.56567	Groningen-Zuid	
GG_ABVNG	HOTSPOTS	53.21937	6.51915	Groningen-West	
GG_ABVNG	HOTSPOTS	53.2207	6.61528	Groningen-Oost	
GG_ABVNG	HOTSPOTS	53.23869	6.56653	Groningen-Noord	
GG_ABVNG	WORDLIST_1	+{aardbeving}[5]			
GG_ABVNG	WORDLIST_1	+{aardschok}[3]			
GG_ABVNG	WORDLIST_1	+{beving}[3]			
GG_ABVNG	WORDLIST_1	+{schok}[1]			
GG_ABVNG	WORDLIST_2	+{aardbeving}+{zware}[2]			
GG_ABVNG	WORDLIST_2	+{aardbevinkje}[2]			
GG_ABVNG	WORDLIST_2	+{aardschokje}[2]			
GG_ABVNG	WORDLIST_2	+{beeft}-{angst}[3]			
GG_ABVNG	WORDLIST_2	+{beven}-{angst}[2]			
GG_ABVNG	WORDLIST_2	+{rammelt}-{honger}[2]			

Figure 4.8: Database table: a random subset of some feature inputs

4.4 Part 2: Setup Approach

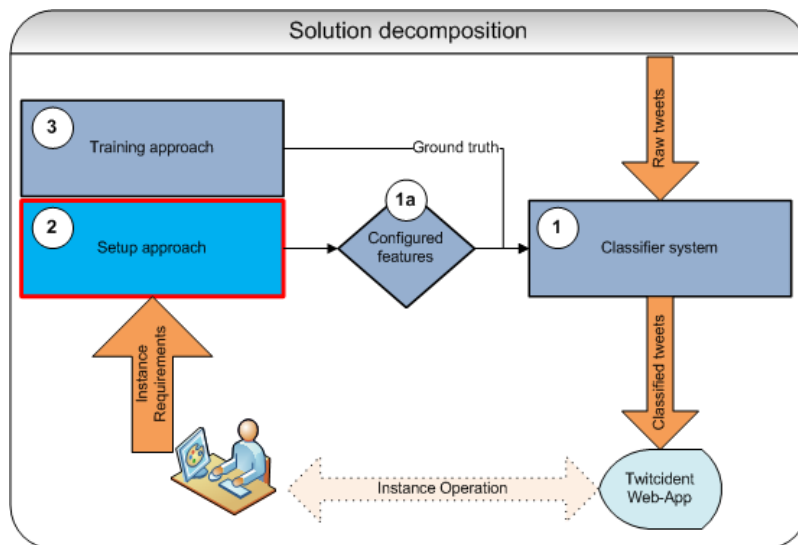


Figure 4.9: Artifact decomposition: Setup Approach highlighted

In this section we will discuss the Setup Approach component, which is highlighted in Figure 4.9. This part consists of a graphical User Interface that interacts with a domain expert as to setup and configure a new instance. In Section 4.3 we have seen that several features require preconfigured instance-specific input. The aim of this part of the system is to identify, determine and collect these inputs. In some cases, we will need to collect such input (suggestions) from data sources.

Figure 4.10 illustrates how we will come up with our Setup Approach. At the start of our project, there was no specific definition (Fig. 4.10: A) on how to setup a Twitcident Instance, and it required a lot of creativity. During this project, we tried to

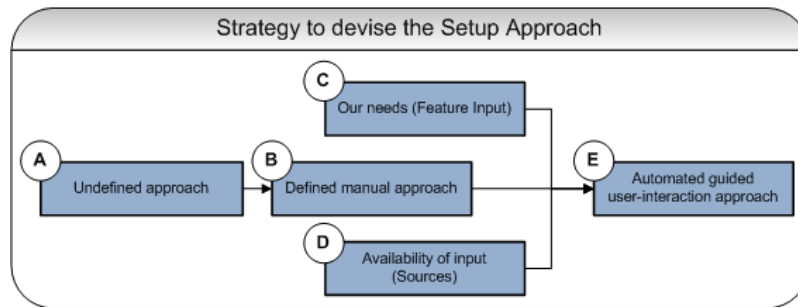


Figure 4.10: Our working strategy to devise the Setup Approach

formalize this process by defining a guideline for it (Fig. 4.10: B), while it still needed to be done by hand without the help of automated tools. We will present the result in Section 4.4.1. By this, we will grasp the idea of what we want to achieve and how to improve on it.

The goal of the Setup Approach is to setup and configure our Classifier System. Basically, this comes down to defining all required Features (Section 4.3.1) inputs, which we consider our needs (Fig. 4.10: C). We will explicate all these inputs required in Section 4.4.2. To help and guide the user in this process, we want to implement several tools that connect to existing data sources, like dictionaries, thesauri, etc. We consider this the availability of input (Fig. 4.10: D), and have listed our analysis of the availability of these sources in Appendix D.

After that, we will have an overview of the source availability, needs and process definition. If we combine these, we can come up with our user-interacted Setup Approach (Fig. 4.10: E) that will guide a domain expert through setting up and configuring the new instance. This solution will be presented in Sections 4.4.3, 4.4.4 and 4.4.5, with its end result being a fully configured list of features.

4.4.1 Manual setup approach

In its former current state, Twitcident Instances were configured manually by defining all keywords and ignore terms by hand. During our thesis work, we attempted to formalize and define this process as a guideline, which could later be used to automate this task.

We proposed the following method:

1. Choose an initial short descriptive topic keyword.
2. Think about observations related to this keyword: What do you see when ...? What do you smell when ...? What do you hear when ...? What do you feel when ...? Repeat step 2-5 for identifying keywords, or extend existing keywords with such terms.
3. Think about the context of this keyword: How do you become a ...? (i.e. 'victim' or 'casualty')
4. For this keyword, generate keyword combinations that would be relevant.

5. For this keyword, obtain synonyms (Dictionary, Thesaurus, Synonym.com, Wiktionary, ...). Generate keyword combinations with this synonym that would be relevant.
6. For this keyword, obtain hyponyms and hypernyms (Wiktionary, ...). If these would be relevant, add keyword combinations for them
7. For this keyword, check street language or slang for it. If existing, add keyword combinations for them.
8. For each of the generated keyword combinations, check whether both their singular or plural form are captured, either by adding both entries or with the use of wildcards.
9. For each of the generated keywords, check if there are alternative spellings or common misspellings (for example, by Googling it). If existing, make sure these are captured, either by adding both entries or with the use of wildcards.
10. For each of the generated keywords, obtain their compound forms (Wiktionary, ...). For example 'fire' would among others yield 'fireworks', 'campfire', 'fireman', 'fireball', 'firefight', 'fireproof', ... Some of these will be relevant and therefore should be added as additional keyword combinations, others will yield noise and could be added as ignore terms.
11. For each of the keywords added in Step 10, repeat Steps 4–9.
12. For each of the generated keywords, obtain potential contexts/definitions. At least one of them will be the intended context/definition, while others would probably yield noise. Add identifying ignore terms in general or to each keyword combination to eliminate irrelevant contents.
13. For each of the generated keywords, check whether there exist proverbs, jokes, sayings and other uses that would introduce noise, and add ignore terms to exclude those.
14. For each of the generated keywords, check and remove for compound entities. For example, ignore 'blood' if it directly occurs after 'true': True Blood is a TV series.
15. For each of the generated keywords, check for uses on other language, and try to exclude these by adding common related terms from the unwanted language as ignore terms.
16. For each of the generated keyword combinations, check and assess its tense. Generally, only present/future tense will be relevant. Past tense forums may be added as ignore terms.
17. For each of the generated keyword combinations, search for it on Twitter and analyze its usage to gain additional insight. Add additional keywords or ignore terms based on the results and repeat Steps 4–16, or remove the keyword combination if it only yields undesired results.

Note that Steps 1–11 are converging mechanisms that expand the amount of data captured, while Steps 11–17 our diverging mechanisms that remove undesired uses and irrelevant content by restricting the scope.

4.4.2 Required inputs

In Section 4.3, we listed and discussed the features that were defined as parts of our artifact. Some of these features require input. In table 4.5 we list the features that require input, and what input they require. This basically is a truncated list of Tables 4.1, 4.2 and 4.3.

Features	Required input
1–12, 27–30	Three degrees of keyword lists
13–15	Three degrees of hypernym lists
16–18	Three degrees of hyponym lists
31	Ignore terms
32	Ignore users
53	Hashtag list
56	Hotspots
64–65	Two list extensions of special keywords

Table 4.5: List of required instance-dependent feature inputs

We have researched the availability of such input resources, and have listed the outcome in Appendix D. Such resources include Dictionaries, Semantic sources, Synonym sources, Encyclopedia, Proverb and saying listings and other sources.

4.4.3 Three levels of wordlists

In our feature list, we have noted that we have three levels of matching features corresponding to three wordlists. We also noted that we want to aggregate keywords in wordlists for a single feature, to be able to adjust the wordlists after training while keeping the amount of features constant. Based on the following definitions of each level, we consider three such levels a sufficient amount:

- The first level represents all words and word expansions that have been collected and approved through the guided process, excluding all ignore terms (therefore matching if the positive terms match).
- The second level represent all words and word expansions that have been collected and approved through the guided process, including all ignore terms (therefore, only matching if all positive terms match but none of the negative terms match).
- Finally, the third level consists of words that are frequently correlated to the words on the former two lists, both positively as well negatively correlated ones (the latter have a negative entry weight).

Note that the second level list actually is our intended base list. However, as we noticed that one can easily exclude relevant tweets by including an ignore term and

value the recall of our system the first level list is included. The third level list is included to add extra contextual information (see Observation 12), and is similar to the pseudo-relevance feedback (PRF) mechanism used by Olteanu et al. [45].

4.4.4 Word expansion

The most valuable and robust source is Wiktionary, and this is also the only source which provides a publicly available API to obtain the content. All other sources need to be scraped from their web-pages. Given a keyword, we request and scrape all information about the word we can possibly obtain, like synonyms, definitions, usages, proverbs, . . . We will call this *word expansion*.

Given a keyword, we first call the Wiktionary ‘parse’ API and request the ‘text’ and ‘templates’ section. Based on the available templates, we can deduct which sections and formatted tables are available on the page. Using lots of regular expressions we parse each such section to collect all available useful information. An overview of this process is listed in Algorithm 3, located in Appendix Section C.3.

Similar processes are executed on the other data-sources to expand the word even further, although we need to directly scrape those sources that do not have an API, and most sources only expand parts of the return object R as they mostly focus on a limited set of aspects. We call this process *word expansion* throughout the rest of this chapter, and the function $expandWord(W)$ will refer to this process with W being the target word. An example output is listed in Figure 4.11. More examples of complete expansions can be found in Appendix E.

```

word => "brand"
type => [ "verb",
         "noun" ]
defs => [
  0 => "eerste persoon enkelvoud tegenwoordige tijd van branden
       Ik brand."
  1 => "gebiedende wijs van branden
       Brand!"
  2 => "(bij inversie) tweede persoon enkelvoud tegenwoordige tijd van branden
       Brand je?"
  3 => "keer dat vuur iets verbrandt"
]
derived_from => [
  word => "branden"
  type => [ "verb",
           "noun" ]
  defs => [ ... truncated ... ]
  verbal_forms => [
    0 => "gebrand"
    1 => "brandend"
    2 => "branden"
    3 => "brand"
    4 => "brandt"
    11 => "brandde"
    16 => "brandden"
  ]
  related => [ "verbranden" ]
]
plural => [ "branden" ]
diminutive => [
  singular => [ "brandje" ]
  plural => [ "brandjes" ]
]
synonyms => [ "fik",
              "hens" ]
related => [ 0 => "vuurzee" ]
hyponyms => [ "aardbrand", "bedrijfsbrand", "bermbrand", "binnenbrand",
              "boombrand", "bosbrand", "builenbrand", "duinbrand",
              ... truncated ...
              "wortelbrand", "zeebrand", "zonnebrand", "zuurbrand" ]
hypernyms => []
usages => [
  0 => "in brand staan"
  1 => "Er is brand uitgebroken."
  2 => "bosbranden"
]
proverbs => [
  0 => "as is verbrande turf. (=aan een belofte (as = als) heb je niets)"
  1 => "bang zijn zich aan koud water te branden (=erg voorzichtig zijn)"
  2 => "beter hard geblazen dan de mond gebrand. (=het is beter dat men
       zich inspant dan dat er door slordigheid of luiheid iets fout gaat)"
  ... truncated ...
  22 => "zijn schepen achter zich verbranden (=obstinaat doorgaan, zodanig
        dat men niet meer terug kan)"
  23 => "zijn vingers aan iets branden (=zich in iets vergissen, nadeel aan
        iets ondervinden)"
]

```

Figure 4.11: A truncated example word expansion of the word 'brand' ('fire')

4.4.5 Setup approach

Algorithm 1 (page 70) describes the process we propose to setup our instance with features and their required inputs. Whenever an undefined variable is encountered, assume an empty object or set. All line numbers mentioned in this section refer to lines in this algorithm.

A *PrimaryWord* consists of three sets of words:

base terms Contain the first input *PrimaryWord* as well as its synonyms and conjugations

combo terms Contains words that may increase relevancy if it occurs together with a base term

ignore terms Contains words that would yield the entry irrelevant: mostly because the word is used in another context than the intended context

First, we collect some initial properties and user choices (lines 3–6), followed by initial descriptive keywords (line 7). We will ask the user control questions, for example what will be observed: what can be seen? or heard? or smelled? or felt? How would Twitter users tweet about it?

We proceed to loop (lines 8–38), composed of a two-phase process for each keyword (lines 17–37), preceded by preliminary word expansion (lines 10–16). The word expansion function is called on all unexpanded base terms of each *PrimaryWord*. During this process, the words may be substituted if they are derived from a more common base form, i.e. for verbs this would be the infinitive while for nouns it would most likely be its singular form in case of plural input.

The first of these two phases performs a base term expansion by listing synonyms, conjugations and diminutives (lines 17–23). Besides base term expansion, new *PrimaryWords* can also be added if desired. The two-phase processing is required because the second phase processing of a word requires all base terms of the *PrimaryWord* to be complete and expanded for further processing.

During the second phase (lines 24–37) the combo terms and ignore terms are added, while also still enabling the user to add additional base terms of even new *PrimaryWords* if encountered in the displayed contexts. This phase is split in two separate views to increase user experience and efficiency: the first part (lines 25–29) focuses upon contexts that are more likely to yield positive terms (combo terms) while the second part (lines 30–34) is more likely to yield negative terms (ignore terms).

After all *PrimaryWords* have passed this two-phase process, we request the user (line 40) to select all base and combo terms that would also be relevant as a hashtag (and enable the user to specify new ones if desired), and attach a weight to all base and combo terms (line 41).

After we have acquired weights for each word, we can start generating the first two degrees of *WordLists* (lines 42–53), as explained in Section A.2.1. *WordEntries* are generated by combining one of the base terms with all subsets of combo terms and adding all ignore terms as negative entries. Recall that the second degree list is composed of all regular *WordEntries*, while the first degree list is the same while excluding all ignore terms. The *WordEntry* weight is calculated by

Type	Term(s)	Weight	Actual matching weight
Base-term	aardbeving	5	
Combo-term	aardbeving	2	
Combo-term	aardbeving	1	
WordEntry	aardbeving	$\lceil (5)^{1.0} \rceil = 5$	5
WordEntry	aardbeving + trillen	$\lceil (5 + 2)^{1.1} \rceil = 9$	$5 + 9 = 14$
WordEntry	aardbeving + lawaai	$\lceil (5 + 1)^{1.1} \rceil = 7$	$5 + 7 = 12$
WordEntry	aardbeving + trillen + lawaai	$\lceil (5 + 2 + 1)^{1.2} \rceil = 11$	$5 + 9 + 7 = 21$

Table 4.6: Example WordEntry weight calculation for base term ‘aardbeving’ and combo terms ‘trillen’ and ‘lawaai’ using $\alpha = 1.0$ and $\beta = 10$

$$\left[\left(\sum_{w \in T^+} weight_w \right)^{\left(\alpha - \frac{1}{\beta} \right) + \left(\frac{|T^+|}{\beta} \right)} \right]$$

where T^+ is the set of positive matching terms and α and β are constant parameters ≥ 1 . The sum of term weights is powered by the amount of terms in a WordEntry, yielding a slightly exponential increase with more terms. This way, more specific combinations are preferred over less specific combinations. Also keep in mind that the matching features sum all match results together, so all less specific matches are also matched and their weights added. Therefore, we consider a very slight exponential increase sufficient, and chose $\alpha = 1.0$ and $\beta = 10$. Table 4.6 illustrates an example case.

After generating the first two degrees of *WordLists*, we construct the hypernym and hyponym lists (lines 54–55). As the terms on these lists are all derived, there is no need to make multiple lists for each degree.

Next, we obtain the final inputs from the user (lines 56–59) required for Features 31–32, 56, 64–65: ignore users, global ignore terms, hotspots, ... These inputs are all optional.

We continue with extracting correlated words that often co-exist when using the currently defined matchers and seem to have a positive or negative impact (lines 60–73). This process compensates for terms the current process did not propose and the user did not think of, and for potential event detection (Appendices G and H). The result is the third degree WordList.

Finally, we allow the user to disable some features if desired (line 74).

Algorithm 1 Setup approach

```

1: procedure SETUPINSTANCE
2:    $\mathcal{E} \leftarrow \{\}$   $\triangleright \mathcal{E}^{word}$  will contain the word-expansion for word
3:   Request instance properties (ID, Title, ...) from user and create new instance
4:   Request which verbal tenses may be relevant and should be retrieved
5:   Request whether news is relevant
6:   Request whether images and links would likely increase relevancy
7:   Request initial descriptive topic terms and put them in PrimaryWords.
8:   while  $\neg nothingDone$  do
9:     nothingDone  $\leftarrow true$ 
10:    for all  $W \in \{PrimaryWords \mid \neg W_{expanded}\}$  do
11:      if  $W$  is derived from another word  $D$  and user wants to replace it then
12:        Substitute  $W$  by  $D$  in PrimaryWords
13:      end if
14:       $\mathcal{E} \leftarrow \mathcal{E} \uplus EXPANDWORD(W)$ 
15:       $W_{expanded} \leftarrow true$ 
16:    end for
17:    if  $\exists W \in \{PrimaryWords \mid \neg W_{phase1\_processed}\}$  then
18:      Show the user a page with all synonyms, verbal conjugations,
19:      plural forms and diminutives from  $\mathcal{E}^W$  and let the user attach a category
20:       $\{irrelevant, relevant, primary\}$  to each of these terms: we name this set  $T$ 
21:       $PrimaryWords \leftarrow PrimaryWords \cup \{t \in T \mid primary\}$ 
22:       $W_{base\_terms} \leftarrow W_{base\_terms} \cup \{t \in T \mid relevant\}$ 
23:       $W_{phase1\_processed} \leftarrow true$ 
24:      nothingDone  $\leftarrow false$ 
25:    end if
26:    if  $\exists W \in \{PrimaryWords \mid \neg W_{phase2\_processed}\}$  and nothingDone then
27:      Show a page of all entries  $\bigcup_{w \in W_{base\_terms}} (\mathcal{E}_{defs}^w \cup \mathcal{E}_{usages}^w \cup \mathcal{E}_{hyponyms}^w)$ 
28:      and make each word individually draggable to any of the following categories
29:       $\{synonym, combo, ignore, primary\}$ 
30:       $PrimaryWords \leftarrow PrimaryWords \cup primary$ 
31:       $W_{base\_terms} \leftarrow W_{base\_terms} \cup synonym$ 
32:       $W_{combo\_terms} \leftarrow W_{combo\_terms} \cup combo$ 
33:       $W_{ignore\_terms} \leftarrow W_{ignore\_terms} \cup ignore$ 
34:      Show a page of all entries  $\bigcup_{w \in W_{base\_terms}} (\mathcal{E}_{proverbs}^w \cup \mathcal{E}_{hyponyms}^w)$  and
35:      make each word individually draggable to any of the following categories
36:       $\{synonym, combo, ignore, primary\}$ 
37:       $PrimaryWords \leftarrow PrimaryWords \cup primary$ 
38:       $W_{base\_terms} \leftarrow W_{base\_terms} \cup synonym$ 
39:       $W_{combo\_terms} \leftarrow W_{combo\_terms} \cup combo$ 
40:       $W_{ignore\_terms} \leftarrow W_{ignore\_terms} \cup ignore$ 
41:       $W_{phase2\_processed} \leftarrow true$ 
42:      nothingDone  $\leftarrow false$ 
43:    end if
44:  end while

```

Algorithm 2 Setup approach (continued, 1)

```

39:   $\mathcal{W} \leftarrow \bigcup_{W \in \text{PrimaryWords}} (W_{\text{base\_terms}} \cup W_{\text{combo\_terms}})$ 
40:  Present all  $\mathcal{W}$  and let user identify which should be added to the Hashtags set.
    Also allow creation of new hashtags.
41:  Present all  $\mathcal{W}$  and let user specify weights for all individual words
42:  for all  $\{W \in \text{PrimaryWords}\}$  do
43:    for all  $b \in W_{\text{base\_terms}}$  do
44:      for all  $\{subset \mid subset \subset W_{\text{combo\_terms}}\}$  do
45:         $WordEntry_{\text{positiveEntries}} \leftarrow b \cup subset$ 
46:         $WordEntry_{\text{negativeEntries}} \leftarrow \emptyset$ 
47:         $WordEntry_{\text{weight}} \leftarrow \lceil (\sum_{w \in WordEntry_{\text{positiveEntries}}} weight_w)^{1.0 + (\frac{|WordEntry_{\text{positiveEntries}}|}{10})} \rceil$ 
48:         $WordList1 \leftarrow WordList1 \uplus WordEntry$ 
49:         $WordEntry_{\text{negativeEntries}} \leftarrow W_{\text{ignore\_terms}}$ 
50:         $WordList2 \leftarrow WordList2 \uplus WordEntry$ 
51:      end for
52:    end for
53:  end for
54:   $HyponymList \leftarrow \bigcup_{W \in \text{PrimaryWords}} (E_{\text{hyponyms}}^{\text{base\_term}})$ 
55:   $HypernymList \leftarrow \bigcup_{W \in \text{PrimaryWords}} (E_{\text{hypernyms}}^{\text{base\_term}})$ 
56:  Request all users to ignore
57:  Request all global ignore terms
58:  Present a map and request the user to geo-tag all hotspots
59:  Present two lists of default special keywords and let the user confirm/extend
60:  for all  $W \in WordList1 \cup WordList2$  do
61:    Sample a set of tweets matching  $W$  while filtering based on given properties
    (ignore terms, ignore users, news relevant, ...) and append to  $\mathcal{T}$ 
62:  end for
63:  Extract most frequent co-occurring terms  $O$  in  $\mathcal{T}$  that are not a term in
     $WordList1 \cup WordList2$ 
64:  Filter  $\mathcal{T}$ : keep elements that contain a term  $\in O$ 
65:  Present  $\mathcal{T}$  and let the user classify these tweets in  $\{\mathcal{T}_{\text{positive}}, \mathcal{T}_{\text{negative}}\}$ 
66:  for all  $o \in O$  do
67:     $Score_o \leftarrow \frac{|\{t \in \mathcal{T}_{\text{positive}} \mid o \in t\}|}{|\{t \in \mathcal{T} \mid o \in t\}|}$ 
68:    if  $Score_o > \theta^+$  then  $\triangleright \theta^+$  is a upper threshold  $0.5 < \theta^+ < 1.0$ 
69:       $WordList3 \leftarrow WordList3 \uplus \langle \{o\}, \emptyset, 1 \rangle$ 
70:    else if  $Score_o < \theta^-$  then  $\triangleright \theta^-$  is a lower threshold  $0.0 < \theta^- < 0.5$ 
71:       $WordList3 \leftarrow WordList3 \uplus \langle \{o\}, \emptyset, -1 \rangle$ 
72:    end if
73:  end for
74:  Present the list of features and let the user confirm it, or disable some of them
75: end procedure

```

4.5 Part 3: Training Approach

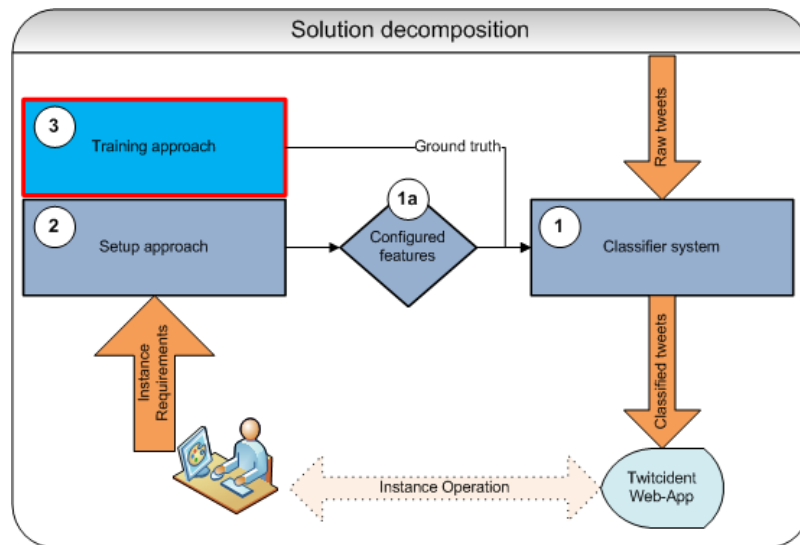


Figure 4.12: Artifact decomposition: Training Approach highlighted

In Section 4.3 we discussed the actual classifier and all potential features. In Section 4.4 we explained how we propose to setup an instance and its feature configuration. With those parts setup, it is time to learn our classifier how to interpret the features by training is on a dataset, so it can become operational. Figure 4.12 illustrates this component.

In this section, we will briefly discuss this component but will consider its concrete implementation outside the scope of this thesis work, as it is another quite complex component requiring research and evaluation on its own. Furthermore, we will not require this training component to evaluate our proposed system, as we will have different means of gathering training data. We elaborate on this in Appendix F.

The problem at hand seems trivial at first, but is quite a complex one. The following difficulties arise:

1. We cannot reasonably expect the configuring user to manually classify hundreds or thousands of tweets, so the dataset should be as limited as possible. How can we keep the amount of effort to a minimum?
2. If we would collect random tweets, we cannot reasonably expect to have *any* relevant tweet due to the extremely low signal we are trying to detect. Even if we would perform a targeted keyword search, the signal can still expected to be very low. Furthermore, performing such a targeted keyword search is likely to introduce a bias in our dataset. How could we prevent such a bias? How can we sample a representative dataset? How can we balance between representativity and the risk of a biased dataset?
3. From what time window to sample tweets? Some events and monitors may be time-relevant while others are not. Sampling from a small window could also

introduce a bias if something special was happening at that time. Should the sampling window be instance specific? How to determine such bounds?

4. We expect every feature needs its full output range to be represented in the dataset in order to be accurately represented and interpreted by the classifier system. For example, if a feature tells us how many links there are in a tweet and no entry in our dataset contains a link, how can the classifier learn the significance of that feature? Furthermore, if in the same case a single document (tweet) contains a link and is classified as relevant, this could mislead the classifier to learn that a link is always relevant, while this cannot be reasonably stated based on a single document. If with our amount of features every feature has its output range fully represented by a number of cases, the size of the dataset is going to be enormous. This is contrasting to our first listed difficulty.

We propose the following conceptual approach, hereby assuming the output of the classifier is a numeric value $0.000 \leq output \leq 1.000$ representing the relevancy of a document (tweet):

1. For each feature, collect at least $n \geq 10$ samples for each of its output values. For text matchers, this means n hits on each of its WordEntries on the given target field. For Feature 50 regarding link occurrences, this means n hits on any amount of links up to a certain value, i.e. up to 2 links. We should determine appropriate values of n for each feature. For strings matchers, $n = 10$ may be sufficient, whereas for the links features a much larger value $n \geq 50$ may be required. This is directly related to the expected significance of the feature on its own: text matchers are pretty domain-specific and can therefore be expected to be more relevant than the amount of links, of which the impact is much harder to estimate and which depends more on its context and thus surrounding features.
2. Apply a static manually constructed pre-training algorithm on this dataset pre-classify these documents (tweets). This algorithm relies on the same calculated feature vector, but we manually *reason* over it: we base this algorithm on what we expect to be the significance of each feature. For example, we can expect the significance of text matchers to be positive with a higher output, while we expect links and images to be less relevant but still a positive impact, and the ‘Is retweet’ or ‘Similar prior tweets’ features to have a negative impact.

This algorithm than aggregates the values and corresponding significances together. In its most simple form this could be by a linear regression formula:

$$\sum_{\text{Feature } \mathcal{F} \in \text{featurevector}} output_{\mathcal{F}} \cdot significance_{\mathcal{F}}$$

However, we could also introduce some feature dependencies to relate feature outputs, i.e. introducing conditional constructs between features. This also leads us to think that Genetic Programming² [29] may very well be of use in this process.

²<http://www.genetic-programming.org/>
<http://www.geneticprogramming.com/>

The pre-training of this algorithm should be executed with a lot of uncertainty. When the actual classifier output would be in the normalized range $[0.000, 1.000]$, this pre-training algorithm should output values in $[0.500 - \delta, 0.500 + \delta]$. Our initial thoughts are that $\delta = 0.100$ would be a sufficient value.

3. We would then repeat the first step to acquire a much smaller dataset that can be annotated by hand, let the user classify these as relevant or irrelevant and attach values on a “more certain” range. For example, when we take $\delta = 0.250$, a tweet classified as irrelevant will receive a score of 0.250 while a relevant tweet would receive a score of 0.750. We will save all classifications for later re-training.
4. Next, we should somehow collect the tweets the classifier is most uncertain about, either where the output score is close to 0.500, or where the retrained classifier differentiates from the pre-trained classifier, and let the user classify these tweets more accurately. Example classifications could include the set $\{\text{Relevant}(0.9), \text{A bit relevant}(0.7), \text{A bit irrelevant}(0.3), \text{Irrelevant}(0.1)\}$, with an example score between parenthesis. We would need to research and explore the possibilities of *Semi-supervised learning* [10,62] and *Active learning* [52] in this case, because our idea seems to be very closely related to these concepts.
5. We could sample tweets from Twitter that the current classifier classifies as relevant, and let the user judge the results. Based on all results (document-score pairs) we could retrain the classifier, and repeat the process.
6. When the instance is operational, a user feedback mechanism should be implemented such that the user can mark tweets as relevant or noise (irrelevant), and use these classifications either to periodically retrain the system, or apply *Online learning* [19] to it.

With this discussion of the training component, we have proposed an entry-point for further research and a concrete implementation. For this thesis work, we do not require this component to evaluate our prototype, and will use an alternative way to obtain training data (Appendix F) for evaluating our prototype artifact.

4.6 Summary

In this Chapter, we tried to answer the following questions:

What choices have been made towards a solution? What constraints or which assumptions have we imposed while designing our artifact?

What artifact have we designed to solve our problem?

How does each of the components of this artifact work and why have we designed it like that?

We have discussed our designed artifact as a solution to the problem, which consists of several components. Figures 4.13 and 4.14 represent the artifact’s architecture and its context.

The artifact is a system that can be split into the following major components:

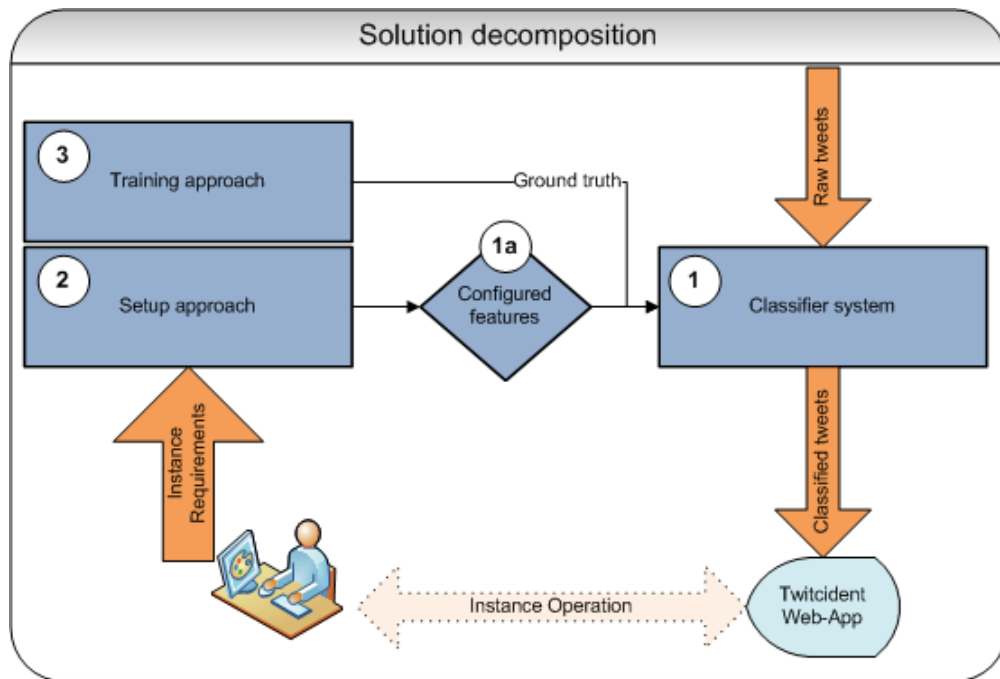


Figure 4.13: Artifact decomposition

1a - Configured features These determine and calculate potentially relevant properties over each document, in our case tweets. Some of these features are Twitcident Instance specific, and need to be configured by the *Setup Approach*.

1 - Classifier The Classifier system is the component that actually classifies the tweet. Based on the pre-calculated vector of feature output values, it attempts to reason over them to classify a tweet as relevant or irrelevant. However, in order to be able to do this, the classifier needs to be trained to ‘learn’ the usability of each feature. This is done by feeding a set of pre-annotated documents (tweets) of which the output value (relevancy) is known a-priori. The *Training Approach* realizes this part.

2 - Setup Approach Some *features* are instance-dependent, and therefore need configuration input such as word lists. These configurations are also prone to error. Therefore, the Setup Approach takes care of this by guiding the user through an interactive approach that consults several *Data Sources*.

3 - Training Approach In order to learn what is relevant, the *Classifier* needs to be trained with a dataset of which the outcomes are known a-priori. The Training Approach is used to compose such a dataset.

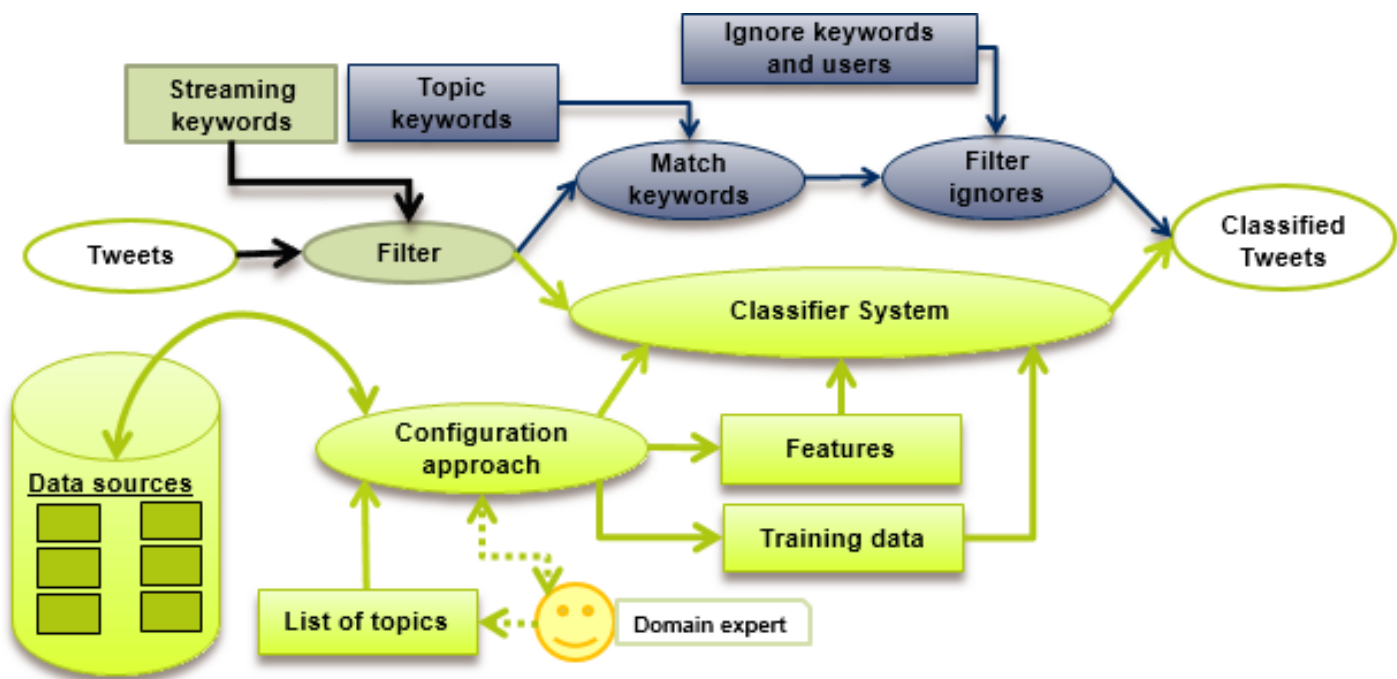


Figure 4.14: Artifact overview: context

Chapter 5

Experiments and Evaluation

Let us recall our main research question:

How can we improve the results of real-time social media filtering with respect to Twitcident?

In Chapter 4 we claimed to have an answer to this question by, and presented our artifact. To validate whether this claim is true, we have to compare our results to the existing Twitcident application in order to verify an actual improvement. We will start by discussing how we collected our ground truth datasets in Section 5.1.1, which is described in more detail in Appendix F. In Section 5.2 we will perform experiments with our designed artifact in order to find a well performing configuration. In Section 5.3 we will compare the performance of our artifact to the existing Twitcident application. Subsequently, this chapter will answer the following questions:

How can we obtain good representative ground truth datasets?

What is a good performing configuration of our artifact? How well does it perform?

Does our artifact actually improve the results with respect to the existing Twitcident application?

5.1 Datasets

5.1.1 Data Collection

To evaluate our Classifier performance, we need training and test data of which the classifications are known a priori. In Appendix F we elaborate on this in detail. In this section, we will give a brief summary of this appendix on how we obtained our ground truth data, and how it is processed. The goal here is to acquire a fully classified representative dataset to evaluate our classifiers.

We have used the existing Twitcident databases to obtain our data, due to the extremely low signal we are trying to detect (Observation 9). These databases contain all tweets that have been streamed, even unclassified ones. Due to the hard problem

domain and data sensitivity, we chose to do the annotation work ourselves within the CrowdSense team, using a developed *Data Annotation application*, where each tweet is annotated by multiple employees to achieve some consensus. In this application, a tweet is annotated on relevancy (irrelevant, relevant, extremely relevant) on both tweet- and topic-level. To construct a representative dataset, we have devised Algorithm 5 (page 147) to generate our annotation batches as balanced as possible.

Next we must process our annotation results to the normalized range 0-100, with 100 being most relevant. Therefore, we have developed a small algorithm that takes into account the facts that a single tweet is always annotated by multiple users, that not every annotating user is as knowledgeable and close to the client, and that each tweet is both annotated on a tweet- and a topic-level. This is described in Section F.4.

In Appendix F we have justified all choices and have elaborated on the Data Annotation in depth. We refer to this appendix for more details, and algorithm specifics.

5.1.2 Resulting datasets

After using this approach we selected all topics that had more than 30 relevant tweets in their dataset, resulting in the following datasets (Table 5.1):

ID	Instance	Total	Relevant
NPL_94	Nationale Politie Limburg Brand (<i>Fire</i>)	600	103 (17.2%)
GG_57	Gaswinning Groningen Aardbeving (<i>Earthquake</i>)	705	110 (15.6%)
NS_19-22	Nederlandse Spoorwegen Storing & Vertraging (<i>Failure & Delay</i>)	629	304 (48.3%)
NS_16	Nederlandse Spoorwegen Zitplaatsen (<i>Seats</i>)	629	35 (5.6%)

Table 5.1: Annotated datasets

Note that the “Storing & Vertraging” dataset has a significantly higher amount of tweets judged relevant. This is due to the fact that tweets from several automatic sources were judged slightly relevant, and these take up a certain proportion of the dataset.

5.1.3 Dataset post-processing

Note that the base datasets all contain normalized numerical classifications. We chose this approach because it maximizes the preservation of information detail, while it is still flexible enough to be converted to other formats.

However, numerical values are only suited for regression testing, but not for binary- or multi-class classification. Therefore, we also construct nominal datasets. The trivial way to do this is to choose boundary values $0.000 < \beta_c < 1.000$ for each nominal class c and assign classes based on this partitioning. For example, if we want a binary classification $\{good, bad\}$ with the boundary at $\beta_0 = 0.4$, we would assign the nominal class ‘bad’ to all tweets with a score lower strictly than 0.4, and ‘good’ to all tweets with a score of 0.4 and higher.

Using this format we constructed several pseudo-datasets for each base dataset that we will experiment with, arbitrarily for exploration purposes. These are listed in Table 5.2:

Postfix ID	Description
_NUM	The original normalized integer class values ranging from 0 to 100.
_NOM_GB_25	A binary classification into $\{good, bad\}$ with $\beta_0 = 0.25$.
_NOM_GB_40	A binary classification into $\{good, bad\}$ with $\beta_0 = 0.40$.
_NOM_GB_60	A binary classification into $\{good, bad\}$ with $\beta_0 = 0.60$.
_NOM_GUB	A multiclass classification into $\{good, useful, bad\}$ with $\beta_0 = 0.35, \beta_1 = 0.55$.
_NOM_C15	A multiclass classification with five class levels $\{C1, C2, C3, C4, C5\}$ with $\beta_0 = 0.20, \beta_1 = 0.40, \beta_2 = 0.60, \beta_3 = 0.80$.

Table 5.2: Nominal classes

5.2 Experiments

In Chapter 4 we have designed our artifact: a solution consisting of multiple components. One of these components was the classifier component; which role can be carried out by a multitude of classifiers. Several questions arise about this component:

- What classifiers and configurations perform well on our problem?
- How do numerical and the different nominal classification schemes compare to each other?
- How does the amount of enabled features affect the classification performance?

By performing initial experiments we intend to answer these questions in this section. Using these answers, we can then compare the performance of our artifact to the existing Twitcident performance.

5.2.1 Suitable classifiers

We created our test instances as listed in Table 5.1 using the Setup Approach (Section 4.4), and ran our Java implementation for calculating the feature values and creating the training sets. We then used the WEKA Experimenter tool to produce results for each classifier in the package. We performed these experiments using 10-fold cross validation with 16 runs (with unique seeding) for each combination of dataset and classifier. We present the averaged results over all datasets for each classifier in Figures 5.1 and 5.2. Although we performed the experiments with all available classifiers and varying options, we only list the top 12 performing configurations.

Figure 5.1 lists the results for all numerical classifiers using regression. We have color-coded the results relative to the other classifiers (vertically) in such a way that the for each measure the best value is marked green and the worst is marked red. Note that we have only shown the best performing classifiers, so a red background does not necessarily mean the classifier performs bad but rather that the others perform relatively better.

Classifier	MAE	RMSE	RAE	RRSE	Corr. coeff.
RandomForest	7,341	11,754	49,940	60,379	0,788
RandomForest (fixed tree size: 40)	7,160	11,970	48,729	61,855	0,778
REPTree	7,956	13,325	54,426	69,013	0,722
REPTree (without pruning)	7,909	14,186	54,090	73,272	0,698
GaussianProcesses (normalized)	10,221	14,459	69,159	74,295	0,678
GaussianProcesses (standardized)	10,128	14,696	68,704	75,622	0,674
MSRules	7,908	13,044	54,163	67,684	0,740
MSRules (unsmoothed)	7,909	13,255	54,261	69,021	0,735
Kstar (globalblend: 20)	7,934	14,675	53,871	75,220	0,659
Kstar (globalblend: 75)	7,804	12,863	52,659	65,846	0,745

MAE: Mean Absolute Error
 RMSE: Root Mean Squared Error
 RAE: Relative absolute error
 RRSE: Relative Root Squared Error

Figure 5.1: Classifier comparison for numerical output

From this figure, we observe that the *Random Forest* classifier performs best with respect to all evaluation measures listed. Interestingly enough, *Neural Networks* (evaluated with various network configurations) and *Support Vector Machines*, two very common types of classifiers are not present in the top performing classifiers.

Classifier	Kappa	TP	FP	TN	FN	Acc	Prec	Rec	F_1	F_2	Mtth	ROC	PRC
Naive Bayes	0.583	86.0	30.6	472.2	52.0	0.872	0.687	0.679	0.669	0.671	0.597	0.915	0.742
Logistic	0.647	96.7	26.5	476.3	41.3	0.895	0.764	0.697	0.728	0.709	0.656	0.906	0.798
Simple Logistic (using AIC)	0.666	92.4	21.1	481.6	45.6	0.896	0.815	0.689	0.745	0.710	0.678	0.930	0.850
SMO (Sup. Vec. Machine) (with standardization)	0.659	96.1	23.9	478.9	41.9	0.898	0.791	0.697	0.740	0.713	0.669	0.820	0.635
JRip (Ripper)	0.726	113.6	33.2	469.5	24.4	0.911	0.765	0.829	0.795	0.815	0.733	0.881	0.718
PART (unpruned, reduced error pruning)	0.668	105.1	35.7	467.0	32.9	0.894	0.748	0.759	0.753	0.757	0.675	0.842	0.650
LMT (using AIC)	0.702	105.5	26.8	475.9	32.5	0.908	0.798	0.755	0.775	0.763	0.710	0.932	0.842
RandomForest (using 100 trees)	0.747	109.2	22.7	480.1	28.8	0.920	0.828	0.792	0.809	0.799	0.753	0.960	0.899
RandomForest (using 200 trees)	0.748	109.0	22.4	480.4	29.0	0.921	0.831	0.792	0.810	0.799	0.754	0.961	0.901
RandomForest (using 200 trees of fixed size 40)	0.741	109.6	24.3	478.5	28.4	0.918	0.810	0.800	0.804	0.801	0.747	0.957	0.892
REPTree	0.706	109.0	30.0	472.7	29.0	0.909	0.770	0.791	0.780	0.786	0.715	0.909	0.756
REPTree + Bagging	0.729	110.3	26.1	476.6	27.7	0.917	0.802	0.788	0.795	0.791	0.736	0.952	0.879

Kappa: Kappa statistic
 TP: Average number of true positives
 FP: Average number of false positives
 TN: Average number of true negatives
 FN: Average number of false negatives
 Acc: Accuracy measure
 Prec: Precision measure
 Rec: Recall measure
 F_1: F1-measure
 F_2: F2-measure
 Mtth: Matthews correlation
 ROC: Area under ROC-curve
 PRC: Area under PRC-curve

Figure 5.2: Classifier comparison for nominal output

Figure 5.2 similarly lists the results for the nominal classifiers, with the corresponding set of evaluation measures. Again, *Random Forest* is one of the best candidates, seemingly followed by *REPTree + Bagging* and *JRip* (the WEKA name; more commonly known as *Ripper*). However, if we remove all other classifier candidates and zoom in (Figure 5.3) we can clearly see that *REPTree* does not outperform the other two on any of the measures.

Classifier	Kappa	TP	FP	TN	FN	Acc	Prec	Rec	F_1	F_2	Mtth	ROC	PRC
JRip (Ripper)	0.726	113.6	33.2	469.5	24.4	0.911	0.765	0.829	0.795	0.815	0.733	0.881	0.718
RandomForest (using 200 trees of fixed size 40)	0.741	109.6	24.3	478.5	28.4	0.918	0.810	0.800	0.804	0.801	0.747	0.957	0.892
REPTree + Bagging	0.729	110.3	26.1	476.6	27.7	0.917	0.802	0.788	0.795	0.791	0.736	0.952	0.879

Figure 5.3: Classifier comparison between top three

This leaves us with *Random Forest* and *JRip* as best candidates. Although *Random Forest* performs better on most measures, we may argue *JRip* to be the better candidate. The aim of this thesis is to filter out noise, which is equivalent to ruling out false positives and thus increasing precision. However, considering our problem carefully, we note that we want to *detect* a low signal of tweets relevant to the operator and missing tweets is more devastating than having some noise left. Therefore, it is essential to keep the amount of false negatives as low as possible, and thus maintaining a recall as high as possible. This is why we also list the F_2 -measure, which values recall twice as much as precision. Re-evaluating the results with this in mind, leads us to observe that *JRip* performs better with respect to reducing false negatives (consequently leading to a higher recall and better F_2 -measure). Therefore, with *Random Forest* leading on accuracy and precision and *JRip* leading with respect to recall, we will not rule out this candidate and consider both classifiers in subsequent experiments.

5.2.2 Output format

In Section 5.1.3 we described how we produce datasets with nominal output classes from the numerically annotated dataset, and mentioned multiple schemes to do so. We are interested in how these schemes compare to each other, and which one to use in subsequent experiments and evaluation.

For each of the test instances, we produced five datasets corresponding to the nominal class schemes described. For each of these combinations, we applied two classifiers (*JRip* and *Random Forest*) using the WEKA Experimenter tool and applied 10-fold cross validation. We repeated this for 16 runs, and averaged the results over these 16 runs. After observing that these values did not show considerable differences, we then averaged the results of both classifiers to a single value for each measure. The results are listed in Figure 5.4.

Key_Dataset	Kappa	TP	FP	TN	FN	Acc	Prec	Rec	F_1	F_2
GG_57_NOM_GB_25	0.651	81.7	36.4	556.6	30.3	0.905	0.696	0.729	0.710	0.721
GG_57_NOM_GB_40	0.647	78.8	34.8	560.3	31.2	0.906	0.699	0.716	0.705	0.711
GG_57_NOM_GB_60	0.453	13.6	11.0	664.1	16.4	0.961	0.563	0.453	0.501	0.471
GG_57_NOM_GUB	0.551	18.4	10.5	661.5	14.7	0.964	0.634	0.556	0.593	0.570
NPL_94_NOM_GB_25	0.800	89.4	19.7	475.3	15.6	0.941	0.821	0.851	0.835	0.845
NPL_94_NOM_GB_40	0.794	87.0	19.4	477.7	16.1	0.941	0.819	0.844	0.831	0.839
NPL_94_NOM_GB_60	0.358	6.1	6.0	579.0	8.9	0.975	0.506	0.407	0.439	0.417
NPL_94_NOM_GUB	0.731	7.2	6.2	577.9	8.8	0.975	0.579	0.450	0.492	0.463
NS_16_NOM_GB_25	0.838	30.9	7.3	586.8	4.2	0.982	0.810	0.881	0.844	0.866
NS_16_NOM_GB_40	0.838	30.9	7.3	586.8	4.2	0.982	0.810	0.881	0.844	0.866
NS_16_NOM_GB_60	0.799	24.2	5.0	594.1	5.8	0.983	0.831	0.807	0.818	0.811
NS_16_NOM_GUB	0.744	24.1	7.4	591.7	5.9	0.979	0.767	0.803	0.784	0.796
NS_19-22_NOM_GB_25	0.678	286.5	52.2	241.9	48.6	0.840	0.846	0.855	0.851	0.853
NS_19-22_NOM_GB_40	0.666	253.6	54.5	270.5	50.4	0.833	0.824	0.834	0.829	0.832
NS_19-22_NOM_GB_60	0.161	6.3	9.2	575.8	37.7	0.925	0.411	0.143	0.212	0.164
NS_19-22_NOM_GUB	0.574	15.8	17.8	556.2	39.2	0.909	0.463	0.287	0.353	0.310

Kappa: Kappa statistic
 TP: Average number of true positives
 FP: Average number of false positives
 TN: Average number of true negatives
 FN: Average number of false negatives
 Acc: Accuracy measure
 Prec: Precision measure
 Rec: Recall measure
 F_1: F1-measure
 F_2: F2-measure

Figure 5.4: Comparison of nominal class output format

First of all, note that the measures output only apply to a single class against all other classes. WEKA automatically only calculates these measures for the first class

against the remainder classes. Therefore, we put our most relevant class (‘good’) first, since we value the precision and recall of this class over the other nominal classes.

This is also the reason why the “_NOM_C15” entry (the one with five nominal classes) is not listed: the ‘best’ class often had next to no positives in it, resulting in divisions by zero or skewed results. After manually analyzing its output using WEKA Explorer, we can conclude this scheme performs far worse than the listed classes. As we can see in Figure 5.4, the “_NOM_GUB” multi-class scheme (with three classes) also performs worse than the other three binary class schemes.

We can also observe that over all, the “_NOM_GB_25” and “_NOM_GB_40” schemes generally perform best, but not with respect to the accuracy measure. This can be explained by the large ratio of negatives: the “bad”-class is considerably more populated than the “good”-class. Furthermore, after manually examining the datasets, we can conclude that generally the “_NOM_GB_40” scheme with a β -boundary of around 30 – 40 best represents the desired output with respect to how the instance was configured, and this is also being represented by the classifier performance. Still, we should note that this highly depends (and correlates) with which intent the instance was constructed during the Setup Approach.

Recall that our initial focus was on constructing a classification scheme that outputs a normalized value in the range [0.000 – 1.000], so we could implement a threshold value as a configuration option (slider) in the application (Twitcident). However, the nominal classifiers seem to perform much better than the numerical classifiers. As an alternative, we propose to train multiple binary classifiers with different β -boundary values, and let the configuration option select the classifier based with the corresponding boundary value.

5.2.3 Amount of features

Another question that arises is: How does the amount of enabled features affect the classification performance? We have a long list of varying features. We could imagine that adding additional (potentially irrelevant) features may reduce classification quality due to limited training data, but also that removing some features may also reduce classification quality. In Section 4.3.1 we proposed all features. We will perform experiments with different subsets of features as listed in Table 5.3.

Subset ID	Enabled features
48F	1, 2, 3, 4, 5, 6, 10, 11, 12, 13, 15, 16, 19, 20, 21, 22, 23, 24, 28, 29, 30, 31, 32, 33, 36, 42, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 56, 59, 60, 61, 62, 63, 64, 65, 67, 68, 69, 70
33F	1, 2, 4, 5, 10, 11, 31, 32, 33, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 56, 59, 60, 61, 62, 63, 64, 65, 67, 68, 69, 70
26F	1, 2, 4, 5, 10, 11, 31, 32, 33, 44, 45, 46, 47, 49, 50, 51, 52, 53, 62, 63, 64, 65, 67, 68

Table 5.3: Subsets of enabled features

We consider the 48F-subset to be our full set as it includes all features that we have currently implemented. For the 33F-subset we disabled some features that we do not expect to contribute, for example features matching our keywords against usernames

and user descriptions. We consider the 26F-subset to be the minimal subset of essential features, of which we are quite confident they are required.

For each of these subsets, we applied nominal as well as numerical classification using the WEKA Experimenter tool and applied 10-fold cross validation. We repeated this for 16 runs, and averaged the results over these 16 runs. The results are listed in Figure 5.5 for numerical classification and Figure 5.6 for nominal classification.

Dataset	#Feat	MAE	RMSE	RAE	RRSE	Corr. Coeff.
GG_57_NUM	48	7.52	13.38	51.79	66.60	0.745
GG_57_NUM	33	7.42	13.34	51.14	66.46	0.745
GG_57_NUM	26	7.70	13.97	53.01	69.62	0.717
NPL_94_NUM	48	5.68	9.72	41.29	53.49	0.848
NPL_94_NUM	33	5.62	9.64	40.87	53.11	0.849
NPL_94_NUM	26	5.78	10.30	42.04	56.75	0.820
NS_16_NUM	48	5.18	8.15	55.30	52.56	0.842
NS_16_NUM	33	5.16	8.13	55.20	52.76	0.837
NS_16_NUM	26	5.22	8.48	55.92	55.15	0.820
NS_19-22_NUM	48	10.97	15.69	51.45	68.39	0.725
NS_19-22_NUM	33	11.19	15.98	52.48	69.67	0.713
NS_19-22_NUM	26	11.31	16.40	53.07	71.48	0.696

#Feat: Number of enabled features RAE: Relative absolute error
 MAE: Mean Absolute Error RRSE: Relative Root Squared Error
 RMSE: Root Mean Squared Error Corr. Coeff.: Correlation Coefficient

Figure 5.5: Comparison of amount of enabled features for numerical classification

Dataset	#Feat	Kappa	TP	FP	TN	FN	Acc	Prec	Rec	F_1	F_2	Mtth	ROC	PRC
GG_57_NOM_GB_40	48	0.647	78.8	34.8	560.3	31.2	0.906	0.699	0.716	0.705	0.711	0.655	0.889	0.678
GG_57_NOM_GB_40	33	0.642	77.4	33.9	561.1	32.7	0.906	0.699	0.703	0.700	0.701	0.649	0.883	0.674
GG_57_NOM_GB_40	26	0.603	72.0	35.2	559.9	38.1	0.896	0.672	0.654	0.663	0.658	0.610	0.858	0.648
NPL_94_NOM_GB_40	48	0.794	87.0	19.4	477.7	16.1	0.941	0.819	0.844	0.831	0.839	0.801	0.942	0.820
NPL_94_NOM_GB_40	33	0.795	87.6	20.0	477.1	15.5	0.941	0.819	0.850	0.832	0.843	0.801	0.943	0.820
NPL_94_NOM_GB_40	26	0.800	87.8	19.7	477.4	15.2	0.941	0.817	0.852	0.834	0.845	0.805	0.943	0.809
NS_16_NOM_GB_40	48	0.838	30.9	7.3	586.8	4.2	0.982	0.810	0.881	0.844	0.866	0.847	0.971	0.868
NS_16_NOM_GB_40	33	0.837	30.7	7.3	586.8	4.4	0.982	0.809	0.876	0.841	0.861	0.845	0.970	0.866
NS_16_NOM_GB_40	26	0.839	30.8	7.2	586.9	4.3	0.982	0.812	0.879	0.843	0.864	0.847	0.974	0.877
NS_19-22_NOM_GB_40	48	0.666	253.6	54.5	270.5	50.4	0.833	0.824	0.834	0.829	0.832	0.671	0.881	0.860
NS_19-22_NOM_GB_40	33	0.652	248.0	53.2	271.9	56.1	0.826	0.824	0.816	0.820	0.817	0.656	0.875	0.854
NS_19-22_NOM_GB_40	26	0.636	247.5	57.9	267.2	56.6	0.818	0.811	0.814	0.812	0.813	0.640	0.869	0.848

#Feat: Number of features enabled Prec: Precision measure
 Kappa: Kappa statistic Rec: Recall measure
 TP: Average number of true positives F_1: F1-measure
 FP: Average number of false positives F_2: F2-measure
 TN: Average number of true negatives Mtth: Matthews correlation
 FN: Average number of false negatives ROC: Area under ROC-curve
 Acc: Accuracy measure PRC: Area under PRC-curve

Figure 5.6: Comparison of amount of enabled features for nominal classification

The results reflect our expectations: the 26-feature subset of essential features is always outperformed by the larger subsets that may have additional relevant features. The 33-feature and 48-feature subsets are always very close to each other. Interestingly, for the numerical classifiers the 33-feature subset generally performs slightly better, while for nominal classifiers the 48-feature subset generally performs slightly better. This leads us to hypothesize that the nominal classifiers are better capable of handling additional low impact features. Ofcourse, this needs to be statistically verified by more extensive experiments before we can assume it.

5.2.4 Classifier performance

Dataset	#Feat	Classifier	Kappa statistic	True pos.	False pos.	True neg.	False neg.	Accuracy	Precision	Recall	F1-measure	F2-measure	
GG_57	48	JRip (Ripper)	0,634	82,3	43,6	551,4	27,7	0,899	0,654	0,748	0,698	0,727	
GG_57	33		0,628	80,0	41,3	553,7	30,0	0,899	0,659	0,727	0,692	0,713	
NPL_94	48		0,774	86,7	23,3	473,8	16,3	0,934	0,789	0,842	0,814	0,830	
NPL_94	33		0,781	87,9	23,4	473,6	15,1	0,936	0,789	0,853	0,820	0,840	
NS_16	48		0,841	31,4	7,8	586,2	3,6	0,982	0,801	0,898	0,847	0,877	
NS_16	33		0,841	31,1	7,5	586,5	3,9	0,982	0,806	0,888	0,845	0,870	
NS_19-22	48		0,662	258,7	60,9	264,1	45,3	0,831	0,810	0,851	0,830	0,842	
NS_19-22	33		0,647	250,9	57,8	267,2	53,1	0,824	0,813	0,825	0,819	0,823	
GG_57	48		Random Forest	0,651	74,1	26,4	568,6	35,9	0,912	0,738	0,674	0,704	0,686
GG_57	33			0,653	74,6	26,8	568,3	35,4	0,912	0,736	0,678	0,706	0,689
NPL_94	48	0,816		87,1	15,3	481,7	15,9	0,948	0,851	0,846	0,848	0,847	
NPL_94	33	0,817		87,8	16,0	481,0	15,2	0,948	0,846	0,853	0,849	0,851	
NS_16	48	0,835		30,3	6,6	587,4	4,8	0,982	0,820	0,864	0,842	0,855	
NS_16	33	0,832		30,1	6,7	587,3	4,9	0,982	0,818	0,861	0,839	0,852	
NS_19-22	48	0,671		248,3	47,5	277,5	55,7	0,836	0,839	0,817	0,828	0,821	
NS_19-22	33	0,654		244,5	49,1	275,9	59,5	0,827	0,833	0,804	0,818	0,810	

Figure 5.7: Final performance measures for our artifact

Figure 5.7 lists our final performance measures, using the (roughly) best performing configurations we have determined in previous sections. We notice both classifiers to perform similarly well, with *JRip* being slightly better at preventing false negatives, and thus increasing recall and F_2 -score, while the *Random Forest* classifier performs better on precision thus displaying less noise, at the cost of missing a few relevant tweets.

Although the accuracy scores are very high, this is also largely due to the huge amount of negatives in the dataset, and thus can be attributed to the low signal. However, considering the difficulty of our problem (Section 2.5), the current absence of several important features¹ and the low consensus among data annotators, we can consider the performance of our artifact with an F_2 -score of 0.7 up to 0.9 very satisfying and future results (with all features implemented) even more promising.

5.3 Results comparison

In Section 5.2 we have performed experiments to find the current most suitable classifier and configuration, and measured its performance. In this section, we will bridge back to our thesis objective. Recall our research question:

How can we improve the results of real-time social media filtering with respect to Twitcident?

In Chapter 4 we claimed to have an answer to this question, and presented an artifact. To evaluate whether this claim is true, we have to compare our results to the existing Twitcident application in order to verify an actual improvement.

Therefore, we will repeat the experiments for the existing Twitcident application and measure its performance. We will then compare these results to our presented

¹For example, the ‘sentiment’, ‘subjectivity’, ‘seems observation’, ‘present and past tense’, ‘poster tweet rates’, ‘slang’ and ‘popular term’ features were all not implemented (and thus not enabled) yet during these experiments

artifact. Because Twitcident is a binary classifier, i.e. it displays a tweet or not, we can only compare the results to the nominal output scheme.

Consider Θ^+ to be the set of tweets displayed by the existing Twitcident application, $score_t$ the actual ground truth score of tweet t in our annotated dataset and a boundary value β (similar to the one described in Section 5.1.1); we can then calculate the amount of true/false positives/negatives using:

$$\begin{aligned} \text{True Positives (TP)} &= \sum_t \begin{cases} 1 & \text{if } score_t \geq \beta \text{ and } t \in \Theta^+ \\ 0 & \text{otherwise} \end{cases} \\ \text{False Positives (FP)} &= \sum_t \begin{cases} 1 & \text{if } score_t < \beta \text{ and } t \in \Theta^+ \\ 0 & \text{otherwise} \end{cases} \\ \text{True Negatives (TN)} &= \sum_t \begin{cases} 1 & \text{if } score_t < \beta \text{ and } t \notin \Theta^+ \\ 0 & \text{otherwise} \end{cases} \\ \text{False Negatives (FN)} &= \sum_t \begin{cases} 1 & \text{if } score_t \geq \beta \text{ and } t \notin \Theta^+ \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

In Section 5.2.2 we have seen that a boundary value $\beta = 0.4$ (corresponding to the `_NOM_GB_40` output scheme) produces the most accurate results. If we want to compare the classification performance of our artifact using this scheme, we should ofcourse compare it to Twitcident's results using the same boundary value. However, we will first check whether this value also produces optimal results for Twitcident on its own.

Dataset	β	TP	FN	FP	TN	Accuracy	Precision	Recall	F1-measure	F2-measure
GG_57	0,25	55	57	34	559	0,871	0,618	0,491	0,547	0,512
NPL_94		99	5	90	399	0,840	0,524	0,952	0,676	0,818
NS_16		32	3	22	570	0,960	0,593	0,914	0,719	0,825
NS_19-22		108	227	30	262	0,590	0,783	0,322	0,457	0,365
GG_57	0,40	53	57	36	559	0,868	0,596	0,482	0,533	0,501
NPL_94		97	5	92	399	0,836	0,513	0,951	0,667	0,812
NS_16		32	3	22	570	0,960	0,593	0,914	0,719	0,825
NS_19-22		107	197	31	292	0,636	0,775	0,352	0,484	0,395
GG_57	0,60	24	6	65	610	0,899	0,270	0,800	0,403	0,574
NPL_94		15	0	174	404	0,707	0,079	1,000	0,147	0,301
NS_16		27	3	27	570	0,952	0,500	0,900	0,643	0,776
NS_19-22		23	21	115	468	0,783	0,167	0,523	0,253	0,366

Figure 5.8: Comparison of existing Twitcident performance for various boundary values

Figure 5.8 shows the performance metrics of the existing Twitcident application for various boundary values. Again we can see an improved accuracy as the amount of positives decreases, which can be attributed to the high proportion of negatives in the datasets. With respect to all other performance metrics, Twitcident seems to be most accurate with $\beta = 0.4$, which is consistent with earlier findings. Therefore, we will compare our designed artifact to Twitcident using $\beta = 0.4$.

Figure 5.9 present the comparison of performance between the designed artifact presented in this thesis and the existing Twitcident application. Although the existing Twitcident application in some cases has a higher recall, we can see that our designed artifact consistently outperforms it when we take precision into account within the F-scores. Sometimes there is just a slight increase in performance (NPL_94 and

Dataset	System	Accuracy	Precision	Recall	F1-score	F2-score
GG_57	Existing Twitcident application	0,868	0,596	0,482	0,533	0,501
	Designed artifact using JRip (Ripper)	0,899	0,654	0,748	0,698	0,727
	Designed artifact using Random Forest	0,912	0,736	0,678	0,706	0,689
NPL_94	Existing Twitcident application	0,836	0,513	0,951	0,667	0,812
	Designed artifact using JRip (Ripper)	0,936	0,789	0,853	0,820	0,840
	Designed artifact using Random Forest	0,948	0,846	0,853	0,849	0,851
NS_16	Existing Twitcident application	0,960	0,593	0,914	0,719	0,825
	Designed artifact using JRip (Ripper)	0,982	0,801	0,898	0,847	0,877
	Designed artifact using Random Forest	0,982	0,820	0,864	0,842	0,855
NS_19-22	Existing Twitcident application	0,636	0,775	0,352	0,484	0,395
	Designed artifact using JRip (Ripper)	0,831	0,810	0,851	0,830	0,842
	Designed artifact using Random Forest	0,836	0,839	0,817	0,828	0,821
Average for existing Twitcident application		0,825	0,619	0,675	0,601	0,633
Average for designed artifact using JRip (Ripper)		0,912	0,764	0,838	0,799	0,822
Average for designed artifact using Random Forest		0,920	0,810	0,803	0,806	0,804

Figure 5.9: Comparison of performance between our designed artifact and existing Twitcident application

NS_16 datasets), while sometimes the performance increase is remarkable (GG_57 and NS_19-22 datasets). On average, the F_2 -score is improved by 30%. In accordance with these results, we can confirm that our designed artifact indeed improves the filtering mechanism and reduces noise while maintaining a high recall.

Chapter 6

Conclusions and Future Work

We will conclude our work with this final chapter. We will summarize our thesis work in Section 6.1 which answers the primary and all secondary research questions stated in our introduction (Section 1.5). With our work summarized, how will it go from there? We will address this by discussing future work to be done in Section 6.2. We will discuss entry-points for future research and define potential areas for improvement to follow up our work. Finally, we will list our contributions in Section 6.3.

6.1 Summary

Since the commencement of Web 2.0, people are able to contribute to the Web through social services like Twitter. Such massive amounts of user generated content contains a lot of valuable information for various interested parties. This enormous flow of information contains vital information which cannot be processed adequately by default means.

To process and filter this information, CrowdSense developed Twitcident. This technology is able to monitor a specific set of topics, events or areas in real-time, providing the monitoring client with potentially valuable information. It basically exploits humans as sensors, filters the signal coming from all those sensors and attempts to extract the relevant meaningful information. An analysis of the relevant information by an operator can result in an estimation of severity, and an operator can act accordingly.

However, among all relevant and useful content that is extracted, also a lot of irrelevant noise is present. Our goal is to improve the filter in such a way that the majority of information presented by Twitcident is relevant. In order to achieve this, we must reduce the noise among presented tweets, resulting in the following research question:

How can we improve the results of real-time social media filtering with respect to Twitcident?

The last part of this question “with respect to Twitcident” imposes some additional constraints on the problem, like *real-time* filtering, dynamic word-lists as a result of a dynamic vague domain, multiple instances with limited setup time and Dutch tweets as primary target (Section 2.5).

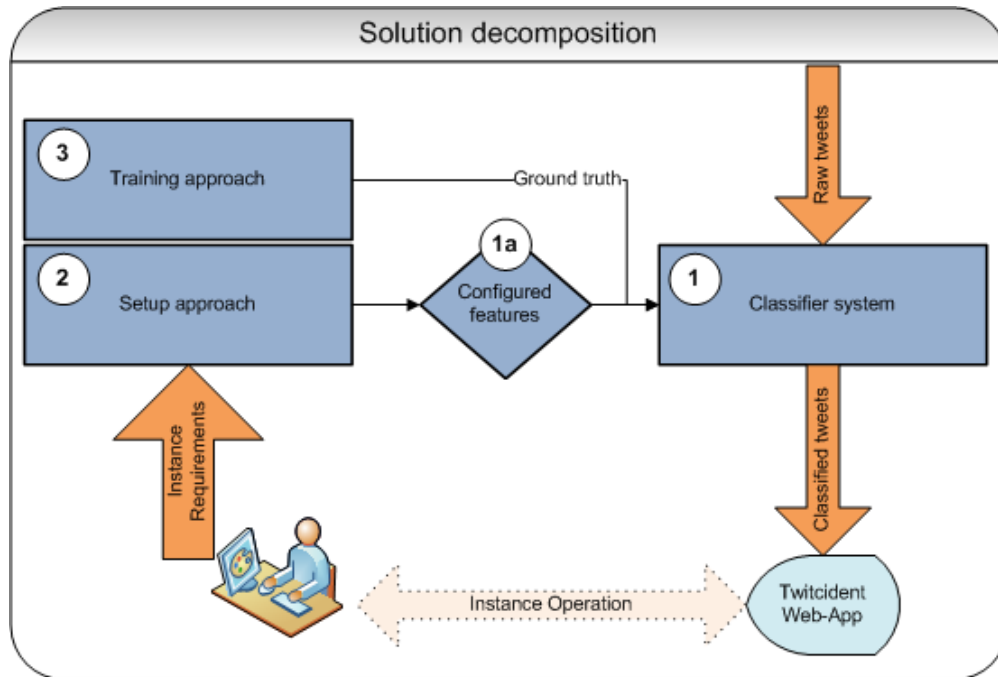


Figure 6.1: Artifact decomposition

In Chapter 2 we elaborate more on the Twitcident application (Section 2.1) and perform a problem analysis to identify where the noise originates from and what factors may attribute to relevancy by observing real life instances. For example, we observe the output quality is largely determined by the quality of the keyword combinations defined, a very low signal of relevant tweets, the importance of synonyms and hyponyms, misspellings, verbal tense, user types, and so on. The full list of observations can be found in Section 2.6.

In order to solve this problem we apply the Design Science methodology (Section 3.1) and design an artifact that improves on the filtering mechanism (Chapter 4). The artifact is a system composed of several major components, as illustrated by Figure 6.1.

Although our artifact is designed to improve Twitcident, its application is not limited to it but can be generalized to any real-time filtering system under the same constraints (i.e. assuming in incoming stream of tweets).

Assuming an incoming stream of raw tweets, the Classifier component (marked 1 in Figure 6.1) calculates the relevancy score of each tweet, which is the output of our artifact. Twitcident can then use this score and display relevant tweets accordingly. This Classifier component is composed of a two phase process for each single tweet. During the first phase the score for each feature is calculated individually by our developed FeatureCalculator. After all scores are known, the feature vector containing all values for the tweet is fed to a Machine Learning classifier. In Section 5.2 we have seen that the JRip (Ripper) and Random Forest classifiers of the WEKA Machine Learning package are very suitable for this task.

The features (marked 1a in Figure 6.1) are an essential part of the Classifier com-

ponent. These are basically the attributes of a tweet that may have an impact on the relevancy of a tweet, upon which the machine learning classifier will learn to reason. Examples of such features include:

- Does the tweet contain keyword combination X and Y while keyword Z is not allowed to be present?
- What is the geographical distance to the closest identified point of interest (hotspot)?
- Is the tweet posted by news media or not?
- Is the tweet written in present or past tense?
- Does the tweet have a positive or negative sentimental value?
- How many similar tweets to the current tweet were posted during the last hour?
- How many nouns/verbs/adverbs does the tweet contain?
- Is a future or past day mentioned?

We have composed an extensive list of 70 features based on our list of observations and our literature study. The full list and their descriptions can be found in Appendix A. We have implemented our Classifier system in Java in such a way that it is fully configurable through a middle-ware database: settings can be configured, features can be turned on or off and their parameters can be tuned. Specifics can be found in Appendix C.

As part of one of the features we also designed an approximate string matching algorithm by adapting Dice's similarity metric, that suited our problem better than other algorithms in this field. This algorithm is presented in Appendix B.

Some features are instance-dependent, and therefore need configuration input such as word lists. These configurations are also prone to error. Therefore, in order to setup such a configuration of features and their parameters for a Twitcident client, we have designed the Setup Approach (marked 2 in Figure 6.1). This is an interactive process that needs to be executed a single time for each topic to be classified and can be performed by a CrowdSense employee as well as a domain expert (client operator). Besides guiding the user, the Setup Approach also aims at reducing setup errors, performs some intensive tasks automatically (data collection, word expansion, combination generation, ...) and helps the user think of scenarios by collecting data from a lot of data sources (Appendix D). The Setup Approach is explained in detail in Sections 4.4 and C.3.

Machine Learning classifiers need to be 'trained' on data of which the outcomes (relevancy) are known a-priori in order to learn how to interpret each feature with respect to the outcome. The Training Approach (marked 3 in Figure 6.1) constitutes this part of our design, and is used to compose a dataset for the classifier component to train on. It is a process directly preceded by the Setup Approach. We have only briefly discussed our ideas on this component (Section 4.5) and considered its concrete implementation outside the scope of this thesis work, as it is another quite complex component requiring research and evaluation on its own.

In order to evaluate our artifact, we have designed an algorithm to collect and sample our ground truth dataset (Sections F.3 and F.4). The aim of this algorithm is to find a suitable proportion of relevant tweets (due to the low signal) and create balanced representative datasets. We also developed and designed a Data Annotation application (Section F.2) to have domain experts annotate the collected datasets to obtain our ground truth data. We have justified our choices with respect to data collection in Section F.1.

The data collection resulted in four datasets (Section 5.1.2) with several different output class formats: one with a numerical class and several nominal class schemes (Section 5.1.3). We have evaluated the performance of our artifact using these datasets (Chapter 5). We have performed initial experiments to find a suitable configuration for our artifact (Section 5.2).

We first evaluated which Machine Learning classifier in the WEKA package was most suitable to constitute the classifier component. We can conclude that the JRip (Ripper) and Random Forest classifiers performed best for the nominal classification schemes, while the Random Forest classifier also performed best on the numerical classification scheme. Second, we have evaluated which nominal classification output scheme best represents the client’s request intentions, and observed that binary classification with a specific threshold performed best. Finally, we have varied the amount of features and observed which impact this had on the performance. We can conclude that adding low impact features did never reduce classification performance, and that increasing the amount of features always increased performance: sometimes slightly, sometimes up to 10%.

Dataset	System	Accuracy	Precision	Recall	F1-score	F2-score
GG_57	Existing Twitcident application	0,868	0,596	0,482	0,533	0,501
	Designed artifact using JRip (Ripper)	0,899	0,654	0,748	0,698	0,727
	Designed artifact using Random Forest	0,912	0,736	0,678	0,706	0,689
NPL_94	Existing Twitcident application	0,836	0,513	0,951	0,667	0,812
	Designed artifact using JRip (Ripper)	0,936	0,789	0,853	0,820	0,840
	Designed artifact using Random Forest	0,948	0,846	0,853	0,849	0,851
NS_16	Existing Twitcident application	0,960	0,593	0,914	0,719	0,825
	Designed artifact using JRip (Ripper)	0,982	0,801	0,898	0,847	0,877
	Designed artifact using Random Forest	0,982	0,820	0,864	0,842	0,855
NS_19-22	Existing Twitcident application	0,636	0,775	0,352	0,484	0,395
	Designed artifact using JRip (Ripper)	0,831	0,810	0,851	0,830	0,842
	Designed artifact using Random Forest	0,836	0,839	0,817	0,828	0,821
Average for existing Twitcident application		0,825	0,619	0,675	0,601	0,633
Average for designed artifact using JRip (Ripper)		0,912	0,764	0,838	0,799	0,822
Average for designed artifact using Random Forest		0,920	0,810	0,803	0,806	0,804

Figure 6.2: Comparison of performance between our designed artifact and existing Twitcident application

In Chapter 4 we claimed to have an answer with respect to our research question, and presented an artifact. To evaluate whether this claim is true we compared our results to the existing Twitcident application in order to verify an actual improvement (Section 5.3). Figure 6.2 lists the final results. Although the existing Twitcident application in some cases has a higher recall, we can see that our designed artifact consistently outperforms it when we take precision into account within the F-scores. Sometimes there is just a slight increase in performance (NPL_94 and

NS_16 datasets), while sometimes the performance increase is remarkable (GG_57 and NS_19-22 datasets). On average, the F_2 -score is improved by 30%. In accordance with these results, we can confirm that our designed artifact indeed improves the filtering mechanism and reduces noise while maintaining a high recall.

Although the accuracy scores are very high, this is also largely due to the huge amount of negatives in the dataset, and thus can be attributed to the low signal. However, considering the difficulty of our problem (Section 2.5) and the current absence of several important features¹, we can consider the performance of our artifact with an F_2 -score of 0.7 up to 0.9 very satisfying and future results (with all features implemented) even more promising.

6.2 Future work

With our work we not only improved on the filtering and noise reduction mechanism, but we also set up a robust framework that allows for many extensions and improvements. In this section we will list and briefly discuss entry-points for future work and research.

Repeating the search-process cycle In Chapter 2 we performed a problem analysis by observing the starting system’s output. Based on these observations we designed a system, and evaluated it. According to Guideline 6 of Design Science Research (Section 3.1), the cycle is now complete and should repeat. By repeating a similar analysis on our new artifact’s output, making note of observations and drawing ideas from it, the presented artifact may be improved.

Complete new feature implementations We have only implemented 48 of the 70 proposed features, due to several constraints. We expect that the unimplemented features will have an additional positive impact on the performance of our system, such as the sentiment, present or past tense and has-popular-term features. Other ideas include profiling of users, and analyzing referenced entities like URL targets and images in more detail. Appendix A lists all features as well as the implementation status.

Feature improvements Some features have a basic implementation or function, and can be improved upon. For example, consider feature 68 which counts the amount of question marks. This could be improved by measuring the question to sentence ratio, with only considering question marks at the end of a sentence. This would involve a sentence detection mechanism for each language, which is available in for example the OpenNLP library. Also the keyword combination generation scheme can be experimented with: are there better performing alternatives still satisfying our constraints?

Study on feature relevancy It would be interesting to have a study on the impact of each feature on the total performance, and how much each feature (combination)

¹For example, the ‘sentiment’, ‘subjectivity’, ‘seems observation’, ‘present and past tense’, ‘poster tweet rates’, ‘slang’ and ‘popular term’ features were all not implemented (and thus not enabled) yet during these experiments

contributes to it. This may involve evaluating all possible subsets of features since there may be inter-feature relations. Consider for example the amount of nouns, verbs, adverbs features. And is there an optimal subset of features that generally performs better than the full set of features?

Study on the dynamism of features It would be interesting to have a study on the impact of changing feature inputs without retraining the classifier. Does the system handle it well when we would add new keywords to the word-lists, add ignore users to lists, add hotspots, and so on. If this is not the case, as an alternative, we may keep records of all training data and retrain the system after changes to its input configuration.

Building an interface for reconfiguration After the previous point has been evaluated (impact of reconfiguration), it would be useful to produce a tool for reconfiguring the system. The Setup Approach now generates a complete configuration which is often too large to manually alter. Consider for example a keyword input of 5 synonyms and 20 combo terms; this would yield $5 \cdot \binom{20}{3} = 5700$ combinations when capped at at most 3 combo terms.

Training Approach The Training Approach has been discussed in a conceptual state, and should be designed in detail. Section 4.5 presents our current ideas as a starting point. How can we effectively train our system? How can we effectively select the tweets for training? Closely related to this is the retraining of the classifier: can we iteratively collect additional data and retrain the system on it? Or apply online learning? As a result, we could implement user feedback loops from the application.

Separate some features as stand-alone filters For some features, it is known a priori what effect it should have. Consider for example the ‘global ignore term’ and ‘ignore user’ features: we could argue these should always prevent a tweet from being judged relevant. According to this reasoning, we could remove the feature from actual classification and use it as a pre- or post-classification filtering step on its own. However, one could also argue that in some cases it should be considered relevant nonetheless, in a potential exceptional case that all other features strongly suggest the tweet is relevant.

Optimizing the artifact configuration We have performed initial experiments with just a rough set of parameters. However, we could also apply another machine learner (genetic programming?) to find the optimal configuration and chaining of components, features and their parameters.

6.3 Contributions

During our work, we have made several contributions. We will conclude this thesis by listing these.

- We have performed a thorough problem analysis resulting in a list of observations. Hereby we have shown that aside from simple string matching a lot of other factors contribute to the relevancy of a tweet.

- We have designed an artifact with utility: it improves the filtering mechanism of Twitcident by reducing noise, it enables clients to configure the system themselves through a user interacted approach and reduces the errors made during this approach.
- The artifact is developed within a robust architecture, enabling for many future improvements and extensions. The existing Twitcident application was limited to keyword and user filtering, whereas our architecture allows for all kinds of future extensions (Section 6.2).
- We have developed a Data Annotation Application including a Data Collection sampling algorithm as well as output processing schemes.
- We have extended the ARFF format with the possibility to add an unbounded amount of meta data to each feature vector, for example to keep track of corresponding IDs to documents. We also implemented an extended interpreter and loader for this format for use with WEKA. This allows for more in-depth analysis of results.
- We have developed a secure SSH-tunneled MySQL database connection handler.
- We have proposed a comprehensive feature list to consider while filtering social media.
- We have coined the idea of having features with dynamic inputs.
- We have designed a very fast approximate string matching algorithm for variable length strings (Appendix B) by adapting Dice's similarity metric.
- We have presented ideas regarding event detection (Appendices G and H).
- We have proposed entry-points and ideas for future work and research to follow up this work.

Bibliography

- [1] Fabian Abel, Claudia Hauff, Geert-Jan Houben, Richard Stronkman, and Ke Tao. Semantics+ filtering+ search= twitcident. exploring information in social web streams. In *Proceedings of the 23rd ACM conference on Hypertext and social media*, pages 285–294. ACM, 2012.
- [2] Fabian Abel, Claudia Hauff, Geert-Jan Houben, Richard Stronkman, and Ke Tao. Twitcident: fighting fire with information from social web streams. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 305–308. ACM, 2012.
- [3] The Association for Computational Linguistics. *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, 2013.
- [4] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.
- [5] Alexandra Balahur, Rada Mihalcea, and Andrés Montoyo. Computational approaches to subjectivity and sentiment analysis: Present and envisaged methods and applications. *Computer Speech & Language*, 28(1):1–6, 2014.
- [6] Hila Becker, Mor Naaman, and Luis Gravano. Beyond trending topics: Real-world event identification on twitter. *ICWSM*, 11:438–441, 2011.
- [7] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [8] Eric Brill and Robert C Moore. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293. Association for Computational Linguistics, 2000.
- [9] Claudio Carpineto and Giovanni Romano. A survey of automatic query expansion in information retrieval. *ACM Computing Surveys (CSUR)*, 44(1):1, 2012.
- [10] Olivier Chapelle, Bernhard Schölkopf, Alexander Zien, et al. *Semi-supervised learning*. Adaptive computation and machine learning. MIT press Cambridge, September 2006.

- [11] Jilin Chen, Allen Cypher, Clemens Drews, and Jeffrey Nichols. Crowde: Filtering tweets for direct customer engagements. In *ICWSM*. Citeseer, 2013.
- [12] Ondřej Chum, Andrej Mikulik, Michal Perdoch, and Jiří Matas. Total recall ii: Query expansion revisited. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 889–896. IEEE, 2011.
- [13] William W Cohen. Fast effective rule induction. In *Proceedings of the twelfth international conference on machine learning*, pages 115–123, 1995.
- [14] Aron Culotta. Towards detecting influenza epidemics by analyzing twitter messages. In *Proceedings of the first workshop on social media analytics*, pages 115–122. ACM, 2010.
- [15] Fred J Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964.
- [16] Peter J Denning. A new social contract for research. *Communications of the ACM*, 40(2):132–134, 1997.
- [17] Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.
- [18] Paul Earle, Michelle Guy, Richard Buckmaster, Chris Ostrum, Scott Horvath, and Amy Vaughan. Omg earthquake! can twitter improve earthquake response? *Seismological Research Letters*, 81(2):246–251, 2010.
- [19] Óscar Fontenla-Romero, Bertha Guijarro-Berdiñas, David Martínez-Rego, Beatriz Pérez-Sánchez, and Diego Peteiro-Barral. Online machine learning. *Efficiency and Scalability Methods for Computational Intellect*, page 27, 2013.
- [20] William B Frakes. Stemming algorithms., 1992.
- [21] Johannes Fürnkranz and Gerhard Widmer. Incremental reduced error pruning. In *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 70–77, 1994.
- [22] Kevin Gimpel, Nathan Schneider, Brendan O’Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics, 2011.
- [23] Roberto González-Ibáñez, Smaranda Muresan, and Nina Wacholder. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 581–586. Association for Computational Linguistics, 2011.
- [24] Erik Hatcher, Otis Gospodnetic, and Michael McCandless. Lucene in action, 2004.

- [25] Alan R Hevner, Salvatore T March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS quarterly*, 28(1):75–105, 2004.
- [26] Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 56–65. ACM, 2007.
- [27] Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 151–160. Association for Computational Linguistics, 2011.
- [28] SB Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, pages 3–24. IOS Press, 2007.
- [29] John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.
- [30] Vasileios Lampos, Tijn De Bie, and Nello Cristianini. Flu detector-tracking epidemics on twitter. In *Machine Learning and Knowledge Discovery in Databases*, pages 599–602. Springer, 2010.
- [31] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.
- [32] Rui Li, Kin Hou Lei, Ravi Khadiwala, and Kevin Chen-Chuan Chang. Tedas: A twitter-based event detection and analysis system. In *Data engineering (icde), 2012 ieee 28th international conference on*, pages 1273–1276. IEEE, 2012.
- [33] CC Liebrecht, FA Kunneman, and APJ van den Bosch. The perfect solution for detecting sarcasm in tweets# not. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 29–37. New Brunswick, NJ: ACL, 2013.
- [34] Chenghua Lin, Yulan He, and Richard Everson. Sentence subjectivity detection with weakly-supervised learning. In *IJCNLP*, pages 1153–1161, 2011.
- [35] Julie B Lovins. *Development of a stemming algorithm*. MIT Information Processing Group, Electronic Systems Laboratory, 1968.
- [36] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.
- [37] Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

- [38] Salvatore T March and Gerald F Smith. Design and natural science research on information technology. *Decision support systems*, 15(4):251–266, 1995.
- [39] Angel R Martinez. Part-of-speech tagging. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(1):107–113, 2012.
- [40] Kamran Massoudi, Manos Tsagkias, Maarten de Rijke, and Wouter Weerkamp. Incorporating query expansion and quality indicators in searching microblog posts. In *Advances in information retrieval*, pages 362–367. Springer, 2011.
- [41] Michael Mathioudakis and Nick Koudas. Twittermonitor: trend detection over the twitter stream. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 1155–1158. ACM, 2010.
- [42] Rada Mihalcea, Carmen Banea, and Janyce Wiebe. Multilingual subjectivity and sentiment analysis. In *Tutorial Abstracts of ACL 2012*, pages 4–4. Association for Computational Linguistics, 2012.
- [43] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.
- [44] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.
- [45] Alexandra Olteanu, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. Crisislex: A lexicon for collecting and filtering microblogged communications in crises. In *In Proceedings of the 8th International AAAI Conference on Weblogs and Social Media (ICWSM' 14)*, 2014.
- [46] Heather S Packer, Sina Samangooei, Jonathon S Hare, Nicholas Gibbins, and Paul H Lewis. Event detection using twitter and structured semantic query expansion. In *Proceedings of the 1st international workshop on Multimodal crowd sensing*, pages 7–14. ACM, 2012.
- [47] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.
- [48] Martin F Porter. Snowball: A language for stemming algorithms, 2001.
- [49] 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*, pages 851–860. ACM, 2010.
- [50] Jagan Sankaranarayanan, Hanan Samet, Benjamin E Teitler, Michael D Lieberman, and Jon Sperling. Twitterstand: news in tweets. In *Proceedings of the 17th acm sigspatial international conference on advances in geographic information systems*, pages 42–51. ACM, 2009.

- [51] Peter H Sellers. The theory and computation of evolutionary distances: pattern recognition. *Journal of algorithms*, 1(4):359–373, 1980.
- [52] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.
- [53] Temple F Smith and Michael S Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.
- [54] Bihao Song. *Weak signal detection on twitter datasets: a non-accumulated approach for non-famous events*. PhD thesis, TU Delft, Delft University of Technology, 2012.
- [55] B Srinivasan and P Mekala. Mining social networking data for classification using reptree. *International Journal*, 2(10), 2014.
- [56] R.J.P. Stronkman. Exploiting twitter to fulfill information needs during incidents. Master’s thesis, Delft University of Technology, 06 2011.
- [57] Teun Terpstra, A de Vries, R Stronkman, and GL Paradies. Towards a realtime twitter analysis during crises for operational crisis management. In *ISCRAM’12: Proceedings of the 9th International ISCRAM Conference*, 2012.
- [58] Erik Tromp. Multilingual sentiment analysis on social media. *Master’s Thesis. Department of Mathematics and Computer Science, Eindhoven University of Technology*, 2011.
- [59] Sarah Vieweg, Amanda L Hughes, Kate Starbird, and Leysia Palen. Microblogging during two natural hazards events: what twitter may contribute to situational awareness. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1079–1088. ACM, 2010.
- [60] Atro Voutilainen. Part-of-speech tagging. *The Oxford handbook of computational linguistics*, pages 219–232, 2003.
- [61] Maximilian Walther and Michael Kaisser. Geo-spatial event detection in the twitter stream. In *Advances in Information Retrieval*, pages 356–367. Springer, 2013.
- [62] Xiaojin Zhu. Semi-supervised learning literature survey. *Computer Sciences Technical Report 1530*, 2005.

Appendix A

Features

A.1 Feature list

#	Feature summary	Instance dependent	Feature input	Document input	Discussion
1	Tweet, hard match, 1st degr.	Yes	Keywords, 1st degr.	Tweet	Section A.2.1
2	Tweet, hard match, 2nd degr.	Yes	Keywords, 2nd degr.	Tweet	Section A.2.1
3	Tweet, hard match, 3rd degr.	Yes	Keywords, 3rd degr.	Tweet	Section A.2.1
4	Tweet, compound match, 1st degr.	Yes	Keywords, 1st degr.	Tweet	Section A.2.1
5	Tweet, compound match, 2nd degr.	Yes	Keywords, 2nd degr.	Tweet	Section A.2.1
6	Tweet, compound match, 3rd degr.	Yes	Keywords, 3rd degr.	Tweet	Section A.2.1
7	Tweet, stemmed match, 1st degr.	Yes	Keywords, 1st degr.	Tweet	Section A.2.1
8	Tweet, stemmed match, 2nd degr.	Yes	Keywords, 2nd degr.	Tweet	Section A.2.1
9	Tweet, stemmed match, 3rd degr.	Yes	Keywords, 3rd degr.	Tweet	Section A.2.1
10	Tweet, approximate match, 1st degr.	Yes	Keywords, 1st degr.	Tweet	Section A.2.1
11	Tweet, approximate match, 2nd degr.	Yes	Keywords, 2nd degr.	Tweet	Section A.2.1
12	Tweet, approximate match, 3rd degr.	Yes	Keywords, 3rd degr.	Tweet	Section A.2.1
13	Hypernym, approximate match, 1st degr.	Yes	Hypernyms, 1st degr.	Tweet	Section A.2.2
14	Hypernym, approximate match, 2nd degr.	Yes	Hypernyms, 2nd degr.	Tweet	Section A.2.2
15	Hypernym, approximate match, 3rd degr.	Yes	Hypernyms, 3rd degr.	Tweet	Section A.2.2
16	Hyponym, approximate match, 1st degr.	Yes	Hyponyms, 1st degr.	Tweet	Section A.2.2
17	Hyponym, approximate match, 2nd degr.	Yes	Hyponyms, 2nd degr.	Tweet	Section A.2.2
18	Hyponym, approximate match, 3rd degr.	Yes	Hyponyms, 3rd degr.	Tweet	Section A.2.2
19	Username, hard match, 1st degr.	Yes	Keywords, 1st degr.	User descr.	Section A.2.3
20	Username, hard match, 2nd degr.	Yes	Keywords, 2nd degr.	User descr.	Section A.2.3
21	Username, hard match, 3rd degr.	Yes	Keywords, 3rd degr.	User descr.	Section A.2.3
22	Username, compound match, 1st degr.	Yes	Keywords, 1st degr.	User descr.	Section A.2.3
23	Username, compound match, 2nd degr.	Yes	Keywords, 2nd degr.	User descr.	Section A.2.3
24	Username, compound match, 3rd degr.	Yes	Keywords, 3rd degr.	User descr.	Section A.2.3

Table A.1: List of features, part 1

#	Feature summary	Instance dependent	Feature input	Document input	Discussion
25	Username, stemmed match, 1st degr.	Yes	Keywords, 1st degr.	User descr.	Section A.2.3
26	Username, stemmed match, 2nd degr.	Yes	Keywords, 2nd degr.	User descr.	Section A.2.3
27	Username, stemmed match, 3rd degr.	Yes	Keywords, 3rd degr.	User descr.	Section A.2.3
28	Username, approximate match, 1st degr.	Yes	Keywords, 1st degr.	User descr.	Section A.2.3
29	Username, approximate match, 2nd degr.	Yes	Keywords, 2nd degr.	User descr.	Section A.2.3
30	Username, approximate match, 3rd degr.	Yes	Keywords, 3rd degr.	User descr.	Section A.2.3
31	Ignore, Keyword	Yes	Ignore keywords	Tweet	Section A.2.4
32	Ignore, Username	Yes	Ignore users	Username	Section A.2.4
33	News and media	No	User-database	Username	Section A.2.5
34	Poster's current tweet rate	No		List of users tweets	Section A.2.6
35	Poster's average tweet rate	No		List of users tweets	Section A.2.6
36	Current stream rate	No		Processed tweet cache	Section A.2.6
37	Positive sentiment	No	<i>Research required</i>	Tweet	Section A.2.7
38	Negative sentiment	No	<i>Research required</i>	Tweet	Section A.2.7
39	Subjectivity value	No	<i>Research required</i>	Tweet	Section A.2.8
40	Present tense	No	<i>Research required</i>	Tweet	Section A.2.9
41	Past tense	No	<i>Research required</i>	Tweet	Section A.2.9
42	Is retweet	No		Retweet attribute, Tweet	Section A.2.10
43	#retweets	No		#retweets attribute	Section A.2.10
44	Is reply	No		Reply attributes	Section A.2.11
45	#pre-mentions	No		Mention attributes	Section A.2.12
46	#inner-mentions	No		Mention attributes	Section A.2.12
47	#news-mentions	No	Media usernames	Mention attributes	Section A.2.12
48	Tweet length	No		Tweet	Section A.2.13
49	Images	No		Media attributes	Section A.2.14
50	Links	No		Link attributes	Section A.2.15
51	#inner-hashtags	No		Hashtag attributes	Section A.2.16
52	#post-hashtags	No		Hashtag attributes	Section A.2.16
53	#list-hashtags	Yes	Hashtag list	Hashtag attributes	Section A.2.16
54	Prior similar tweets	No		Processed tweet cache, Tweet	Section A.2.17
55	Has popular term	No		Processed tweet cache, Tweet	Section A.2.18
56	Distance from hotspot	Yes	Hotspots	Geo attributes	Section A.2.19
57	Seems observation	No		Tweet, Geo attributes	Section A.2.20
58	Contains street language/slang	No		Tweet	Section A.2.21
59	#Nouns	No		Tweet	Section A.2.22
60	#Verbs	No		Tweet	Section A.2.22
61	#Adverbs	No		Tweet	Section A.2.22
62	Future day mentioned	No		Tweet	Section A.2.23
63	Past day mentioned	No		Tweet	Section A.2.23
64	Contains special keyword 1	??	Special keywords	Tweet	Section A.2.24

Table A.2: List of features, part 2

#	Feature summary	Instance dependent	Feature input	Document input	Discussion
65	Contains special keyword 2	??	Special keywords	Tweet	Section A.2.24
66	User language	No		Language attribute	Section A.2.25
67	Tweet language	No		Tweet	Section A.2.25
68	Question marks	No		Tweet	Section A.2.26
69	Subsequent dots	No		Tweet	Section A.2.26
70	Interpunction-ratio	No		Tweet	Section A.2.26

Table A.3: List of features, part 3

A.2 Feature discussions and implementations

A.2.1 Keyword matches

Features 1–30 relate to keyword matches within tweets and usernames, whereas features 1–12 specifically address keyword matches within tweets. This section will mainly address the latter subset, but does also apply to the complete set of matcher features. However, sections A.2.2 and A.2.3 will extend on this section for other subsets of the matcher features.

The main mechanism of the existing Twitcident classification mechanism relies on keyword matches (Observation 2), and we want to extend on this process. Therefore, a core subset of features is dedicated to keyword matches. However, due to Observation 6 we consider it a high priority to support dynamic keyword lists that can be altered during runtime. As Machine Learning classifiers depend on a fixed set of features over which they are trained, we have chosen for an atypical alternative construction: we define a set of fixed features that are configured with a keyword-list, rather than a single keyword. We regard this as a potential new kind of Machine Learning setup: *features do not have to be static, but may be dynamic*.

This way, the keyword list is extensible and adjustable during runtime, without changing the machine feature and its meaning: this meaning can be interpreted as “relevancy of this tweet with respect to keywords in keyword list KL ”. The main drawback is that all keyword matches are treated equally. For example, if a keyword list contains both “throwing+a+bike” and “shooting+at+police”, and this features value tells the classifier that there is a match, the classifier will not know which particular entry was matched. However, we value the *dynamic extensibility* over this due to Observations 6 and 14, and will capture and resolve the consequences.

Each feature is configured with a keyword list $KL = \{KE_1, KE_2, \dots\}$ with $|KL| \geq 0$, where each keyword-entry KE_i consists of:

- A set of positive keywords $P_{KE_i} = \{PK_1, PK_2, \dots\}$ with $|P| \geq 1$. Each of the individual keywords PK_j must be present to result in a match for this entry.
- A set of negative keywords $N_{KE_i} = \{NK_1, NK_2, \dots\}$ with $|N| \geq 0$. If at least one of the individual keywords NK_k is present, this keyword entry KE_i will never result in a match.
- A weight $W_{KE_i} \in \mathbb{Z} \setminus 0$ for this keyword entry, that will be added to the final output value for this feature for each match.

The set of negative keywords NK serves to restrict the space captured by PK , and define its context (see Observation 12). An example of a very short keyword list KL could be:

$$KL = \left\{ \begin{array}{l} KE_1 = \left\{ \begin{array}{l} PK = \{ \text{'aardbeving'}, \text{'in'} \} \\ NK = \emptyset \\ W = 5 \end{array} \right. \\ KE_2 = \left\{ \begin{array}{l} PK = \{ \text{'aardbeving'} \} \\ NK = \{ \text{'in'} \} \\ W = 1 \end{array} \right. \end{array} \right.$$

In this example, the additional word ‘in’ could indicate that the tweet contains additional information like a place, which would increase the relevancy of the match. Therefore, a match including ‘in’ will increase the feature output value more than a match without an occurrence of ‘in’. Note that in this example, both entries are mutually exclusive, but this does not necessarily be the case.

The final score S of this numerical feature is defined as the sum of keyword entry match weights multiplied by the amount of matches for each entry:

$$S = \sum_{i=1}^{|KL|} (W_{KE_i} \cdot \#matches_{KE_i})$$

Note that if a keyword entry does not match, $\#matches_{KE_i}$ will be zero and the weight is not added to the output value. Also note, that an individual entry is thus allowed to match multiple times.

Suppose another case, in which we are looking for the word ‘bank’. In Dutch this word has multiple contexts, two of them literally translating to a ‘bench’/‘sofa’ and the other one to a ‘bank’ associated with money. On the former one, you can sit. The latter definition one is our intended context. We could now construct the following definitions to more specifically address this context:

$$KL = \left\{ \begin{array}{l} KE_1 = \left\{ \begin{array}{l} PK = \{ \text{'bank'} \} \\ NK = \emptyset \\ W = 1 \end{array} \right. \\ KE_2 = \left\{ \begin{array}{l} PK = \{ \text{'bank'} \} \\ NK = \left\{ \begin{array}{l} \text{'zit'}, \text{'tv'}, \text{'beeldbank'}, \text{'bloedbank'}, \\ \text{'boekenbank'}, \text{'spermabank'}, \text{'kennisbank'}, \\ \text{'achterbank'}, \text{'databank'}, \text{'zonnebank'}, \dots \end{array} \right\} \\ W = 5 \end{array} \right. \\ KE_3 = \left\{ \begin{array}{l} PK = \{ \text{'bank'}, \text{'zit'}, \text{'tv'} \} \\ NK = \emptyset \\ W = -1 \end{array} \right. \end{array} \right.$$

The first entry defines a match on the raw target word ‘bank’, and adds 1 to the output value for each match. The second entry also specifies negative keywords, to eliminate all unintended meanings: if one of them does occur, this entry will not match. If none of them occur but ‘bank’ does occur, it add 5 to the output value.

Because we do not want to put *all* keywords on a single pile due to some keyword sets being more specific and relevant than others, we defined multiple identical features

of this type that will just be configured with their own keyword list. We chose to create multiple degrees (levels) of keyword lists to support this. We will elaborate on this the specific meanings of each degree later on in Section 4.4.

Finally, we also defined multiple *matching schemes*:

Hard match A hard match results in a match for a keyword if it is matched exactly as a word, thus surrounded by spaces.

Compound match A compound match results in a match for a keyword if it is matched exactly at either the beginning or the end of a word, thus has at least one space on its boundaries.

Stemmed match A stemmed match results in a match for a keyword if its stemmed form exactly matches the stemmed form of one of the words of the input document (Tweet). A language specific stemmer is required.

Approximate match To account for typos (Observation 19) and to compensate a bit for the lack of wildcards (Observation 3), we also include an approximate matcher that returns a similarity value in $\langle 0, 1 \rangle$ for each keyword. If this approximation value exceeds a certain threshold (additional feature parameter), it is counted as a match with the weight scaled by this approximation value. After experimenting with well-known string approximation algorithms, we adapted Dice’s similarity metric resulting in an algorithm that suits our problem better. We elaborate on this in Appendix B.

Table A.4 illustrates an example of the different matching schemes on input (assume every keyword has weight 1):

“Vandaag is het weer kinderboekendag! De volgende boekjes zijn genomineerd: ...”
(*The typo in ‘kinderboekendag’ is on purpose*)

Keyword	Hard	Compound	Stemmed	Our approximation algorithm	Approximate Smith-Waterman
dag	0	1	0	1.00	1.00
boek	0	2	0	1.00	1.00
dagboek	0	0	0	0.00	0.57
kinderboekendag	0	0	0	0.84	0.90

Table A.4: Illustration match value for different matching schemes

Although we think we have covered the core concepts with this feature, something that remains open for further research is whether order may be relevant and how to integrate that in the matching features (see Observation 26). Furthermore, we could introduce additional constraints to keyword entries, for example, a maximum distance between two individual words.

A.2.2 Hypernyms and hyponyms

By Observation 11 we have seen that hypernyms and hyponyms may be relevant. Therefore, we add additional matching features to hyponyms and hypernyms of the

corresponding keyword lists. However, as these are less crucial than our main keywords, we only apply the approximate matching schema, as this also catches exact and compound matches but at the cost of capturing ‘too much’. Features 13–18 cover this.

A.2.3 Username matches

Also usernames can contain a keyword. For example, the word ‘aardbeving’ (‘earthquake’) or ‘treinvertraging’ (‘train delays’) could be part of a username. These may provide relevant information given a specific subject, as these are mostly very specialized accounts. Features 19–30 cover this.

A.2.4 Ignore features

By Observation 4 and 5 we have seen that it may be useful to ignore tweets if they contain a certain keyword or are posted by a certain user. Features 31–32 cover this, and are practically a negation of a regular matching feature, although it operates on a simpler list just defining words, rather than word combinations with weights.

A.2.5 News and media

By Observation 27 we have seen that tweets posted by (news) media are often irrelevant to most instances, but may be relevant to a few. This feature (33) checks whether the posting user is related to (news) media, based on a precompiled database of users.

A.2.6 Tweet and stream rates

When something ‘happens’, twitters usually start to tweet about it, causing an increase in tweet rates (amount of tweets per interval). With Feature 34 we intend to measure the user’s tweet rate, and with Feature 35 the average user’s tweet rate, both measuring the amount of tweets over a fixed interval (parameter). Feature 36 measures the incoming Twitter stream rate.

A combination of the output of these features may provide contextual information for other features. For example, the amount of hits by match features can implicitly be related to the stream rate. Also, sudden increases in stream volume can indicate something is happening, so we might add an additional feature that calculates the current derivative of a normalized stream rate to detect and measure an increase or decrease in stream volume.

However, to measure the user’s tweet rates, a call to the Twitter API is required for each processed tweet which will significantly slow down the feature calculation part of the classification process. Therefore, implementation of these features has been postponed.

The Twitter stream rate feature is implemented by maintaining a look-up cache of all prior processed tweets over the past time window (fixed parameter, actual value is irrelevant), which is also used by some other features (for example, Feature 54, Section A.2.17). The output value of this feature is then represented by the size of this cache. As the look-up cache is bounded to a maximum size for performance reasons, we can use this maximum value to normalize the output.

A.2.7 Sentiment

Positive and negative sentiment could affect the relevancy of a tweet. For example, in case of detecting earthquakes, an abundance of positive sentiment is highly likely to yield the tweet irrelevant. On the other hand, we have that several instances have a dedicated sentiment topic, including the earthquake instance. Features 37–38 cover positive sentiment and negative sentiment respectively, and output a normalized value indicating the sentiment likeliness.

Due to the large amount of research required, the complexity of these features (think about detecting sarcasm) and the fact our target language is Dutch and most research is done on English, these features have not been implemented yet, and are in a conceptual state. Pointers for future research include the 4th Workshop on Computational Approaches for Subjectivity, Sentiment and Social Media analysis 2013 [3] and the work of Tromp [58], Jiang et al. [27], Gonzalez et al. [23] and Liebrecht et al. [33].

A.2.8 Subjectivity

Some instances are interested in public opinion, whereas others are focused upon objective user observations (also see Section A.2.20). Therefore, the amount of subjectivity could be of interest. Feature 39 covers this, and outputs a normalized ratio.

Due to the large amount of research required, the complexity of these features¹ and the fact our target language is Dutch and most research is done on English, these features have not been implemented yet, and are in a conceptual state. Pointers for future research include the work of Mihalcea et al. [42], Balahur et al. [5] and Lin et al. [34].

A.2.9 Tense

By Observation 22 we have seen that the verbal tense can have significant impact on the relevancy of a tweet. For Real-Time social monitoring, present tense is most important.

Due to the complexity of this feature and to maintain a scope on this thesis work, this feature has not been implemented yet, and is in a conceptual state.

A.2.10 Retweet(s)

The retweet attribute of a tweet may be relevant, as well as the ‘RT: ’ prefix indicator of a retweet. Feature 42 checks both and will output 1 in case of a retweet, 0 otherwise. In general, retweets are not relevant as the original tweet should be present in our incoming stream too. Feature 43 directly outputs the attribute of a tweet that indicates the amount that tweet has been retweeted. This feature is expected to be obsolete, since all tweets should be processed real-time and therefore can not be retweeted yet. Included for completeness.

Only Feature 42 is implemented yet, while Feature 43 remains not implemented.

¹In [34] Lin, He and Everson state that subjectivity classification is even harder than sentiment classification

A.2.11 Reply

The reply attributes of a tweet may be relevant, indicating conversations or retweets with additional appended information. Feature 44 checks the reply attributes of the tweet object and outputs 1 if it is a reply, 0 otherwise.

A.2.12 Mentions

Mentions are special constructs in a tweet identified by an '@' sign, indicating a referred user. This usually indicates retweets, replies (conversations) or notifications. We distinguish three types of mentions:

- Pre-mentions: a mentioned user at the beginning of a tweet, excluding known prefixes (for example 'RT: '). If a tweet starts with three consecutive mentions, we consider all of these pre-mentions. These usually indicate retweet or reply type mentions.
- Inner-mentions: a mentioned user that is NOT mentioned at the beginning of a tweet, but in the middle or at the end. These usually indicate notification or reference type mentions.
- News-mentions: More generalized a 'list-mention': the mention can be cross-referenced to a precompiled list (parameter input) of usernames. In our case we will use this specifically to indicate mentions of news-media, because media-tweets often mention themselves, and to identify newsmedia-retweets more accurately.

We constructed three features (45–47) which respectively output the amount of occurrences for each of the defined mention-types.

A.2.13 Tweet length

The length of the tweet may provide additional information on its relevancy. For example, extremely short tweets are irrelevant, but also more complex cases: there may be a correlation between the length of a tweet and the amount of URLs or mentions with respect to its relevancy. Feature 48 outputs the length of the tweet.

A.2.14 Images

By Observation 13 we have seen that images may provide additional relevant information. Feature 49 outputs the amount of images contained within the tweet.

A.2.15 Links

Web link occurrences may provide additional information about the relevancy of a tweet, and are another indicator of potential news media related content, as news-related tweets tend to always contain an URL. Feature 50 outputs the amount of images contained within the tweet.

A.2.16 Hashtags

Hashtags are special constructs in a tweet identified by an ‘#’ sign, to put emphasis on a word or to indicate a relation to a certain subject. We distinguish three types of mentions:

- **Post-hashtags:** a hashtag at the end of a tweet. If a tweet ends with multiple consecutive hashtags, we consider all of these post-hashtags. These usually indicate a relation to a certain subject, or express a conclusive feeling.
- **Inner-hashtags:** a hashtag occurring in the middle of a tweet. These usually indicate emphasis on the word they are applied to.
- **List-hashtags:** the hashtag can be cross-referenced to a precompiled list (parameter input) of keywords. Implemented as additional feature that may provide to be useful.

We constructed three features (51–53) which respectively output the amount of occurrences for each of the defined hashtag-types.

A.2.17 Prior similar tweets

Often in case of special events or incidents, tweets can be extremely similar and have only minor differences (if any) due to excessive retweeting and replying. Sometimes, a tweet is retweeted multiple times in a chain-linked fashion, each only appending a new ‘pre-mention’ at the start, but without actually providing any new content.

Therefore, we constructed a feature (54) that looks up the amount of similar prior tweets. This is implemented by maintaining a processed tweet cache (bounded by a maximum size), and attach this value to every processed tweet. This cache is the same as mentioned in Section A.2.6. By traversing this cache from most recent to less recent, we can stop traversing this list when we find the first similar tweet containing such an attached precalculated value, and incrementing it by one. This feature outputs the amount of prior similar tweets given a fixed time window (parameter).

A.2.18 Contains popular term

We may extract additional information on the relevancy of a tweet if it contains a word that suddenly became more popular. This is similar to the bursty keyword detection presented in [41]. This can further be corroborated by the research presented in Appendices G and H. Therefore, we could construct a feature (55) that indicates whether the tweet contains a suddenly popular term. However, to efficiently implement such a feature, we would need to rely on an Information Retrieval engine that indexes prior processed tweets, which we postponed for now. Furthermore, we should identify what exactly defines a term as ‘popular’, and maintain the popularity of terms to detect ‘sudden’ increases.

A.2.19 Distance from hotspot

By Observation 32 we have seen that geo-information attached to a tweet can provide additional information, especially since most Twitcident Instances monitor a spe-

cific area. Therefore, Feature 56 outputs the straight line distance to the closest defined hotspot: this feature requires a list of hotspots as configured input. If no geo-information is attached this feature outputs a distance so large it cannot be relevant to indicate this. We consider a value of 100 kilometers sufficient. For normalization purposes, we also cap the maximum distance to this value; i.e. if someone tweets 250 kilometers away from the closest hotspot, this feature will still output 100 kilometers.

A.2.20 Seems observation

Especially relevant are user observations: what do they notice? what do they see? smell? hear? etc. We track special (combinations of) words that indicate observations, and construct a feature (57) that outputs the likeliness that a tweet contains an observation. Furthermore, if a geo-location is available and the distance to the closest hotspot exceeds a threshold, we will not consider it an observation.

A.2.21 Street language and slang

By Observation 17 we have seen that some people commonly use street language or slang in their tweets. This could provide information about the type of user posting, which may contribute to estimating the relevancy of a tweet. We precompiled a list of common slang terms, which is cross-referenced by feature (58) outputting the amount similar to the matching features.

A.2.22 Part of Speech attributes

Part of Speech tagging is a mechanism that provide additional useful information on how a tweet is composed. It identifies the part of speech of a word, i.e. whether it is a noun, verb, adverb, . . . We constructed three features (59–61) that count the amount of nouns, verbs and adverbs respectively. The ratio of nouns to verbs may tell something about the usefulness of a tweet, while the amount of adverbs tells us how descriptive a tweet is. These features are powered by the Apache OpenNLP library² for the actual Part Of Speech tagging.

A.2.23 Future or past days mentioned

Tweets posted by newsmedia often mention a specific day, often a day in the past. For example, on a Thursday, newsmedia often mention ‘yesterday’, ‘Wednesday’ or ‘Tuesday’. Such a past day mentioned provides an additional indication that it may be news related content. A future day mentioned on the other hand could indicate an upcoming event, which may be relevant. Therefore, we constructed two features (62–63) that detect days of week mentioned. When such a day is within the next four days with respect to the tweet timestamp, we consider it a future day; a past day otherwise.

A.2.24 Contains special keywords

During our analysis, we have observed that some words have a significant impact on relevancy and to emphasize this, we constructed two additional features for these. The

²<https://opennlp.apache.org/>

first feature (64) counts keywords (parameter) that have a highly likely positive effect on relevancy like ‘help’, ‘gewond(e)’ (casualty), ‘foto(s)’ (photo), ‘wtf’, . . . , while the second feature (65) counts keywords (parameter) that have a highly likely negative impact on tweet relevancy, like ‘waarschijnlijk’ (probably), ‘incident’, ‘nieuws’ (‘news’), . . .

A.2.25 Language

In general, most instances serve only a specific audience with a single language, in our case mostly Dutch. Some ambiguous words are used in another language too, and often cause noise. For example, the Dutch word ‘brand’ means ‘fire’, but has a totally different meaning in English, which also applies to conjugations of it: brand-manager.

The user has an attribute indicating his language set, which is captured by Feature 66. However, this is not very reliable, since this is a profile setting which is set to English by a lot of Dutch people. Therefore, we provide an additional feature (67) that attempts to identify a match to the target language by checking for specific words (parameter) indicating one of the target languages, or contra-indications due to words from another language (parameter).

A.2.26 Interpunction

Interpunction within a tweet can also provide additional information about the relevancy of a tweet. For example, a question mark usually indicates a question which in turn might indicate that the tweet does not necessarily provide relevant information itself. Therefore, we constructed three features (68–70) that respectively output:

- The amount of question marks in a tweet,
- The amount of subsequent dots in a tweet (i.e. ‘...’ counts as 2, while ‘....’ followed by ‘.’ counts as $3+1=4$),
- The interpunction-ratio: the amount of non-interpunction characters divided by the amount of interpunction characters

Appendix B

Approximate String Matching algorithms

During our work on the approximate keyword match and similar prior tweets features, we encountered the problem of approximate string matching, involving string similarity metrics. Given a set of keywords, we wanted to know whether this combination occurs within a tweet, taking potential typos, misspellings or alternative forms into account. We are looking for a lightweight fast performing algorithm outputting a score representing the likelihood that a given string occurs within a larger string. To this end, we present our algorithm in this Appendix, which is an adaptation of Dice's similarity measure [17].

B.1 Related work

Most algorithms are closely related to sequence alignment in bio-informatics. The most widely known string metric is the Levenshtein Distance [31], returning a number equivalent to the number of substitutions and deletions needed in order to transform one input string into another, known as an edit distance. The Damerau-Levenshtein [8, 15] distance also allows transposition operations. While these algorithms focus on minimizing the edit distance, the Needleman-Wunsch [44] algorithm focuses on maximizing similarity which turns out to be equivalent. Whereas the Needleman-Wunsch algorithm translates to global sequence alignment, the Smith-Waterman [53] algorithm performs local sequence alignment. Dice's coefficient [17] is a set similarity metric, which can be applied on strings by splitting them in sets of n -grams.

Most of these metrics and algorithms calculate the similarity or distance between two strings, with a length difference also imposing a penalty. However, when we want to know if a smaller string (i.e. a keyword) occurs misspelled within a larger string (i.e. a tweet) we need to apply string approximation algorithms for variable length, or approximate substring matching. A brute-force approach would be to compute the edit distance (for example by applying Levenshtein) to the smaller string for all substrings of the larger string, and then choose the substring with the minimum distance. However, this algorithm would have asymptotic running time $O(n^3m)$. Sellers [51] improves on this by applying dynamic programming (DP) while introducing a parameter k representing the maximum amount of mismatches. Other techniques involve

indexing. However, as we are not interested in the exact edit distance but rather in a lightweight fast-performing algorithm measuring similarity, we have developed our own algorithm by adapting Dice’s similarity coefficient.

After experimenting with dozens of string metrics from two commonly used libraries, SecondString¹ and SimMetrics², we concluded that the Smith-Waterman and Monge-Elkan algorithm performed best in terms of matching quality yet not satisfying enough with respect to speed. Monge-Elkan also performed extremely slow. The fact that *most* of the algorithms or metrics was able to perform satisfying enough can be attributed to the *variable* length of the strings: we actually want to check for an approximate substring match. Smith-Waterman is able to handle this best because it searches for an optimal *local* sequence alignment.

B.2 Problem definition

Given a set of n keywords $\mathcal{K}_0 \dots \mathcal{K}_n$ and a string \mathcal{T} , our task is to output a normalized approximation score $0.000 \leq \mathcal{S} \leq 1.000$ representing the likeliness that string \mathcal{T} contains (approximations of) all keywords \mathcal{K} . A score of $\mathcal{S} = 1$ should indicate a perfect match.

Note that \mathcal{K} can also be a single ‘keyword’ that may be even longer than \mathcal{T} , in the case of comparing two tweets.

B.3 Our solution

We have developed several algorithms to solve this task covering a wide range of ideas. While several of them outperform metrics from both libraries, we will only discuss our best performing solution. It also sparkles of simplicity, being based on a well known metric: Dice’s coefficient. We basically adapt this metric to suit our needs.

Given two sets of samples A and B , Dice’s coefficient is defined by:

$$\mathcal{S} = \frac{2|A \cap B|}{|A| + |B|}$$

When applying this to strings, it is common to define both sets by the sets of bigrams of both strings. Let us consider two sample strings ‘night’ and ‘nacht’, then sets A and B will be:

$$\begin{aligned} A &= \{ni, ig, gh, ht\} \\ B &= \{na, ac, ch, ht\} \end{aligned}$$

and Dice’s coefficient will be:

$$\mathcal{S} = \frac{2|A \cap B|}{|A| + |B|} = \frac{2 \cdot 1}{4 + 4} = 0.25$$

Now consider the case we have a single keyword $\mathcal{K}_0 = \text{‘aardbeving’}$ and a string $\mathcal{T} = \text{‘Vanmorgen aardbeving gevoeld in Middelstum e.o. Sterkte nog niet bekend,’}$

¹<http://secondstring.sourceforge.net/>

²<https://github.com/Simmetrics/simmetrics>

maar een bewoner schat tussen 2,5 en 3 op Richter.’, we would end up with $S \approx 0.095$, although we are looking to achieve a perfect score of 1.000 here.

We therefore adapt this metric to suit our needs. First we introduce a parameter \mathcal{N} indicating the n-gram size, so if $\mathcal{N} = 2$ we would be using bigrams. Now let us define the function $ngrams(s, \mathcal{N})$ as returning all \mathcal{N} -grams of string s . Then the sets A and B are defined as:

$$A = \bigoplus_{i=1}^n (ngrams(\mathcal{X}_i, \mathcal{N}))$$

$$B = ngrams(\mathcal{T}, \mathcal{N})$$

We now define our approximation score by:

$$S = \frac{|A \cap B|}{\min(|A|, |B|)}$$

B.4 Demonstration

Although we have performed extensive experiments, we will only present a demonstration of results here. Although there are many other (edge) cases we can think of, we can state that results below reflect other cases too.

ID	Target string
A	aardbeving
B	Vanmorgen aardbeving gevoeld in Middelstum e.o. Sterkte nog niet bekend, maar een bewoner schat tussen 2,5 en 3 op Richter.
C	Aardbevin gevoeld vanmorgen in Middelstum e.o. Sterkte nog niet bekend, maar een bewoner schat tussen 2,5 en 3 op Richter.
D	Dijken sneller verstevigd om aardbeviingen in Groningen http://t.co/ygSlrkKBuF #Nieuwsflitser
E	Vanmorgen beving gevoeld in Middelstum e.o. Sterkte nog niet bekend, maar een bewoner schat tussen 2,5 en 3 op Richter.
F	gevoeld vanmorgen in Middelstum e.o. Sterkte nog niet bekend, maar een bewoner schat tussen 2,5 en 3 op Richter.
M	Because reality. RT @AdamKilgoreWP: "If we got in, we'd be the team to beat." – Dan Haren.
N	hahahahahahahaha RT @AdamKilgoreWP: "If we got in, we'd be the team to beat." – Dan Haren."
O	Classy "@JoeStrauss: They were team to beat (nightly) for 4.5 months RT @AdamKilgoreWP: "If we got in, we'd be the team to beat." Dan Haren"
P	Dan Haren: "The Nationals will be a scary team next year. Nobody wanted to play us this year. If we got in, we'd be the team to beat." Lolz

Figure B.1: Approximate substring matching: Test cases

Figure B.1 lists our demonstration cases. Cases A to F will be targeted by the keywords ‘aardbeving’ and ‘beving’. Now note that A to F are ordered with respect to desired output value according to the first keyword ‘aardbeving’: Cases A to D should receive high scores, case E a doubtful score and F should be as low as possible. The results for the second keyword ‘beving’ should be similar, although case E should be ranked higher than C and D due to containing a perfect match.

Cases M, N, O and P all contain similar strings and will all be matched against eachother. The difference here is that one string is not considerably shorter than the others, which yields a different use-case (similar tweet matching). Our aim is that all of these strings should be considered somewhat similar.

Figure B.2 lists the results of our test cases with various algorithms and metrics. Tables 1 and 2 are vertically color-coded since we want to compare the values relatively to eachother as the cases differ. Table 3 is horizontally color-coded since all cases

Table 1	Keyword = 'aardbeving' vs	A	B	C	D	E	F
	Our solution, with N=2	1,000	1,000	0,889	1,000	0,778	0,556
	Our solution, with N=3	1,000	1,000	0,875	0,875	0,625	0,125
	Our solution, with N=4	1,000	1,000	0,857	0,714	0,429	0,000
	SS::Monge-Elkan	1,000	1,000	0,900	0,880	0,600	0,380
	SM::Dice Coefficient	1,000	0,095	0,000	0,000	0,000	0,000
	SM::Smith-Waterman	1,000	1,000	0,850	0,900	0,600	0,300
Table 2	Keyword = 'beving' vs	A	B	C	D	E	F
	Our solution, with N=2	1,000	1,000	0,800	1,000	1,000	0,600
	Our solution, with N=3	1,000	1,000	0,750	0,750	1,000	0,000
	Our solution, with N=4	1,000	1,000	0,667	0,333	1,000	0,000
	SS::Monge-Elkan	1,000	1,000	0,833	0,800	1,000	0,500
	SM::Dice Coefficient	0,000	0,000	0,000	0,000	0,095	0,000
	SM::Smith-Waterman	1,000	1,000	0,917	0,833	1,000	0,333
Table 3	Full tweet comparison	M vs N	M vs O	M vs P	N vs O	N vs P	O vs P
	Our solution, with N=2	0,822	0,833	0,644	0,753	0,570	0,551
	Our solution, with N=3	0,820	0,775	0,528	0,739	0,511	0,409
	Our solution, with N=4	0,818	0,750	0,500	0,725	0,484	0,353
	SS::Monge-Elkan	0,824	0,681	0,451	0,664	0,436	0,305
	SM::Dice Coefficient	0,848	0,650	0,476	0,667	0,488	0,417
	SM::Smith-Waterman	0,824	0,764	0,456	0,739	0,441	0,299

Figure B.2: Approximate substring matching: Results on test cases

match positively against each other, so we want to compare the different metrics against each other.

The results show that our algorithm works as intended and performs competitively to or better than well-known algorithms. Based on these results and extensive experiments, we have chosen to use our algorithm using trigrams ($\mathcal{N} = 3$) and a threshold value of 0.6, i.e: we consider something an approximate match if it scores above 0.6 and return the score, otherwise we return 0.

B.5 Performance considerations

\mathcal{N} -grams can be obtained in linear time ($O(n)$, where $n = \max(|A|, |B|)$). To calculate the intersection, we first apply a sorting algorithm and then make a single pass through the single list (linear time). Sorting algorithms can run in $O(n \cdot \log(n))$ time, yielding an asymptotic time complexity for our algorithm of $O(n \cdot \log(n))$.

Furthermore, we have implemented this algorithm in a highly optimized way by noting the fact that \mathcal{N} -grams of up to 4 characters can be stored in a single unsigned integer. Using this implementation resulted in a 300 times faster performance than using the conventional libraries, which is useful in our case of comparing tweets to many prior tweets.

Finally, we also implemented this algorithm using precalculated caches, by exploiting the fact that the lookup keywords remain the same. Therefore, we calculate the set of \mathcal{N} -grams A only once and caching it. Since we only store integers for each \mathcal{N} -gram, the maximum memory usage is linear ($O(n)$).

Appendix C

Reproducibility information

C.1 SQL statements

C.1.1 Database structure, schema: Classification

```
CREATE DATABASE IF NOT EXISTS `Classification`
USE `Classification`;

-- Default data for features that are not instance-specific, for example lists
-- of proverbs, default words for features (i.e. special keywords features), etc
CREATE TABLE `preconfig_featureinput` (
  `id_entry` `int`(11) NOT NULL AUTO_INCREMENT,
  `id_instance` `varchar`(10) NOT NULL,
  `id_featureinput` `varchar`(50) NOT NULL,
  `entry1` `varchar`(255) NOT NULL,
  `entry2` `varchar`(255) NOT NULL,
  `meta1` `varchar`(255) NOT NULL,
  `meta2` `varchar`(255) NOT NULL,
  PRIMARY KEY (`id_entry`),
  KEY `INDEX` (`id_instance`, `id_featureinput`)
) ENGINE=MyISAM AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;

-- List of features for each instance
CREATE TABLE `tbl_config_feature` (
  `id_instance` `varchar`(10) NOT NULL,
  `id_feature` `varchar`(30) NOT NULL,
  `feature_order` `tinyint`(4) NOT NULL DEFAULT '0',
  `feature_enabled` `tinyint`(1) NOT NULL DEFAULT '1',
  `feature_class` `varchar`(60) NOT NULL,
  `feature_param1` `varchar`(50) NOT NULL,
  `feature_param2` `varchar`(50) NOT NULL,
  `featureinput1_class` `varchar`(100) NOT NULL,
  `featureinput1_param1` `varchar`(50) NOT NULL,
  `featureinput1_param2` `varchar`(50) NOT NULL,
  `featureinput2_class` `varchar`(100) NOT NULL,
  `featureinput2_param1` `varchar`(50) NOT NULL,
  `featureinput2_param2` `varchar`(50) NOT NULL,
  PRIMARY KEY (`id_instance`, `id_feature`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

-- Input data for each feature, for example lists of words, hotspots, etc.
CREATE TABLE `tbl_config_featureinput` (
  `id_entry` `int`(11) NOT NULL AUTO_INCREMENT,
  `id_instance` `varchar`(10) NOT NULL,
  `id_featureinput` `varchar`(50) NOT NULL,
  `entry1` `varchar`(255) NOT NULL,
  `entry2` `varchar`(255) NOT NULL,
```

```

    `metal` varchar(255) NOT NULL,
    `meta2` varchar(255) NOT NULL,
    PRIMARY KEY (`id_entry`),
    KEY `INDEX` (`id_instance`,`id_featureinput`)
) ENGINE=MyISAM AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;

-- Settings for each instance
CREATE TABLE `tbl_config_setting` (
  `id_instance` varchar(10) NOT NULL DEFAULT '',
  `key` varchar(50) NOT NULL,
  `value` varchar(200) DEFAULT NULL,
  PRIMARY KEY (`id_instance`,`key`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

-- Instances, in which each instance is a single topic classifier
CREATE TABLE `tbl_instance` (
  `id_instance` varchar(10) NOT NULL,
  `name` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`id_instance`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

-- Our buffer, for application restarts
CREATE TABLE `tbl_processedtweets` (
  `id_instance` varchar(10) NOT NULL,
  `dt_tweet` varchar(50) DEFAULT NULL,
  `id_tweet` bigint(20) NOT NULL,
  `text` varchar(150) NOT NULL,
  `prior_similar` int(11) NOT NULL DEFAULT '0',
  `serialized_feature_vector` varchar(500) NOT NULL,
  `classifier_output` varchar(10) NOT NULL,
  PRIMARY KEY (`id_instance`,`id_tweet`),
  KEY `dt_tweet` (`dt_tweet`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

```

C.1.2 Database structure, schema: DataAnnotator

```

CREATE DATABASE IF NOT EXISTS `DataAnnotator`
USE `DataAnnotator`;

-- Relates which batches our linked to which user, and additional state props.
CREATE TABLE `lnk_user_batch` (
  `id_user_batch` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `id_user` int(10) unsigned NOT NULL DEFAULT '0',
  `id_batch` int(10) unsigned NOT NULL DEFAULT '0',
  `userbatch_state` enum('OPEN','DONE','HIDDEN_OPEN','HIDDEN_DONE') NOT NULL DEFAULT 'OPEN',
  `userbatch_dt_complete` datetime DEFAULT NULL,
  PRIMARY KEY (`id_user_batch`),
  KEY `INDEX` (`id_user`,`id_batch`,`userbatch_state`)
) ENGINE=MyISAM AUTO_INCREMENT=2321 DEFAULT CHARSET=utf8;

-- The actual annotations. Tweet- and topic-levels our stored as 0, 1 or 2.
CREATE TABLE `tbl_annotation` (
  `id_annotation` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `id_tweet` int(10) unsigned DEFAULT NULL,
  `id_user` int(10) unsigned DEFAULT NULL,
  `id_topic` int(10) unsigned DEFAULT NULL,
  `tweet_level` tinyint(4) DEFAULT NULL,
  `topic_level` tinyint(4) DEFAULT NULL,
  PRIMARY KEY (`id_annotation`),
  KEY `INDEX` (`id_tweet`,`id_user`,`id_topic`,`tweet_level`,`topic_level`)
) ENGINE=MyISAM AUTO_INCREMENT=107988 DEFAULT CHARSET=utf8;

-- Batches. Usually a batch consists of 15 tweets of tbl_tweet
CREATE TABLE `tbl_batch` (
  `id_batch` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `id_instance` int(10) unsigned DEFAULT NULL,

```

```

    `dt_start` datetime DEFAULT NULL,
    `dt_end` datetime DEFAULT NULL,
    `sample_ratio` double DEFAULT NULL,
    `dt_due` datetime DEFAULT NULL,
    PRIMARY KEY (`id_batch`),
    KEY `INDEX` (`id_instance`)
) ENGINE=MyISAM AUTO_INCREMENT=280 DEFAULT CHARSET=utf8;

-- Topics to be annotated for each instance and their names
CREATE TABLE `tbl_topic` (
  `id_topic` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `id_instance` int(10) unsigned DEFAULT NULL,
  `topic_name` varchar(45) DEFAULT NULL,
  `topic_sourcetable_id` int(10) unsigned DEFAULT NULL,
  PRIMARY KEY (`id_topic`),
  KEY `INDEX` (`id_instance`)
) ENGINE=MyISAM AUTO_INCREMENT=97 DEFAULT CHARSET=utf8;

-- The tweets to be annotated for each batch. May contain duplicates, if
-- several instances share a tweet. This removes the need for a separate
-- link table, and enables the use of instance-specific properties.
CREATE TABLE `tbl_tweet` (
  `id_tweet` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `id_batch` int(10) unsigned DEFAULT NULL,
  `tweet_id_twitter` bigint(20) DEFAULT NULL,
  `tweet_content` varchar(255) DEFAULT NULL,
  `tweet_postuser` varchar(80) DEFAULT NULL,
  `tweet_created` datetime DEFAULT NULL,
  `was_marked` bit(1) NOT NULL DEFAULT b'0',
  `was_noise` bit(1) NOT NULL DEFAULT b'0',
  `was_archived` bit(1) NOT NULL DEFAULT b'0',
  PRIMARY KEY (`id_tweet`),
  KEY `INDEX` (`id_batch`,`tweet_id_twitter`)
) ENGINE=MyISAM AUTO_INCREMENT=4198 DEFAULT CHARSET=utf8;

-- The annotating users.
CREATE TABLE `tbl_user` (
  `id_user` int(10) unsigned NOT NULL,
  `user_name` varchar(45) DEFAULT NULL,
  `user_email` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id_user`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

```

C.1.3 Database structure, schema: TwitterBase

```

CREATE DATABASE IF NOT EXISTS `TwitterBase`
USE `TwitterBase`;

-- A huge base for tweets
CREATE TABLE `TweetBase` (
  `id` bigint(20) unsigned NOT NULL,
  `is_processed` tinyint(4) unsigned NOT NULL DEFAULT '0',
  `process_result` tinyint(4) unsigned NOT NULL DEFAULT '0',
  `in_annotation` tinyint(4) unsigned NOT NULL DEFAULT '0',
  `in_gg` tinyint(4) unsigned NOT NULL DEFAULT '0',
  `in_npl` tinyint(4) unsigned NOT NULL DEFAULT '0',
  `in_tno2` tinyint(4) unsigned NOT NULL DEFAULT '0',
  `in_ns` tinyint(4) unsigned NOT NULL DEFAULT '0',
  `themed_gg` tinyint(4) unsigned NOT NULL DEFAULT '0',
  `themed_npl` tinyint(4) unsigned NOT NULL DEFAULT '0',
  `themed_tno2` tinyint(4) unsigned NOT NULL DEFAULT '0',
  `themed_ns` tinyint(4) unsigned NOT NULL DEFAULT '0',
  `created_at` datetime DEFAULT NULL,
  `text` varchar(145) NOT NULL,
  `source` varchar(60) NOT NULL,
  `truncated` tinyint(1) NOT NULL DEFAULT '0',

```

```

`in_reply_to_status_id` bigint(20) NOT NULL DEFAULT '0',
`in_reply_to_user_id` bigint(20) NOT NULL DEFAULT '0',
`in_reply_to_screen_name` varchar(40) NOT NULL,
`user_id` bigint(20) NOT NULL DEFAULT '0',
`longitude` varchar(20) NOT NULL,
`latitude` varchar(20) NOT NULL,
`place` varchar(550) NOT NULL,
`retweet_count` int(11) NOT NULL DEFAULT '0',
`favorite_count` int(11) NOT NULL DEFAULT '0',
`ent_hashtags` varchar(1000) NOT NULL,
`ent_symbols` varchar(255) NOT NULL,
`ent_urls` varchar(2000) NOT NULL,
`ent_user_mentions` varchar(2000) NOT NULL,
`ent_media` varchar(2000) NOT NULL,
`lang` varchar(10) NOT NULL,
PRIMARY KEY (`id`),
KEY `is_processed` (`is_processed`),
KEY `in_annotation` (`in_annotation`),
KEY `in_gg` (`in_gg`),
KEY `in_npl` (`in_npl`),
KEY `in_tno2` (`in_tno2`),
KEY `in_ns` (`in_ns`),
KEY `themed_gg` (`themed_gg`),
KEY `themed_npl` (`themed_npl`),
KEY `themed_tno2` (`themed_tno2`),
KEY `themed_ns` (`themed_ns`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

-- All users that have a tweet posted in TweetBase
CREATE TABLE `UserBase` (
  `id` bigint(20) unsigned NOT NULL,
  `is_processed` tinyint(4) unsigned NOT NULL DEFAULT '0',
  `process_result` tinyint(4) unsigned NOT NULL DEFAULT '0',
  `created_at` datetime DEFAULT NULL,
  `name` varchar(30) NOT NULL,
  `screen_name` varchar(20) NOT NULL,
  `location` varchar(50) NOT NULL,
  `description` varchar(100) NOT NULL,
  `url` varchar(100) NOT NULL,
  `ent_url` varchar(500) NOT NULL,
  `ent_description` varchar(500) NOT NULL,
  `protected` tinyint(1) NOT NULL DEFAULT '0',
  `followers_count` int(11) NOT NULL DEFAULT '0',
  `favorites_count` int(11) NOT NULL DEFAULT '0',
  `friends_count` int(11) NOT NULL DEFAULT '0',
  `listed_count` int(11) NOT NULL DEFAULT '0',
  `utc_offset` varchar(50) NOT NULL,
  `time_zone` varchar(30) NOT NULL,
  `geo_enabled` tinyint(1) NOT NULL DEFAULT '0',
  `statuses_count` int(11) NOT NULL DEFAULT '0',
  `lang` varchar(10) NOT NULL,
  `is_translator` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`),
  KEY `is_processed` (`is_processed`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- All IDs of tweets that have been annotated, primarily used as a temporary
-- table to construct MissingAnnotatedTweetIDs
CREATE TABLE `AnnotatedIDs` (
  `id` bigint(20) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

-- All IDs of tweets that were unable to be enriched because they have been
-- deleted or are protected.
CREATE TABLE `MissingAnnotatedTweetIDs` (

```



```

'id' bigint(20) NOT NULL,
PRIMARY KEY ('id')
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

-- A table of Twitter tokens to rotate on when importing and enriching large
-- amounts of data.
CREATE TABLE `Tokens` (
'id' int(11) NOT NULL AUTO_INCREMENT,
'consumer_key' varchar(100) NOT NULL,
'consumer_secret' varchar(100) NOT NULL,
'bearer_token' varchar(150) NOT NULL,
'owner' varchar(50) NOT NULL,
PRIMARY KEY ('id')
) ENGINE=MyISAM AUTO_INCREMENT=183 DEFAULT CHARSET=utf8;

```

C.1.4 Initial Project X analysis

```

USE twitcident_projectx;

CREATE OR REPLACE VIEW AnnotatedData AS
SELECT tweet.text,
       tweet.fromUser,
       tweet.createdAt,
       tweet.toUser,
       tweet.source,
       CONCAT(IF (tweet.latitude, tweet.latitude, ''),
              IF (tweet.longitude, CONCAT('_', tweet.longitude), ''),
              IF (tweet.location, CONCAT('_', tweet.location), ''),
              IF (tweet.placeName, CONCAT('_', tweet.placeName), ''),
              IF (tweet.placeCountry, CONCAT('_', tweet.placeCountry), '')) AS location_data,
       tweet.isoLanguageCode,
       tweet.stream AS stream_id,
       streamingrequest.name AS stream,
       theme.name AS theme,
       tweetLabels.name AS tweetLabel,
       userLabels.name AS userLabel
FROM ((( tweet
      LEFT JOIN label AS tweetLabels ON tweet.label = tweetLabels.id)
     LEFT JOIN label AS userLabels ON tweet.userLabel = userLabels.id)
    LEFT JOIN theme ON tweet.theme = theme.id)
   LEFT JOIN streamingrequest ON tweet.stream = streamingrequest.id)
WHERE (tweet.label IS NOT NULL)
ORDER BY tweet.createdAt;

SELECT * FROM AnnotatedData LIMIT 50000;

```

C.2 Implementation specifics

We have implemented the feature calculation and classifier system in Java, because the original Twitcident was implemented in Java, the large support for libraries like WEKA and the fact that the processes developed for this part run as consoled daemons. This part of the project has the following package structure:

TestBench This packages contains most classes with a *main()* routine to run the parts of the project, for development, example and evaluation purposes. Most parts of the project can individually be started, like the *FeatureCalculator*, *ClassificationTest*, *DataFinder* and *TrainingSetCreator*, but some classes are also used for library evaluation (for example *SimMetrics*¹, *SecondString*², *OpenNLP*³, ...)

¹<https://github.com/Simmetrics/simmetrics>

²<http://secondstring.sourceforge.net/>

³<https://opennlp.apache.org/>

as well as our own project parts (for example our own String Approximation algorithms).

MyShared This package contains generic classes, like our internal input Tweet representations, a Twitcident Data Connector that tunnels the database connection over SSH-tunnels and VPNs to access our data and the main project helpers with some configuration parameters like default settings and globally available helper functions.

WekaClassification This package contains WEKA-extensions and wrappers to connect and transform our input representations to WEKA's desired representation.

FeatureCalculation This package contains the actual Feature Calculation engine, its support classes, feature representation classes and the following subpackages:

Features This packages includes all the features we implemented for this project.

FeatureHelpers This packages includes support classes used by features, like stemmers, tense helpers, tokenizers, POS-taggers, ... These are often language specific (and thus configurable).

StringApproximationHelpers This contains a special subset of feature helpers regarding string approximation. It contains wrappers for the SimMetrics and SecondString libraries, as well as our own three string approximation algorithms.

FeatureInputs This package contain the feature input classes and their representations, like our word lists (and their parsers and database loaders) and hotspot lists.

This part of the system is fully database driven, meaning that about everything is configurable through the MySQL databases, including settings, parameters, feature definitions, feature inputs like word-lists, ... We considered this the most robust form of implementation, with other system parts being able to easily communicate with this part of the system while still being completely separate modules. It also allows for multiple programming languages to communicate through a shared interface.

The full database specification can be found in Appendix C.1.1. A noteworthy part is the specification and configuration of features, which our dynamically loaded, as depicted in Figure C.1.

Every feature is instantiated by the class given by *feature_class* that is passed four parameters: two optional variable field parameters and two optional *FeatureInput* classes. Each such *FeatureInput* is also given two variable field parameters. The interpretation of all variable field parameters depend on the feature class or the input class. All *FeatureInput* classes load their data from a table as depicted in Figure C.2.

id_instance	id_feature	feature_order	feature_enabled	feature_class	feature_param1	feature_param2	featureinput1_class	featureinput1_param1	featureinput1_param2	featureinput2_class	featureinput2_param1	featureinput2_param2
GG_ABVNG	13_MATCH_TWT_APPR_1	13	1	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_1NicksAlteredNGramDiceCoefficient	3	0.6	
GG_ABVNG	14_MATCH_TWT_APPR_2	14	1	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_2NicksAlteredNGramDiceCoefficient	3	0.6	
GG_ABVNG	15_MATCH_TWT_APPR_3	15	1	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_3NicksAlteredNGramDiceCoefficient	3	0.6	
GG_ABVNG	16_MATCH_TWT_APPR_4	16	0	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_4NicksAlteredNGramDiceCoefficient	3	0.6	
GG_ABVNG	17_MATCH_HPR_APPR_1	17	0	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_HYPER_1NicksAlteredNGramDiceCoefficient	3	0.6	
GG_ABVNG	18_MATCH_HPR_APPR_2	18	0	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_HYPER_2NicksAlteredNGramDiceCoefficient	3	0.6	
GG_ABVNG	19_MATCH_HPR_APPR_3	19	0	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_HYPER_3NicksAlteredNGramDiceCoefficient	3	0.6	
GG_ABVNG	20_MATCH_HPR_APPR_4	20	0	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_HYPER_4NicksAlteredNGramDiceCoefficient	3	0.6	
GG_ABVNG	21_MATCH_HPO_APPR_1	21	0	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_HYPO_1NicksAlteredNGramDiceCoefficient	3	0.6	
GG_ABVNG	22_MATCH_HPO_APPR_2	22	0	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_HYPO_2NicksAlteredNGramDiceCoefficient	3	0.6	
GG_ABVNG	23_MATCH_HPO_APPR_3	23	0	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_HYPO_3NicksAlteredNGramDiceCoefficient	3	0.6	
GG_ABVNG	24_MATCH_HPO_APPR_4	24	0	ApproximateMatchFeature	TEXT		FeatureInputs.WordList	WORDLIST_HYPO_4NicksAlteredNGramDiceCoefficient	3	0.6	
GG_ABVNG	41_IGNORE_TERM	41	1	IgnoreMatchFeature	TEXT		FeatureInputs.SimpleList	IGNORE_TERMS				
GG_ABVNG	42_IGNORE_USER	42	1	IgnoreMatchFeature	USER_NAME		FeatureInputs.SimpleList	IGNORE_USERS				
GG_ABVNG	47_SENTIMENT_POS	47	0	SentimentFeature	POSITIVE							
GG_ABVNG	48_SENTIMENT_NEG	48	0	SentimentFeature	NEGATIVE							
GG_ABVNG	52_IS_RETWEET	52	0	MatchesRegExFeature	TEXT	^RT.*						
GG_ABVNG	55_MENTION_PRE	55	1	MentionsFeature	PRE_MENTION							
GG_ABVNG	56_MENTION_INNER	56	1	MentionsFeature	INNER_MENTION							
GG_ABVNG	MENTION_POST	56	0	MentionsFeature	POST_MENTION							
GG_ABVNG	57_MENTION_NEWS	57	1	MentionsFeature	LIST_MENTION		FeatureInputs.SimpleList	USERLIST_MENTIONS				
GG_ABVNG	61_HASHTAG_INNER	61	1	HashtagsFeature	INNER_HASHTAG							
GG_ABVNG	62_HASHTAG_POST	62	1	HashtagsFeature	POST_HASHTAG							
GG_ABVNG	63_HASHTAG_LIST	63	1	HashtagsFeature	LIST_HASHTAG		FeatureInputs.SimpleList	HASHTAGS_1				
GG_ABVNG	64_SIMILAR_TWEETS	64	0	RecentSimilarFeature	0.51	25						
GG_ABVNG	66_HOTSPOT_DIST	66	1	HotspotDistanceFeature	1000000		FeatureInputs.HotspotList	HOTSPOTS				
GG_ABVNG	71_NOUNS	71	1	POSTagFrequencyFeature	NOUN							
GG_ABVNG	72_VERBS	72	1	POSTagFrequencyFeature	VERB							
GG_ABVNG	73_ADVERBS	73	1	POSTagFrequencyFeature	ADVERB							
GG_ABVNG	74_MENTDAY_FUTR	74	1	RelativeDayMentionedFeatu	1,2,3							
GG_ABVNG	75_MENTDAY_PAST	75	1	RelativeDayMentionedFeatu	0,6,5,4							
GG_ABVNG	76_SPECIALKW_1	76	1	CompoundMatchFeature	TEXT		FeatureInputs.WordList	SPECIALKW_POS				
GG_ABVNG	77_SPECIALKW_2	77	1	CompoundMatchFeature	TEXT		FeatureInputs.WordList	SPECIALKW_NEG				
GG_ABVNG	80_QUESTION_MARKS	80	1	RegExOccurrenceFeature	TEXT	\?						
GG_ABVNG	81_SUBSEQ_DOTS	81	1	RegExOccurrenceFeature	TEXT	[.]{2}						
GG_ABVNG	82_INTERPUNC_RATIO	82	1	InterpunctionRatioFeature								

Figure C.1: Database table: a random subset of a feature configuration

id_instance	id_featureinput	entry1	entry2	meta1	meta2
GG_ABVNG	HASHTAGS_1	aardbeving			
GG_ABVNG	HASHTAGS_1	trillen			
GG_ABVNG	HOTSPOTS	53.18605	6.56567	Groningen-Zuid	
GG_ABVNG	HOTSPOTS	53.21937	6.51915	Groningen-West	
GG_ABVNG	HOTSPOTS	53.2207	6.61528	Groningen-Oost	
GG_ABVNG	HOTSPOTS	53.23869	6.56653	Groningen-Noord	
GG_ABVNG	WORDLIST_1	{aardbeving}(5)			
GG_ABVNG	WORDLIST_1	{aardschok}(3)			
GG_ABVNG	WORDLIST_1	{beving}(3)			
GG_ABVNG	WORDLIST_1	{schok}(1)			
GG_ABVNG	WORDLIST_2	{aardbeving}{zware}(2)			
GG_ABVNG	WORDLIST_2	{aardbevinkje}(2)			
GG_ABVNG	WORDLIST_2	{aardschokje}(2)			
GG_ABVNG	WORDLIST_2	{beeft}{angst}(3)			
GG_ABVNG	WORDLIST_2	{beven}{angst}(2)			
GG_ABVNG	WORDLIST_2	{rammelt}{honger}(2)			

Figure C.2: Database table: a random subset of some feature inputs

C.3 Word Expansion Algorithm

Algorithm 3 Expanding a word using Wiktionary API

```

1: procedure EXPANDWORDUSINGWIKTIONARY(  $W, R, [D = \text{true}]$  )  $\triangleright W = \text{Word}$ 
   ,  $R$  is an expandable return object
2:    $S \leftarrow \text{text for } W$ 
3:    $S \leftarrow \text{find Dutch section using regular expressions in } S$ 
4:    $T \leftarrow \text{templates for } W$ 
5:   if 'nld'  $\notin T \wedge$  '=nld='  $\notin T$  then
6:     return  $R$   $\triangleright$  No Dutch section, so return  $R$  unaffected
7:   end if
8:    $R[\text{word}] \leftarrow W$   $\triangleright$  Add word to output object
9:   if  $D \wedge$  ('noun-pl'  $\in T \vee$  'noun-dim'  $\in T \vee$  'noun-dim-pl'  $\in T \vee$ 
'conjugatable'  $\in T$ ) then  $\triangleright$  This word is a conjugated form of another base word
10:     $A \leftarrow \text{obtain the base word this word is derived from using regular expressions}$ 
11:     $R[\text{derived\_from}] \leftarrow \text{EXPANDWORD}(A, [], \text{false})$   $\triangleright$  This is partially
recursive
12:    end if
13:    if 'verb'  $\in T$  then  $\triangleright$  This word is a verb
14:       $R[\text{type}] \leftarrow R[\text{type}] \cup \{ \text{'verb'} \}$ 
15:      if 'conjugatable'  $\notin T$  then
16:         $R[\text{verbal\_forms}] \leftarrow \text{scrape from '/vervoeging' and parse verbal forms using reg. expr.}$ 
17:      end if
18:       $Section \leftarrow \text{extract content for section 'Werkwoord' from } S \text{ using regular expressions}$ 
19:       $A \leftarrow \text{parse definitions from } Section \text{ using regular expressions}$ 
20:       $R[\text{defs}] \leftarrow R[\text{defs}] \cup A$ 
21:    end if
22:    if 'noun'  $\in T$  then  $\triangleright$  This word is a noun
23:       $R[\text{type}] \leftarrow R[\text{type}] \cup \{ \text{'noun'} \}$ 
24:      if 'noun-pl'  $\in T \vee$  'noun-dim'  $\in T \vee$  'noun-dim-pl'  $\in T$  then
25:         $R[\text{plural}] \leftarrow R[\text{plural}] \cup \text{parse plural form from } S \text{ using regular expressions}$ 
26:         $R[\text{dim.}][\text{sing.}] \leftarrow R[\text{dim.}][\text{sing.}] \cup \text{parse diminutive singular form from } S \text{ using reg. expr.}$ 
27:         $R[\text{dim.}][\text{plur.}] \leftarrow R[\text{dim.}][\text{plur.}] \cup \text{parse diminutive plural form from } S \text{ using reg. expr.}$ 
28:      end if
29:       $Section \leftarrow \text{extract content for section 'Zelfstandig naamwoord' from } S \text{ using reg. expr.}$ 
30:       $A \leftarrow \text{parse definitions from } Section \text{ using regular expressions}$ 
31:       $R[\text{defs}] \leftarrow R[\text{defs}] \cup A$ 
32:    end if
33:    if '-syn-'  $\in T$  then  $\triangleright$  This word has synonyms
34:       $Section \leftarrow \text{extract content for section 'Synoniemen' from } S \text{ using regular expressions}$ 
35:       $A \leftarrow \text{parse definitions from } Section \text{ using regular expressions}$ 
36:       $R[\text{synonyms}] \leftarrow R[\text{synonyms}] \cup A$ 
37:    end if

```

Algorithm 4 Expanding a word using Wiktionary API (continued, 1)

```

38:   if '-rel-'  $\in T$  then                                ▷ This word has related terms
39:      $Section \leftarrow$  extract content for section 'Verwante begrippen' from  $S$  using reg. expr.
40:      $A \leftarrow$  parse definitions from  $Section$  using regular expressions
41:      $R[\text{related}] \leftarrow R[\text{related}] \cup A$ 
42:   end if
43:   if '-hypo-'  $\in T$  then                                ▷ This word has hyponyms
44:      $Section \leftarrow$  extract content for section 'Hyponiemen' from  $S$  using regular expressions
45:      $A \leftarrow$  parse definitions from  $Section$  using regular expressions
46:      $R[\text{hyponyms}] \leftarrow R[\text{hyponyms}] \cup A$ 
47:   end if
48:   if '-hyper-'  $\in T$  then                               ▷ This word has hypernyms
49:      $Section \leftarrow$  extract content for section 'Hyperniemen' from  $S$  using regular expressions
50:      $A \leftarrow$  parse definitions from  $Section$  using regular expressions
51:      $R[\text{hypernyms}] \leftarrow R[\text{hypernyms}] \cup A$ 
52:   end if
53:   if '-typ-'  $\in T$  then                                 ▷ This word has typical usages
54:      $Section \leftarrow$  extract content for section 'Typische woordcombinaties' from  $S$  using reg. expr.
55:      $A \leftarrow$  parse definitions from  $Section$  using regular expressions
56:      $R[\text{usages}] \leftarrow R[\text{usages}] \cup A$ 
57:   end if
58:   if '-expr-'  $\in T$  then                               ▷ This word has expressions
59:      $Section \leftarrow$  extract content for section 'Uitdrukkingen en gezegden' from  $S$  using reg. expr.
60:      $A \leftarrow$  parse definitions from  $Section$  using regular expressions
61:      $R[\text{proverbs}] \leftarrow R[\text{proverbs}] \cup A$ 
62:   end if
63:   if '-prov-'  $\in T$  then                               ▷ This word has proverbs
64:      $Section \leftarrow$  extract content for section 'Spreekwoorden' from  $S$  using regular expressions
65:      $A \leftarrow$  parse definitions from  $Section$  using regular expressions
66:      $R[\text{proverbs}] \leftarrow R[\text{proverbs}] \cup A$ 
67:   end if
68:   return  $R$ 
69: end procedure

```

Appendix D

Potential data sources

Name	Why useful / Description
Van Dale Woordenboek + Thesaurus	Provide not only a few synonyms and related words, but also instances in which the word can be used. At least one such instance defines the semantic relation and/or order of words we are looking for. Also useful if the singular form is not contained in the plural form: both are listed.
http://www.mijnwoordenboek.nl/ww.php	This source is able to conjugate verbs to all applicable forms, which might be very useful due to the irregularity of some verbs.
woorden.org	Another public online dictionary.

Table D.1: Setup sources: dictionaries

Name	Why useful / Description
nl.wiktionary.org	A very special and important source. Can be used as a dictionary like described above, but also provides semantic relations to other words like synonyms, hyponyms, antonyms and derived compound words. Licensed under Creative Commons.
Cornetto	Dutch alternative to WordNet. Only free for non-commercial academic purposes.
WikiPedia	The category structure and semantic relationships may be exploited in some way or another.

Table D.2: Setup sources: semantic sources

Name	Why useful / Description
synoniemen.net	Provides synonyms, but also reversed: words that contain the input word as a synonym. This feature could be used to enhance the confidence in returned synonyms.
puzzelwoordenboek.nl/woordenboek/	A puzzle dictionary for synonyms. Given a word, this source returns all similar words / synonyms ordered by the number of letters in the returned word.
mijnwoordenboek.nl/puzzelwoordenboek	This is another puzzle dictionary, but does not return actual synonyms, but more very closely related words.

Table D.3: Setup sources: synonyms

Name	Why useful / Description
WikiPedia	General encyclopedia. Word frequency of related content might be useful.
DBPedia	More structured version of Wikipedia. We might be able to identify certain relationships or certain types of relationships we could exploit.
Encyclo.org	Another encyclopedia.

Table D.4: Setup sources: encyclopedia

Name	Why useful / Description
spreekwoord.nl	A source of proverbs.
woorden.org/spreekwoord.php	Another source of proverbs .
dbnl.org/tekst/stoe002nede01_01/	Another proverb source; with all proverbs listed on a single page.

Table D.5: Setup sources: proverbs

Name	Why useful / Description
woordenlijst.org	Contains valid word listings; we could use this to identify terms.
Google Translate	Google Translate provides machine-learned alternatives/synonyms that we could use. Furthermore, it provides several example cases in which the word is used. We could also translate back-and-forth to obtain alternatives. We could also exploit Google Translate to get the English version, consult major English dictionaries like we use the Dutch (i.e. thesaurus.org), and translate the results back to Dutch.
Twitter	Although it seems a bit paradoxical, we might search the current term on Twitter to obtain related terms / terms the our often used if our current term is used, and ‘verify’ the usability and context of these terms using our other source-types. Twitter also provides related terms, but this seems to be very limited and not useful enough.
Google Images	I noticed that when I was searching for a keyword I was looking for on Google Images, that most of the time the keyword appears in the context we are looking for. This makes sense, since we are mostly looking for disturbances and discrepancies to which some party has to respond; something that people find most interesting. We might acquire relevant tags/terms from those images?
taalkabaal.nl/scheldwoorden/indexa.php	Contains a list of invectives and abuse words. These maybe could be used for sentiment analysis.
voorbegginers.info/bargoens/alfabetisch-a-m.htm	Contains a list of Dutch Slang words and their descriptions.
chatwoordenboek.nl	Contains chat slang and abbreviations. Might be useful for interpreting tweets.
taalkabaal.nl/turbotaal	Contains modern slang proverbs; can be used in the same way as ‘official’ proverbs.

Table D.6: Setup sources: miscellaneous

Appendix E

Example word expansions

E.1 Aardbeving

```
word = aardbeving
type = [ 0 = noun ]
defs = [ 0 = keer dat de grond schudt ]
plural = [ 0 = aardbevingen ]
diminutive = [
  singular = [ 0 = aardbevinkje ]
  plural = [ 0 = aardbevinkjes ]
]
synonyms = [
  0 = aardschok
  1 = beving
]
related = [ 0 = aarde ]
usages = [ 0 = grote verwoestingen door een zware aardbeving ]
```

E.2 Auto

```
word => "auto"
type => [ 0 => "noun" ]
defs => [ 0 => "motorrijtuig op vier wielen" ]
plural => [ 0 => "auto's" ]
diminutive => [
  singular => [ 0 => "autootje" ]
  plural => [ 0 => "autootjes" ]
]
synonyms => [
  0 => "automobiel"
  1 => "wagen"
]
hyponyms => [
  0 => "afroepauto"
  1 => "bankauto"
  2 => "bedrijfsauto"
  3 => "bestelauto"
  4 => "blusauto"
  5 => "boemauto"
  6 => "bomauto"
  7 => "boodschappenauto"
  8 => "botsauto"
  9 => "brandweerauto"
  10 => "bromauto"
  11 => "conceptauto"
  12 => "deelauto"
```

```

13 => "dienstauto"
14 => "dieselauto"
15 => "driedeursauto"
16 => "geldauto"
17 => "gezinsauto"
18 => "huurauto"
19 => "hybrideauto"
20 => "inruilauto"
21 => "kampeerauto"
22 => "kiepauto"
23 => "koelauto"
24 => "koersauto"
25 => "kraanauto"
26 => "ladderauto"
27 => "leaseauto"
28 => "lesauto"
29 => "lijkauto"
30 => "loopauto"
31 => "luxeauto"
32 => "maanauto"
33 => "melkauto"
34 => "miniaturauto"
35 => "modelauto"
36 => "pantserauto"
37 => "personenauto"
38 => "politieauto"
39 => "postauto"
40 => "raceauto"
41 => "rallyauto"
42 => "ruimteauto"
43 => "rupsauto"
44 => "sleepauto"
45 => "sloopauto"
46 => "speelgoedauto"
47 => "sportauto"
48 => "stadsauto"
49 => "strooiauto"
50 => "takelauto"
51 => "tankauto"
52 => "terreinauto"
53 => "testauto"
54 => "trapauto"
55 => "tweedeursauto"
56 => "verhuisauto"
57 => "vervangauto"
58 => "vijfdeursauto"
59 => "vluchtauto"
60 => "volgauto"
61 => "vrachtauto"
62 => "vuilnisauto"
63 => "waterstofauto"
64 => "zakenauto"
65 => "zandauto"
66 => "ziekenauto"
67 => "zonneauto"
68 => "zwerfauto"
]
usages => [
0 => "met de auto naar je werk rijden"
1 => "een auto huren op het vliegveld"
2 => "de auto parkeren"
3 => "leaseauto"
]
proverbs => [
0 => "de auto is voor velen een heilige koe. (=iets heiligs, waar je niet aan mag komen.)"
1 => "een vogel in de auto rijden (=elk geval kan overal mee leven)"
]

```

```
]
```

E.3 Bank

```
word => "bank"
type => [ 0 => "noun" ]
defs => [ 0 => "zitmeubel voor meer dan één persoon" ]
plural => [ 0 => "banken" ]
diminutive => [
  singular => [ 0 => "bankje" ]
  plural => [ 0 => "bankjes" ]
]
synonyms => [
  0 => "canapé"
  1 => "divan"
  2 => "sofa"
  3 => "zandbank"
  4 => "zitbank"
]
related => []
hyponyms => [
  0 => "aanrechtbank"
  1 => "ab-bank"
  2 => "achterbank"
  3 => "adressenbank"
  4 => "afstootbank"
  5 => "afvoerbank"
  6 => "arbeidersbank"
  7 => "asbank"
  8 => "bedbank"
  9 => "beeldbank"
  10 => "beenmergbank"
  11 => "beklaagdenbank"
  12 => "beleenbank"
  13 => "bidbank"
  14 => "bierbank"
  15 => "bijbank"
  16 => "bloedbank"
  17 => "boorbank"
  18 => "bosbank"
  19 => "botralibank"
  20 => "circulatiebank"
  21 => "communiebank"
  22 => "databank"
  23 => "depositobank"
  24 => "discontobank"
  25 => "doggersbank"
  26 => "donorbank"
  27 => "draaibank"
  28 => "driezitsbank"
  29 => "effectenbank"
  30 => "eurobank"
  31 => "europabank"
  32 => "financieringsbank"
  33 => "folterbank"
  34 => "freesbank"
  35 => "gegevensbank"
  36 => "genenbank"
  37 => "geselbank"
  38 => "getuigenbank"
  39 => "glijbank"
  40 => "grindbank"
  41 => "grondbank"
  42 => "hakbank"
  43 => "hakselbank"
```

44 => "handelsbank "
45 => "herenbank "
46 => "hersensbank "
47 => "hoekbank "
48 => "holbank "
49 => "houtdraaibank "
50 => "hypotheebank "
51 => "incassobank "
52 => "informatiebank "
53 => "informatiseringsbank "
54 => "investeringsbank "
55 => "kaaibanbank "
56 => "kannenbank "
57 => "kennisbank "
58 => "kerfbank "
59 => "kerkbank "
60 => "klachtenbank "
61 => "klapbank "
62 => "kniebank "
63 => "koeienbank "
64 => "koorbank "
65 => "koraalbank "
66 => "kotterbank "
67 => "kredietbank "
68 => "kruispuntbank "
69 => "landbouwbank "
70 => "lattenbank "
71 => "leenbank "
72 => "ligbank "
73 => "loungebank "
74 => "mastbank "
75 => "melkbank "
76 => "mestbank "
77 => "mistbank "
78 => "modderbank "
79 => "mosselbank "
80 => "nagelbank "
81 => "nevelbank "
82 => "oerbank "
83 => "oesterbank "
84 => "onderbank "
85 => "ontwikkelingsbank "
86 => "oppositiebank "
87 => "organenbank "
88 => "pandenbank "
89 => "parelbank "
90 => "pijnbank "
91 => "Postbank "
92 => "pottenbank "
93 => "rabobank "
94 => "raiffeisenbank "
95 => "rechtbank "
96 => "regeringsbank "
97 => "rekbank "
98 => "reservebank "
99 => "retailbank "
100 => "richtbank "
101 => "roeibank "
102 => "rolbank "
103 => "rollenbank "
104 => "rollerbank "
105 => "rotsbank "
106 => "rustbank "
107 => "schaafbank "
108 => "schelpenbank "
109 => "schepenbank "

```

110 => "schommelbank"
111 => "schoolbank"
112 => "schroefbank"
113 => "slaapbank"
114 => "slachtbank"
115 => "slagersbank"
116 => "sneeuwbank"
117 => "snijbank"
118 => "sollicitantenbank"
119 => "spaarbank"
120 => "speelbank"
121 => "spelersbank"
122 => "spermabank"
123 => "staatsbank"
124 => "stadsbank"
125 => "strafbank"
126 => "systeembank"
127 => "tekstbank"
128 => "thuisbank"
129 => "tijdbank"
130 => "tijgerbank"
131 => "toonbank"
132 => "trekbank"
133 => "tuinbank"
134 => "tweezitsbank"
135 => "vacaturebank"
136 => "vandiktebank"
137 => "vensterbank"
138 => "verdachtenbank"
139 => "verzekeringsbank"
140 => "visbank"
141 => "vlakbank"
142 => "vleesbank"
143 => "voedselbank"
144 => "voetbank"
145 => "volkskredietbank"
146 => "waterbank"
147 => "weefselbank"
148 => "Wereldbank"
149 => "werkbank"
150 => "wisselbank"
151 => "wolkenbank"
152 => "zaadbank"
153 => "zaagbank"
154 => "zadenbank"
155 => "zakenbank"
156 => "zandbank"
157 => "zeebank"
158 => "zetbank"
159 => "zitbank"
160 => "zodenbank"
161 => "zondaarsbank"
162 => "zonnebank"
]
hypernyms => []
usages => [
  0 => "tv kijken op de bank"
  1 => "kerkbank"
  2 => "een bankje op een mooi punt van het wandelpad"
]
proverbs => [
  0 => " op de bank zitten"
  1 => "als een lam ter slachtbank geleid worden (=weerloos zijn)"
  2 => "als warme/hete broodjes over de toonbank gaan. (=zeer goed verkopen)"
  3 => "door de bank genomen. (=gemiddeld; meestal; gewoonlijk.)"
  4 => "het is kruis of munt, zei de non en ze trouwde de bankier. (=een keuze voor het materiële kan

```

```

5 => "het niet onder stoelen of banken steken (=je niet stil houden, maar je mening openlijk uite
6 => "iemand achter de bank schuiven (=iemand minachtend behandelen)"
7 => "iemand op de pijnbank leggen (=iemand het moeilijk maken en daarmee dwingen iets te doen)"
8 => "je mening niet onder stoelen of banken steken. (=je mening niet verbergen, openlijk voor je
9 => "niet onder stoelen en banken steken (=er rond voor uitkomen)"
10 => "niet onder stoelen en of banken steken (=er rond voor uitkomen)"
11 => "onder stoelen of banken steken (=verbergen)"
12 => "op het zondaarsbankje zitten (=schuld bekennen)"
13 => "voor stoelen en banken praten (=maar weinigen die naar iemands verhaal luisteren)"
14 => "zo zeker als de bank (=iemand die in alles te vertrouwen is)"
]

```

E.4 Brand

```

word => "brand"
type => [
  0 => "verb"
  1 => "noun"
]
defs => [
  0 => "eerste persoon enkelvoud tegenwoordige tijd van branden
      Ik brand."
  1 => "gebiedende wijs van branden
      Brand!"
  2 => "(bij inversie) tweede persoon enkelvoud tegenwoordige tijd van branden
      Brand je?"
  3 => "keer dat vuur iets verbrandt"
]
derived_from => [
  word => "branden"
  type => [
    0 => "verb"
    1 => "noun"
  ]
  defs => [
    0 => "(absoluut) verteerd worden door vuur
        De kaars brandde de hele nacht"
    1 => "(overgankelijk) aan vuur blootstellen, roosteren
        De koffie werd er vrij licht gebrand."
    2 => "(overgankelijk) als brandstof gebruiken
        De olie die daar gebrand wordt, stinkt."
  ]
  verbal_forms => [
    0 => "gebrand"
    1 => "brandend"
    2 => "branden"
    3 => "brand"
    4 => "brandt"
    11 => "brandde"
    16 => "brandden"
  ]
  related => [ 0 => "verbranden" ]
]
plural => [ 0 => "branden" ]
dimunitive => [
  singular => [ 0 => "brandje" ]
  plural => [ 0 => "brandjes" ]
]
synonyms => [
  0 => "fik"
  1 => "hens"
]
related => [ 0 => "vuurzee" ]
hyponyms => [
  0 => "aardbrand"
]

```



```

1 => "bedrijfsbrand"
2 => "bermbrand"
3 => "binnenbrand"
4 => "boombrand"
5 => "bosbrand"
6 => "builenbrand"
7 => "duinbrand"
8 => "haardbrand"
9 => "heidebrand"
10 => "Hildebrand"
11 => "IJsbrand"
12 => "natuurbrand"
13 => "rijstbrand"
14 => "roggebrand"
15 => "schipholbrand"
16 => "schoorsteenbrand"
17 => "stadsbrand"
18 => "steenbrand"
19 => "stokebrand"
20 => "stuifbrand"
21 => "veenbrand"
22 => "voorbrand"
23 => "vriesbrand"
24 => "wereldbrand"
25 => "wortelbrand"
26 => "zeebrand"
27 => "zonnebrand"
28 => "zuurbrand"
]
hypernyms => []
usages => [
  0 => "in brand staan"
  1 => "Er is brand uitgebroken."
  2 => "bosbranden"
]
proverbs => [
  0 => "as is verbrande turf. (=aan een belofte (as = als) heb je niets)"
  1 => "bang zijn zich aan koud water te branden (=erg voorzichtig zijn)"
  2 => "beter hard geblazen dan de mond gebrand. (=het is beter dat men zich inspant dan dat er door s"
  3 => "branden als een (tiere)lier (=een heel erg hevige brand)"
  4 => "branden als een fakkel (=zeer fel branden)"
  5 => "brandende kwestie (=een dringende, actuele zaak)"
  6 => "de schepen achter zich verbranden (=een besissing nemen en niet meer terug kunnen)"
  7 => "dominee brand je bekje niet (=pas op! Het eten of de drank is heet!)"
  8 => "ergens op gebrand zijn (=iets heel erg fijn vinden en er naar streven)"
  9 => "gauw aangebrand zijn (=gauw geïrriteerd zijn)"
  10 => "het geld brandt hem in de zak. (=hij geeft zijn geld graag en gemakkelijk uit.)"
  11 => "iemand uit de brand helpen (=iemand uit de nood helpen)"
  12 => "kijken of men water ziet branden (=heel erg verbaasd kijken)"
  13 => "moord en brand schreeuwen (=uiterst verontwaardigd zijn)"
  14 => "niet brandschoon zijn (=dingen misdaan hebben)"
  15 => "ook tussen de mooie bloemen groeien brandnetels. (=de schoonheid van de omgeving biedt geen g"
  16 => "te veel vuur in een stoof doet ze branden (=teveel is schadelijk)"
  17 => "uit de brand helpen (=uit de nood helpen)"
  18 => "uit de brand zijn (=geholpen zijn, problemen opgelost)"
  19 => "wie zijn billen brandt, moet op de blaren zitten. (=als je iets doms doet, moet je de gevolge"
  20 => "wie zijn gat brandt, moet op de blaren zitten. (=wie een risico neemt, moet de gevolgen drage"
  21 => "zijn kaars aan twee kanten branden (=zijn krachten of mogelijkheden al te vroeg verspillen)"
  22 => "zijn schepen achter zich verbranden (=obstinaat doorgaan, zodanig dat men niet meer terug kan"
  23 => "zijn vingers aan iets branden (=zich in iets vergissen, nadeel aan iets ondervinden)"
]

```

E.5 Bureau

```
word => "bureau"
```

```

type => [ 0 => "noun" ]
defs => [
  0 => "tafel met laden om aan te werken"
  1 => "kantoor van de politie of andere organisatie"
  2 => "werkkamer"
]
plural => [ 0 => "bureaus" ]
diminutive => [
  singular => [ 0 => "bureautje" ]
  plural => [ 0 => "bureautjes" ]
]
synonyms => [
  0 => "schrijftafel"
  1 => "schrijfbureau"
  2 => "kantoor"
]
hyponyms => [
  0 => "adresbureau"
  1 => "advertentiebureau"
  2 => "adviesbureau"
  3 => "arbeidsbureau"
  4 => "architectenbureau"
  5 => "atoombureau"
  6 => "bagagebureau"
  7 => "bedrijfsbureau"
  8 => "bemiddelingsbureau"
  9 => "classificatiebureau"
  10 => "consultatiebureau"
  11 => "detacheringsbureau"
  12 => "districtsbureau"
  13 => "escortbureau"
  14 => "evenementenbureau"
  15 => "gastouderbureau"
  16 => "headhuntersbureau"
  17 => "hoofdbureau"
  18 => "incassobureau"
  19 => "informatiebureau"
  20 => "ingenieursbureau"
  21 => "interimbureau"
  22 => "kamerverhuurbureau"
  23 => "mailbureau"
  24 => "merkenbureau"
  25 => "modellenbureau"
  26 => "persbureau"
  27 => "plaatsingsbureau"
  28 => "planbureau"
  29 => "politiebureau"
  30 => "politiebureau"
  31 => "projectbureau"
  32 => "reclamebureau"
  33 => "reisbureau"
  34 => "relatiebureau"
  35 => "schrijfbureau"
  36 => "stembureau"
  37 => "terugkeerbureau"
  38 => "toeristenbureau"
  39 => "vaccinebureau"
  40 => "zorgbureau"
]
usages => [
  0 => "aan/achter je bureau zitten schrijven"
  1 => "politiebureau"
  2 => "adviesbureau"
  3 => "Komt u even langs op mijn bureau."
]

```

E.6 Regen

```

word => "regen"
type => [
  0 => "verb"
  1 => "noun"
]
defs => [
  0 => "meervoud verleden tijd van rijgen
      Wij regen.
      Jullie regen.
      Zij regen."
  1 => "gebiedende wijs van regenen"
  2 => "water dat in druppels uit de lucht valt"
  3 => "regenbui"
]
derived_from => [
  word => "rijgen"
  type => [ 0 => "verb" ]
  defs => [ 0 => "(overgankelijk) met een naald een draad ergens doorvoeren
              Ze reeg eerst de zoom om te kunnen zien of deze op de juiste lengte was." ]
  verbal_forms => [
    0 => "geregen"
    1 => "rijgend"
    2 => "rijgen"
    3 => "rijg"
    4 => "rijgt"
    11 => "reeg"
    14 => "reegt"
    16 => "regen"
  ]
  related => [ 0 => "rijgnaald" ]
]
plural => [ 0 => "regens" ]
diminutive => [
  singular => [ 0 => "regentje" ]
  plural => [ 0 => "regentjes" ]
]
hyponyms => [
  0 => "druilregen"
  1 => "goudenregen"
  2 => "goudregen"
  3 => "ijsregen"
  4 => "lintjesregen"
  5 => "motregen"
  6 => "prijzenregen"
  7 => "slagregen"
  8 => "sterrenregen"
  9 => "stofregen"
  10 => "stortregen"
  11 => "vonkenregen"
  12 => "zandregen"
  13 => "zwavelregen"
]
proverbs => [
  0 => "Van de regen in de drup."
  1 => "Regen in mei, dan is april voorbij."
  2 => "als het in de kajuit regent druipt het in de hut (=als de baas problemen heeft, krijgen ook de
  3 => "als het melk regent, staan mijn schotels omgekeerd. (=wanneer ergens iets voordeligs te verkrij
  4 => "als honden konden bidden zou het kluiven regenen. (=als is een niet ter zake doende opmerking.
  5 => "Avondrood, mooi weer aan boord, morgenrood, regen in de sloot (=Eerste deel is 60% waar, tweed
  6 => "de boon van de koek gekregen hebben (=geluk gehad hebben)"
  7 => "de regen schuwen en in de sloot vallen (=door iets onaangenaams te ontwijken in nog groter pro
  8 => "het regent bakstenen (=gezegd van een hevige hagelbui)"
  9 => "het regent pijpestelen (=het regent heel hard)"
  10 => "hij heeft een klap van de molen gekregen. (=hij is niet goed meer bij zijn verstand)"
]

```

```

11 => "na regen komt zonneschijn. (=na een periode van tegenslag, komt er een betere tijd.)"
12 => "regen in mei, dan is april voorbij. (=de natuur kiest vanzelf de goede volgorde.)"
13 => "van de regen in de drup (=niet veel opschieten, van moeilijke omstandigheden in nog moeilij
14 => "verrijzen als paddestoelen na een regenachtige dag (=plots tevoorschijn komen)"
]
usages => [
  0 => "in de stromende regen"
  1 => "tropische regens"
]

```

E.7 Trillen

```

word => "trillen"
type => [ 0 => "verb" ]
defs => [
  0 => "(inergatief) snel heen een weer bewegen
      De snaar trilde totdat de harpist deze met zijn hand afdempte."
  1 => "heel snel met kleine bewegingen heen en weer gaan"
]
verbal_forms => [
  0 => "getrild"
  1 => "trillend"
  2 => "trillen"
  3 => "tril"
  4 => "trilt"
  11 => "trilde"
  16 => "trilden"
]
synonyms => [
  0 => "vibreren"
  1 => "bibberen"
  2 => "beven"
  3 => "rillen"
]
usages => [ 0 => "trillen van woede" ]

```

E.8 Veeg

```

word => "veeg"
type => [
  0 => "verb"
  1 => "noun"
]
defs => [
  0 => "eerste persoon enkelvoud tegenwoordige tijd van vegen
      Ik veeg."
  1 => "gebiedende wijs van vegen
      Veeg!"
  2 => "(bij inversie) tweede persoon enkelvoud tegenwoordige tijd van vegen
      Veeg je?"
  3 => "beweging waarbij je langs iets strijkt"
  4 => "vlek in de vorm van een streep die je maakt door ergens langs te vegen"
]
derived_from => [
  word => "vegen"
  type => [
    0 => "verb"
    1 => "noun"
  ]
  defs => [
    0 => "(overgankelijk) zonder water schoonmaken met een borstel
        Vergeet niet de vloer nog te vegen!"
    1 => "(figuurlijk) door ergens langs te strijken verplaatsen of verwijderen"
  ]
]

```

```

verbal_forms => [
  0 => "geveegd"
  1 => "vegend"
  2 => "vegen"
  3 => "veeg"
  4 => "veegt"
  11 => "veegde"
  16 => "veegden"
]
related => [ 0 => "opvegen" ]
usages => [
  0 => "de vloer vegen"
  1 => "een schoorsteen vegen"
  2 => "je voeten aan de deurmat vegen"
  3 => "de tranen van je wangen vegen"
]
proverbs => [
  0 => " een voorstel van tafel vegen"
  1 => " onder het tapijt vegen"
  2 => " van de kaart vegen"
]
]
plural => [ 0 => "vegen" ]
diminutive => [
  singular => [ 0 => "veegje" ]
  plural => [ 0 => "veegjes" ]
]
usages => [
  0 => "Met één veeg alles uitwissen"
  1 => "Er zit een zwarte veeg op je mouw."
]
]
proverbs => [
  0 => "een veeg uit de pan krijgen (=een klap incasseren / op zijn donder krijgen / een standje krijgen)"
  1 => "een voetveeg zijn. (=iemand zijn die voor minderwaardige klusjes gebruikt wordt.)"
  2 => "iemand's voetveeg zijn (=iemand's slaaf zijn (zich alles moeten laten welgevallen))"
  3 => "zo veeg als een luis op een kam (=in groot gevaar verkerend)"
]
]

```


Appendix F

Data Collection

To evaluate our Classifier performance, we need training and test data of which the classifications are known a priori. This section describes how we obtained our classified data, and how it is processed. Section F.1 will go more into the problem, and discusses the choices we made. Section F.2 will be dedicated to the Data Annotation Application. Next, in Section F.3 we will discuss the Data Collection algorithm. Finally, in Section F.4 we will discuss our output processing algorithm on how we aggregated our results.

The goal here is to acquire a fully classified representative dataset to evaluate our classifiers. We will use the terms classify and annotate interchangeably.

F.1 Strategy Considerations

F.1.1 What?

With Observation 9 we have noticed that the signal we are trying to detect is extremely low, so the first question to arise is: How are we going to construct our dataset if the relevancy ratio is so low? If we would perform a random unbiased search, we cannot expect to have any relevant entries in our dataset.

That is why we will use the existing Twitcident databases to obtain our data. These databases contain all tweets that have been streamed, even unclassified ones (recall Section 2.1). And even among the originally classified data, the relevancy ratio is very low (hence our thesis work). This also enables us to evaluate real-life operational instances, rather than some random generated instance.

F.1.2 Who?

In Section 2.5 we have seen that the problem involves a very difficult domain. Even domain experts have difficulties annotating a tweet and judging whether it is relevant. Furthermore, detailed knowledge of what the client wants with a Twitcident Instance is extremely important and often hard to grasp. Without this knowledge, it is even for CrowdSense employees nearly impossible to annotate.

Due to this fact in combination with the data sensitivity (client data may not be published) we were unable to employ CrowdSourcing, and lead us to do the annotation

work ourselves within the CrowdSense team. After having our internal annotation consensus analyzed, we consider this choice justified.

F.1.3 How much?

The maximum allowed effort available per employee was about ten hours, or five minutes per day over a period of half a year, with an average of six employees. Furthermore, we wanted each tweet to be annotated by at least three to four employees, due to the problem being hard and to achieve some consensus aggregation.

Our five minutes per day restriction translates to annotating a batch of about 15 tweets to be annotated. Therefore, we split up our work in packages of 15 tweets called *batches*. A batch always relates to a single instance to focus on the instance goal. Every tweet presented must be annotated for all Twitcident Topics within the instance: we will elaborate on this process in Section F.2.

Because the amount of work that can be done is limited, we had to set our dataset limits as low as possible where we can. Therefore, we aimed at the following minimum target sizes, which is also the maximum we could ask for:

- Each annotation instance should have at least 500 tweets
- Each annotation instance should have at least 200 relevant tweets
- Each annotation topic within an instance should have at least 50 relevant tweets

When we take all those factors into account, we were able to create four or five of such datasets.

F.2 Data Annotation Application

Twitcident Data Annotation Application

Annotatie voor VPS-13: TNO2



1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Link naar originele tweet

@3fm criminaliteit in Utrecht lager? No way, dit wknd onze klassieke Porsche gestolen uit garage!

Is deze tweet relevant voor de VPS-13: TNO2 monitor?

✖ ? ✔

Figure F.1: Data Annotation Application: first annotation phase

We implemented the actual Data Annotation Application in which users annotate in a combination of web-languages: PHP, HTML, JavaScript, CSS, MySQL, jQuery and Twitter Bootstrap. After users login, they are presented with their batch overview, and can choose a batch to start annotating. Each batch consists of 15 tweets that are sequentially annotated in a two-phase process, as the Data Annotation application was designed for fast and efficient annotation.

Figure F.1 illustrates the first annotation phase: a user is presented with a tweet and given three options: the tweet is irrelevant (red button), the tweet might be relevant (orange button) or the tweet is relevant (green button). When a tweet is considered irrelevant, a user is immediately presented the next tweet. Any of the other two options initiate the second annotation phase for a tweet.

@STM criminaliteit in Utrecht lager? No way, dit wknd onze klassieke Porsche gestolen uit garage!

Is deze tweet relevant voor de VPS-13: TNO2 monitor?

Voor welke categorie(ën) is deze tweet (mogelijk) relevant?

Leefbaarheid:

Overlast:

Veiligheidsbeleving:

Slachtofferschap:

Politie in de buurt:

Niet genoemd:

Volgende >>

Figure F.2: Data Annotation Application: second annotation phase

Figure F.2 illustrates the second annotation phase, which is initiated when a tweet is annotated as (potentially) relevant. In this phase, we go a level deeper and annotate on a Twitcident Topic level. The user now identifies to which topics this tweet is relevant, and to what degree. By default, all topics are set to irrelevant (marked by an X). For each topic, the user can mark it relevant to the tweet (marked by a V) or extremely relevant to a tweet (marked by a star). The last category is used for tweets the Twitcident Operator can attach a direct action or response to, and are considered the actual tweets a Twitcident Client is looking for.

When the user presses the blue button (Next), the next tweet is displayed. Using the navigation bar at the top, a user can jump to a any other (previous) tweet. As soon as a tweet is annotated, it is marked green on this bar. As soon as all tweets are annotated, the user is prompted with a dialog whether to save the annotations and is returned to the batch overview screen. The annotations are then stored in a MySQL database (schema specification can be found in Appendix C.1).

F.3 Data Collection Algorithm

To construct a representative dataset, we have devised an algorithm. The Twitcident databases contain a few interesting fields we could exploit. For each tweet we exploit the following fields from the available databases:

id Twitter Tweet ID

text Tweet text

fromUser Username

fromUserId Twitter Username ID

createdAt Date of tweet

stream The incoming stream that collected the tweet

theme The classification the current state of Twitcident attached based on theme keywords. Is zero for tweets that were not classified.

isMarked Boolean indicating whether the tweet is marked in Twitcident. This can roughly be translated to being an interesting/relevant tweet. However, we have to remark that very few Twitcident Operators ever mark a tweet, so this field is just a ‘bonus’.

isNoise Boolean indicating whether the tweet is marked as noise in Twitcident, and thus irrelevant. Same as with *isMarked*, we have to remark that very few Twitcident Operators ever mark a tweet as noise, so also this field is just a ‘bonus’.

Using these fields, we devised Algorithm 5 to generate our annotation batches: \mathcal{T} is the set of tweets in the target instance database, \mathcal{W} is the time window we want to collect our tweets from, η is the batch size we want to generate, μ is the maximum amount of marked tweets, ν is the maximum amount of tweets marked as noise and ρ is the desired ratio of pre-classified tweets (theme is not zero).

In lines 2–4 we generate four sets of tweets. Note that these are, under the assumption a tweet is not both marked as relevant and noise, mutually exclusive sets, such that no tweet exists in more than one set. Furthermore, it holds that $\mathcal{T} \subseteq (\mathcal{T}_{\text{marked}} \cup \mathcal{T}_{\text{noise}} \cup \mathcal{T}_{\text{themed}} \cup \mathcal{T}_{\text{unthemed}})$ if $\mathcal{W} = \infty$, and under the same assumption even equals \mathcal{T} . These sets are created by MySQL queries to the targeted Twitcident Instance database.

In lines 7–9 we initially determine how many tweets we want to fetch from each set. However, as this does not always sum up to the desired batch size η (for example, when every tweet is marked or noisy), lines 11–18 attempt to fill up by loosing the restriction parameters. The execution of these lines will be extremely rare, but just in case.

In lines 20–32 the sizes of the remaining sets are calculated according to the themed ratio parameter ρ . Note that these calculations are done under the assumption that all marked and noisy tweets are also themed, which is always the case in the Twitcident Instance databases.

Algorithm 5 Generating a data annotation batch

```

1: procedure BATCHGENERATION(  $\mathcal{T}, \mathcal{W}, \eta, \mu, \nu, \rho$  )
2:    $\mathcal{T}_{\text{marked}} \leftarrow \{t \in \mathcal{T} \mid t_{\text{createdAt}} \in \mathcal{W} \wedge t_{\text{isMarked}}\}$ 
3:    $\mathcal{T}_{\text{noise}} \leftarrow \{t \in \mathcal{T} \mid t_{\text{createdAt}} \in \mathcal{W} \wedge t_{\text{isNoise}}\}$ 
4:    $\mathcal{T}_{\text{themed}} \leftarrow \{t \in \mathcal{T} \mid t_{\text{createdAt}} \in \mathcal{W} \wedge t_{\text{theme}} \neq 0 \wedge \neg t_{\text{isMarked}} \wedge \neg t_{\text{isNoise}}\}$ 
5:    $\mathcal{T}_{\text{unthemed}} \leftarrow \{t \in \mathcal{T} \mid t_{\text{createdAt}} \in \mathcal{W} \wedge t_{\text{theme}} = 0 \wedge \neg t_{\text{isMarked}} \wedge \neg t_{\text{isNoise}}\}$ 
6:    $N_{\text{marked}} \leftarrow \min(|\mathcal{T}_{\text{marked}}|, \mu)$ 
7:    $N_{\text{noise}} \leftarrow \min(|\mathcal{T}_{\text{noise}}|, \nu)$ 
8:    $N_{\text{remainder}} \leftarrow \min(|\mathcal{T}_{\text{themed}}| + |\mathcal{T}_{\text{unthemed}}|, \eta)$ 
9:    $N_{\text{fill}} \leftarrow \eta - (N_{\text{marked}} + N_{\text{noise}} + N_{\text{remainder}})$ 
10:  if ( $N_{\text{fill}} > 0$ )  $\wedge$  ( $|\mathcal{T}_{\text{noise}}| > \nu$ ) then
11:     $N_{\text{noise}} \leftarrow \min(|\mathcal{T}_{\text{noise}}|, \nu + N_{\text{fill}})$ 
12:     $N_{\text{fill}} \leftarrow \eta - (N_{\text{marked}} + N_{\text{noise}} + N_{\text{remainder}})$ 
13:    if ( $N_{\text{fill}} > 0$ )  $\wedge$  ( $|\mathcal{T}_{\text{marked}}| > \mu$ ) then
14:       $N_{\text{marked}} \leftarrow \min(|\mathcal{T}_{\text{marked}}|, \mu + N_{\text{fill}})$ 
15:    end if
16:  end if
17:   $N_{\text{themed}} \leftarrow \max(0, \lceil (\rho \cdot \eta) - (N_{\text{marked}} + N_{\text{noise}}) \rceil)$ 
18:   $N_{\text{themed}} \leftarrow \min(|\mathcal{T}_{\text{themed}}|, N_{\text{themed}})$ 
19:   $N_{\text{unthemed}} \leftarrow N_{\text{remainder}} - N_{\text{themed}}$ 
20:   $N_{\text{unthemed}} \leftarrow \min(|\mathcal{T}_{\text{unthemed}}|, N_{\text{unthemed}})$ 
21:  if  $N_{\text{themed}} + N_{\text{unthemed}} < N_{\text{remainder}}$  then
22:    if ( $N_{\text{unthemed}} = |\mathcal{T}_{\text{unthemed}}|$ )  $\vee$  ( $N_{\text{themed}} < |\mathcal{T}_{\text{themed}}|$ ) then
23:       $N_{\text{fill}} \leftarrow \min(N_{\text{remainder}} - N_{\text{unthemed}} - N_{\text{themed}}, |\mathcal{T}_{\text{themed}}| - N_{\text{themed}})$ 
24:       $N_{\text{themed}} \leftarrow N_{\text{themed}} + N_{\text{fill}}$ 
25:    else if ( $N_{\text{themed}} = |\mathcal{T}_{\text{themed}}|$ )  $\vee$  ( $N_{\text{unthemed}} < |\mathcal{T}_{\text{unthemed}}|$ ) then
26:       $N_{\text{fill}} \leftarrow \min(N_{\text{remainder}} - N_{\text{unthemed}} - N_{\text{themed}}, |\mathcal{T}_{\text{unthemed}}| - N_{\text{unthemed}})$ 
27:       $N_{\text{unthemed}} \leftarrow N_{\text{unthemed}} + N_{\text{fill}}$ 
28:    end if
29:  end if
30:   $\mathcal{B}_{\text{marked}} \leftarrow$  Randomly sample  $N_{\text{marked}}$  tweets from  $\mathcal{T}_{\text{marked}}$ 
31:   $\mathcal{B}_{\text{noise}} \leftarrow$  Randomly sample  $N_{\text{noise}}$  tweets from  $\mathcal{T}_{\text{noise}}$ 
32:   $\mathcal{B}_{\text{themed}} \leftarrow$  Randomly sample  $N_{\text{themed}}$  tweets from  $\mathcal{T}_{\text{themed}}$ 
33:   $\mathcal{B}_{\text{unthemed}} \leftarrow$  Randomly sample  $N_{\text{unthemed}}$  tweets from  $\mathcal{T}_{\text{unthemed}}$  return
     $\text{shuffle}(\mathcal{B}_{\text{marked}} \cup \mathcal{B}_{\text{noise}} \cup \mathcal{B}_{\text{themed}} \cup \mathcal{B}_{\text{unthemed}})$ 
34: end procedure

```

In lines 34–38 the batch gets sampled from the sets and returned after being randomly shuffled.

We performed this batch generation process every week for over half a year. For every instance one batch with $\eta = 15$ tweets was sampled with the window \mathcal{W} set to three weeks ago. This was to ensure clients had the opportunity to mark tweets as relevant or noise, which would increase our dataset quality, but also to fetch pretty recent tweets so the annotating user was relatively context-aware at the moment.

After initial experiments, we used the following values for the other parameters to get representative balanced datasets: $\mu = 8, \nu = 4, \rho = 0.85$. The latter themed ratio

value of 0.85 seems high at first, but this is largely due to the fact that the ‘themed’ tweets also have a large proportion of noise. Generally speaking, if there were no marked tweets (often the case), the themed set would contain a single potentially relevant tweet on average out of 15 tweets, and the unthemed set none. However, we still wanted a 15% fraction of tweets that were not classified by the current Twitcident application as to remove its bias a bit.

F.4 Output Processing Algorithm

The aim of our classifier is to output a value in the normalized range $[0.000, 1.000]$. This means that we must process our annotation results to this range. Furthermore, a single tweet is always annotated by multiple users, and we need to aggregate these annotation together to a single classification score.

We must also note that not every annotating user is as knowledgeable and close to the client, and therefore, for each instance I , we attached a confidence weight $C_{I,U} \in \mathbb{Z}^+$ to each user U .

Recall that for every annotated tweet we have a relevancy of the tweet which we will denote by $\mathcal{R}_{tweet,U} \in \{0, 1, 2\}$ for every user U , where 0 means irrelevant and 2 relevant. We also have a relevancy for each topic T which we will denote by $\mathcal{R}_{topic,U}^T \in \{0, 1, 2\}$ for every user U . Then for each tweet, we calculate its score $\mathcal{S}_{U,T,\mathcal{R},C}$ by:

$$\mathcal{S}_{U,T,\mathcal{R},C} = \alpha \cdot \frac{\sum_{\forall U} (\mathcal{R}_{tweet,U} \cdot C_{I,U})}{\sum_{\forall U} (2 \cdot C_{I,U})} + (1 - \alpha) \cdot \frac{\sum_{\forall U} (\mathcal{R}_{topic,U}^T \cdot C_{I,U})}{\sum_{\forall U} (2 \cdot C_{I,U})}$$

The first part of this formula calculates the partial tweet score, whereas the second part calculates the partial topic score. These are both normalized weighted averages of the annotations. These are then merged together using a ratio parameter $\alpha \in \langle 0.00, 1.00 \rangle$, which indicates to which extent the tweet relevancy must be taken into account when calculating for this topic score. An example value would be $\alpha = 0.2$, indicating that the tweet relevancy would account for 20% and the topic relevancy for 80%. In our opinion, the tweet relevancy should be taken into account because Twitcident topics can be pretty similar and annotating users can easily miss a topic when the tweet is relevant to multiple topics.

Appendix G

Analysis of Prinsjesdag 2013

Analyse data Prinsjesdag

De volgende analyse is gedaan met behulp van Kibana op de data van Prinsjesdag. Deze data is afkomstig uit een vooraf geconfigureerde Twitcident monitor, en met enige pre-processing geëxporteerd: de themas zijn geherclassificeerd om alle matchende thema's aan de tweet gekoppeld te hebben in plaats van alleen het eerste matchende thema.

Relatie tussen streams en themas:

Op streams worden tweets gezocht en binnengehaald. Daarna wordt de tweet geclassificeerd met een of meerdere themas's.

Algemene dataset statistieken

De volgende statistieken zijn, zonder enige configuratie, direct afleesbaar:

Monitoring-periode

Start:	13-09-2013 00:00
Eind:	28-10-2013 21:00

Aantallen tweets

Totaal:	617700 tweets
Classificaties met thema:	74214 tweets
Markeringen:	21 tweets

Configuratie

Streams:	20 streams
Themas:	6 themas

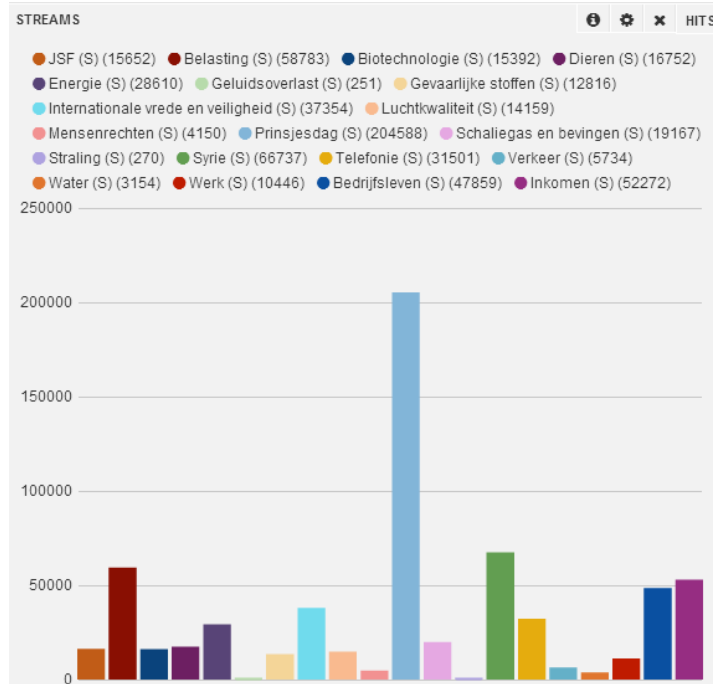
Inhoud

Analyse data Prinsjesdag	1
Algemene dataset statistieken	1
Streams en themas	2
Ongeclassificeerde data (<i>in het kader van Event Detection</i>)	3
Case: Zwerfkatten	4
Case: Zwarte Piet.....	6
Case: Windmolens, Windenergie	9
Conclusie ongeclassificeerde data	12
Interessante markeringen (9/21)	13
Zoom-in op specifieke periode's	14
Timeline overzicht	14
De aanloop naar prinsjesdag.....	15
Zoom-in op relevante dossiers	18
Klimaat	18
Innovatie	19
Kernenergie.....	20
Bouwsteen 'Milieu & duurzaamheid'	22

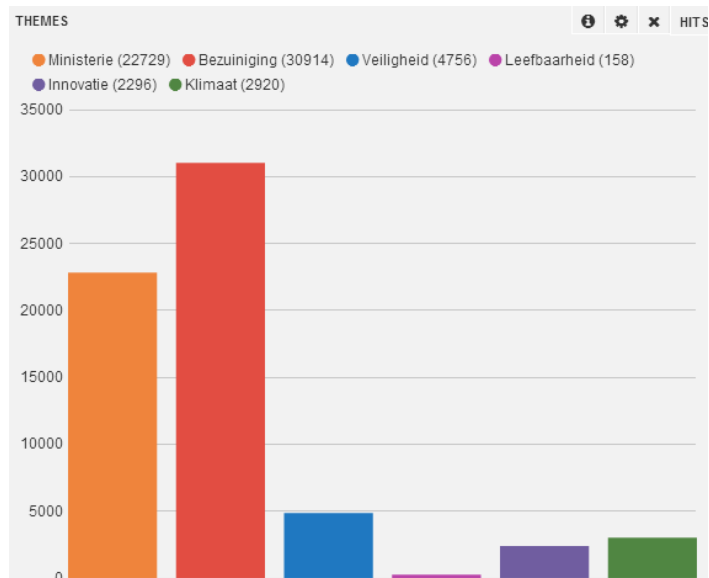
Streams en thema's

De aantallen per stream en per thema zijn ook direct afleesbaar, en geven een beeld van de configuratie en relatieve verdeling:

Streams



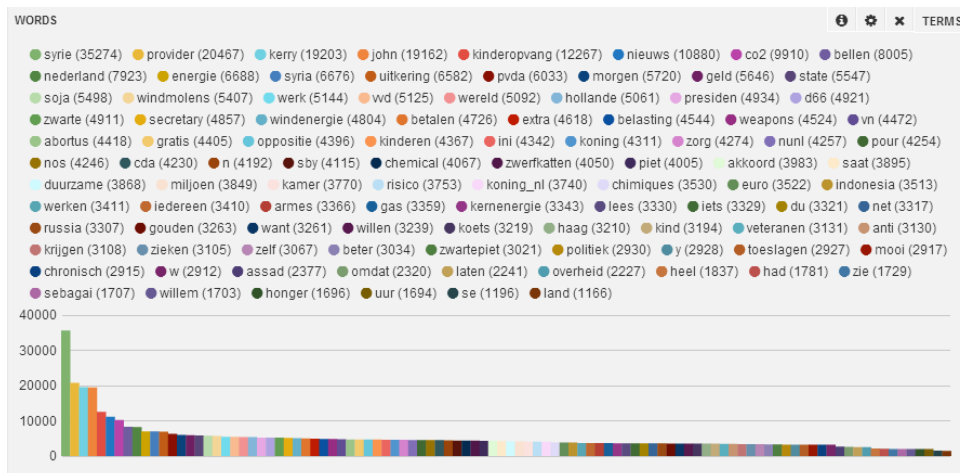
Themas



Ongeclassificeerde data (in het kader van Event Detection)

Om mogelijk nieuwe interessante thema's te vinden waarover veel getweet wordt, kunnen we de term frequenties van de ongeclassificeerde tweets bekijken. Dit kan interessant zijn in het kader van event-detection.

Door een filter te maken waardoor alleen tweets zonder gekoppeld thema getoond worden, en vervolgens de term-frequenties inzichtelijk te maken waarbij we alle stream-zoektermen en algemene termen wegfilteren, krijgen we het volgende beeld:

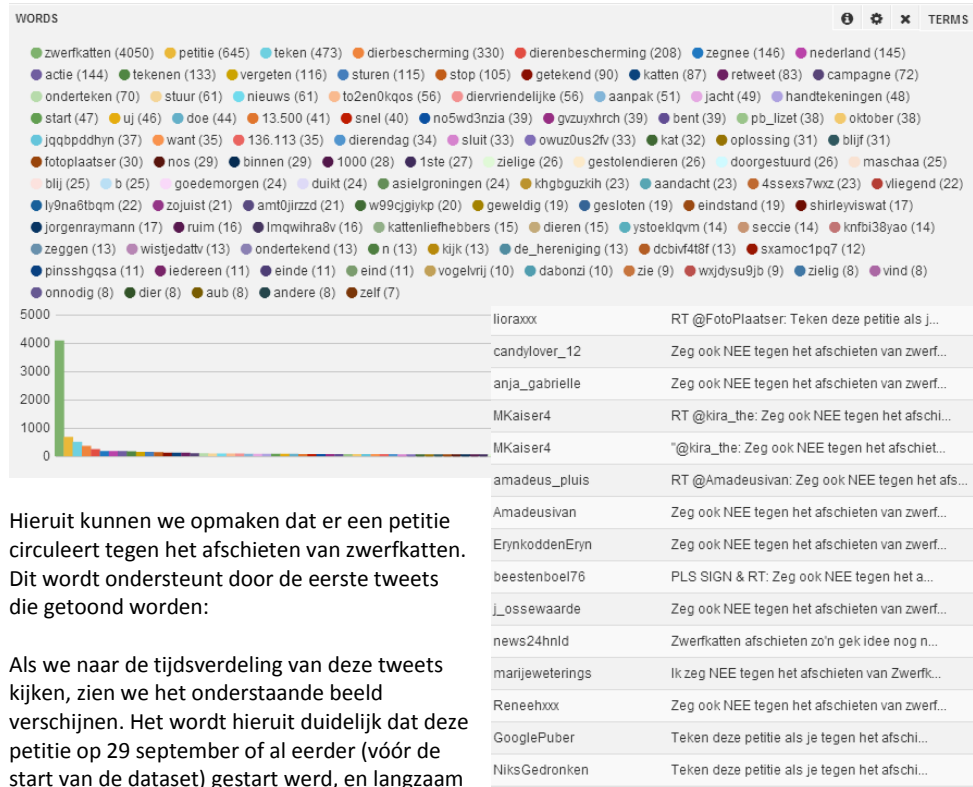


Opvallendheden:

- Syrië wordt vaak genoemd. Blijkbaar staat deze niet op zichzelf staand in de streaming keywords, anders zou deze uit de lijst gefilterd worden.
- Provider
- Kinderopvang
- Energie en CO2
- Windmolens en windenergie
- Geld
- Abortus
- Zwerfkatten
- Zwarte piet
- John Kerry / Russia (buitenland)
- Honger

We kunnen elk van deze opvallendheden dieper analyseren, door de ongeclassificeerde data te filteren op deze veel voorkomende woorden. Op die manier krijgen we een beeld van wát er precies over gezegd wordt, maar ook wanneer. We illustreren dit aan de hand van 3 cases: Zwerfkatten, Zwarte Piet en Windmolens/windenergie/energie/CO2.

Case: Zwerfkatten



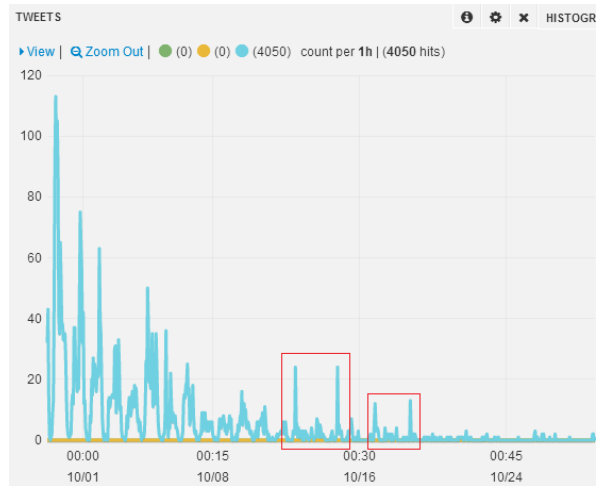
Hieruit kunnen we opmaken dat er een petitie circuleert tegen het afschieten van zwerfkatten. Dit wordt ondersteunt door de eerste tweets die getoond worden:

Als we naar de tijdsverdeling van deze tweets kijken, zien we het onderstaande beeld verschijnen. Het wordt hieruit duidelijk dat deze petitie op 29 september of al eerder (vóór de start van de dataset) gestart werd, en langzaam aandacht verliest.

Opvallend zijn de 2 gemarkeerde piekjes die later terugkomen. Na hierop in te zoomen (tijdsfilter) zien we dat de eerste set piekjes nieuwe aanwakkeringen met nieuwe tweekteksten zijn.

De 2^e set piekjes blijken een retweet te zijn over het feit dat de petitie is gesloten:

“RT @Dierbescherming: Zojuist is de petitie #zegnee tegen het afschieten van zwerfkatten gesloten op een eindstand van 136.113. Geweldig, bedankt!”





Dierenbescherming

@Dierbescherming



Follow

Zeg ook NEE tegen het afschieten van
zwerfkatten in Nederland. Onderteken de
petitie op ow.ly/pjzjo. #zegnee

View translation

Reply Retweet Favorite More

<https://twitter.com/Dierbescherming/status/384592372417699840>



Dierenbescherming

@Dierbescherming



Follow

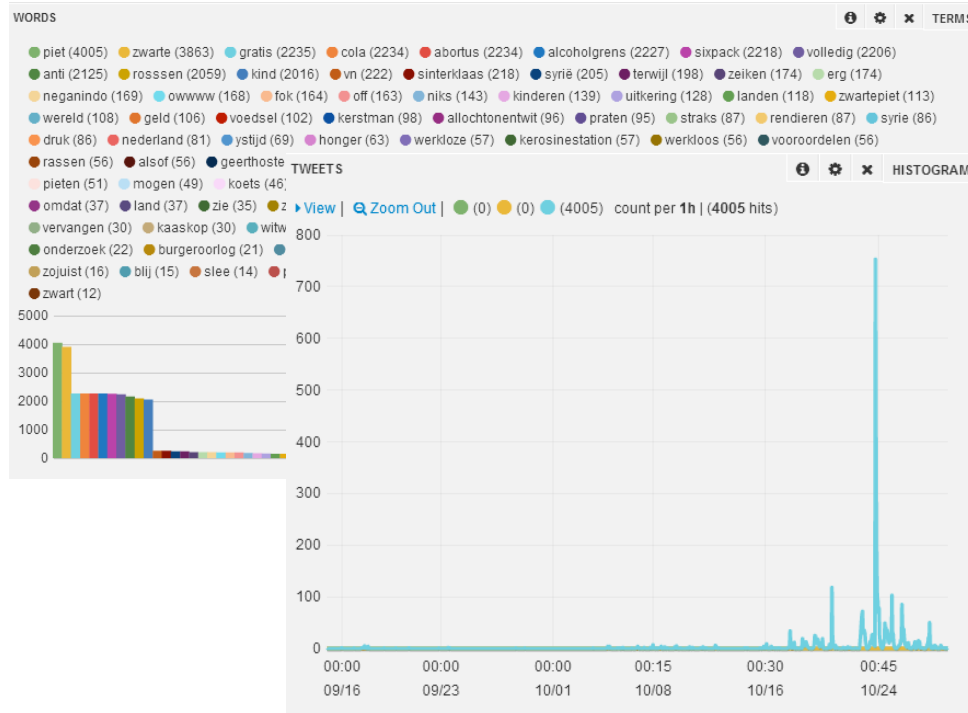
Zojuist is de petitie #zegnee tegen het
afschieten van zwerfkatten gesloten op een
eindstand van 136.113. Geweldig, bedankt
allemaal!

View translation

Reply Retweet Favorite More

<https://twitter.com/Dierbescherming/status/391218082028322817>

Case: Zwarte Piet



Het eerste dat hierbij sterk opvalt is het grote verband tussen Zwarte Piet, Cola, Abortus en alcoholgrens. Na inzoomen op de data blijkt dit door de zeer grote populariteit van de volgende tweet te komen: (2011 retweets)

RT @Rosssen: Zwarte Piet weg, alcoholgrens naar 18. Nu alleen nog een gratis abortus bij een sixpack cola en dan zijn we volledig anti-kind

We kunnen na filteren op deze tweet ook eenvoudig zien dat deze tweet voor het eerst gelanceerd is op 24 oktober en daarna razend populair werd. Gezien de eerdere piekjes is het ook interessant te kijken wat zich juist buiten deze tweet om speelde: daarom draaien we het filter om, zodat ‘de rest’ rondom zwarte piet overblijft. Verder filteren we alle tweets met cola en abortus er ook uit:



Heersende en populaire tweets hierbij zijn de volgende:

RT @NegaNindo: en terwijl wij zeiken of zwarte piet is er oorlog in Syrië maar owwww nee zwarte piet is echt erg FOK OFF MAN

RT @Koning_NL: Liefst 4 VN-rapporteurs buigen zich over #zwartepiet. Oorlog en honger in de wereld zijn blijkbaar opgelost. Hoera!

Dit impliceert dat naast alle ophef de publieke opinie ook met name relativerend tegenover het Zwarte Piet verhaal staat.

Als we de originele tweeters van deze 3 berichten bekijken, kunnen we het volgende opmerken:

@Rossen

Als we het Twitter gebruikersprofiel bekijken, zien we dat deze gebruiker veel 'originele' en 'rakende' tweets tweet met soms een vleugje humor, maar vaak met een kern van inhoud. Verder geeft hij veel zijn mening in reacties op andere tweets. Zijn tweets worden regelmatig geretweet. Als we kijken naar de gegevens in onze dataset, zien we de volgende tweets:

Zwarte Piet weg, alcoholgrens naar 18. Nu alleen nog een gratis abortus bij een sixpack cola en dan zijn we volledig anti-kinderen. Mooi.
Flikker lekker op met je filmpjes van dierenleed, dieren mishandeling en andere onzin. Iedereen is zich er al bewust, en ondertussen moe van.
Hoi ik ben rossen, werk fulltime met een bovengemiddeld inkomen en hoef dat niet met foto's van geld te bewijzen zoals jij @KushingBubble
Leer het verschil tussen een boven gemiddeld inkomen, en uursalaris. In een callcenter kan je zelfs 20 euro per uur maken. @kisseskyra
"De VS zit achter 9/11 want ze wilden Irak binnenvallen voor de olie en winst maken." Denk eens na. Die oorlog KOSTTE de VS 3000 miljard..
Morgen tijdens #prinsjesdag is de wederopstanding van Prins Friso omdat hij GTA V wilt spelen Als je productiemedewerker bent van Nickelsonjassen maak je het soms wel iets te bont
@Heteroald: Ik wacht op de derde wereldoorlog.☹
jonge doe normaal mijn opa gaat dood in de derde wereld oorlog #niet #grappig #receptloos
Je zou vandaag maar investeren in windenergie..
Amerika haalde geen winst uit die oorlog. Het KOSTTE ze 3000 miljard en veroorzaakte de crisis. Oorlog is nooit winstgevend. @Tunahanerd
RT @Insayno: Ooit opgevallen? Mag wel eens veranderen, vind je niet? #Prinsjesdag http://t.co/CvtyOvlepU
Morgen is het #prinsjesdag Niemand boeit die shit. Morgen komt ook GTA V uit.

@NegaNindo

Dit blijkt een 'gewone grootgebruiker' van Twitter, met geen hoog inhoudsgehalte. Tweet vaak per dag, en geeft regelmatig zijn mening, zo ook over Zwarte Piet, welke dit keer al snel populair werd. Als we in onze dataset kijken, blijkt dit ook de enige tweet van hem.

@Koning_NL

Een fake-account van Willem-Alexander, vooral gericht op satire. Enkele tweets die in onze dataset voorkomen:

Wij zijn even luid blazend op Onze vuvuzela met de Gouden Koets door de Schilderswijk aan het racen, Landgenoten. #turned
Verdraaid! Er is een velletje in de Gouden Koets blijven liggen! Over privacy en dat het briefgeheim ook voor mail en zo geldt. #Prinsjesdag
De A'tjes begrijpen niet dat de #PvdA tegen #JSF kan zijn. Al die zingende kinderen, daar wordt een mens toch alleen maar vrolijk van?
Fijn dat u Ons nu al complimenteert met de inhoud van Onze eerste #Troonrede, mijnheer Obama, maar #Prinsjesdag is morgen pas. #nsa



Tijl Beckand
@TijlMTBeckand



Follow

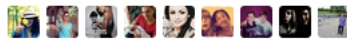
Iedereen heeft t over die Zwarte Pieten,
maar een oude man die tot ver na zn
pensioen met een paard t dak op moet vindt
iedereen normaal..

[View translation](#)

[Reply](#) [Retweet](#) [Favorite](#) [More](#)

672
RETWEETS

43
FAVORITES



<https://twitter.com/TijlMTBeckand/status/390052663150133248>



Rossen
@Rosssen



Follow

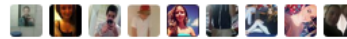
Zwarte Piet weg, alcoholgrens naar 18. Nu
alleen nog een gratis abortus bij een sixpack
cola en dan zijn we volledig anti-kinderen.
Mooi.

[View translation](#)

[Reply](#) [Retweet](#) [Favorite](#) [More](#)

2,598
RETWEETS

292
FAVORITES



8:16 AM - 23 Oct 13

<https://twitter.com/Rosssen/status/393033106631835648>



Willem-Alexander
@Koning_NL



Follow

Liefst 4 VN-rapporteurs buigen zich over
[#zwartepiet](#). Oorlog en honger in de wereld
zijn blijkbaar opgelost. Hoera!
bit.ly/1d5dmmQ

[View translation](#)

[Reply](#) [Retweet](#) [Favorite](#) [More](#)

2,760
RETWEETS

105
FAVORITES



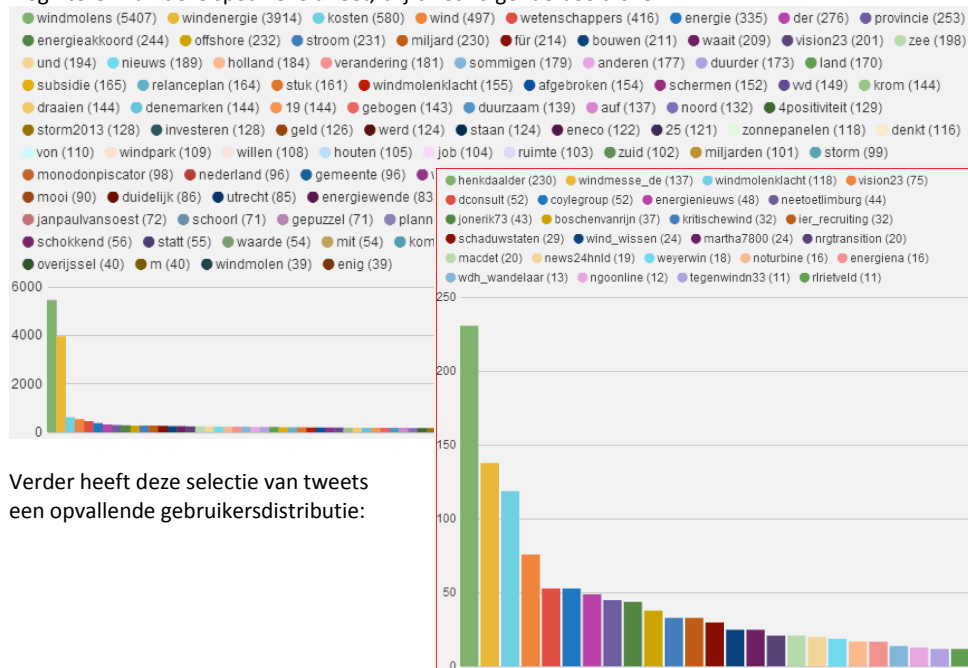
2:15 AM - 19 Oct 13

https://twitter.com/Koning_NL/status/391492832483352576

Case: Windmolens, Windenergie

Deze woorden blijken duidelijk niet in het actieve zoekbestand te staan (bijv bij Energie), anders zouden zij weggefilterd worden. Dit geeft in eerste instantie aanleiding deze woorden als zoekwoorden te overwegen.

Na een eerste blik op de termfrequenties blijken er veel Duitse woorden in voor te komen. Dit was te verklaren door een populaire recruiting tweet rondom windenergie in Duitsland. Na het wegfilteren van deze specifieke tweet, blijft het volgende beeld over:

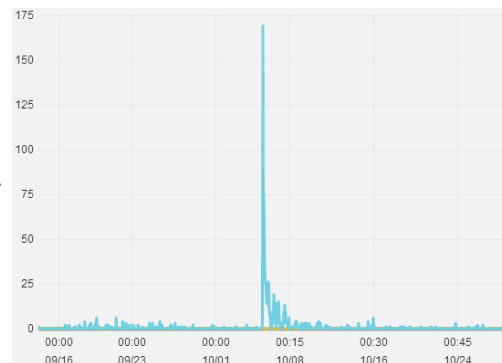


Op basis van de veel voorkomende gerelateerde woorden zien we als interessant:

- Kosten
- Wetenschappers
- Energieakkoord
- Miljard

We voeren nu 2 wijzigingen in de filters door:

1. We filteren de tweets op bovenstaande 4 woorden, om het beeld te focussen op de interessante woorden, en te achterhalen wat hierover gezegd wordt.
2. We voegen de geclassificeerde data weer toe, omdat dit wel binnen de vooraf vastgestelde thematiek valt, om zodoende het beeld volledig te maken.



Wat dan direct opvalt is de distributie over tijd:

We passen dan ook direct een tijdsfilter op deze periode toe, om de omliggende 'ruis' weg te nemen.

Als we vervolgens naar de tweets kijken die aan deze criteria voldoen, zien we de volgende tweets overheersen:

<i>Wetenschappers: windmolens kosten miljarden meer: De plannen van de overheid om flink meer windenergie te winnen, k...</i> http://t.co/RNSe8kf0Ug
<i>'Windmolens kosten veel meer dan in Energieakkoord staat':</i> http://t.co/NFDjWNUQNd
<i>Dus die lelijke dingen staan alleen maar geld te kosten? Groene Rekenkamer windenergie </i> http://t.co/V0EbMM1Ke2 : http://t.co/YqMnlfdrhR
<i>RT @kritischewind: Windenergie: een 'rekenfoutje' van 15 miljard </i> http://t.co/uDzHmTnmZR : http://t.co/Auzv3mqmHg

En:

RT @janpaulvansoest: Na enig gepuzzel werd duidelijk wie de 'wetenschappers' zijn die betogen dat windenergie te duur is: <http://t.co/IJdUa>

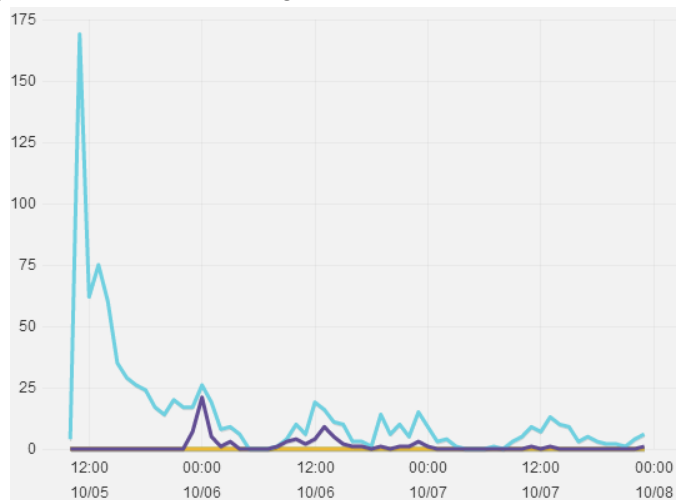
Opnieuw blijkt het retweet-gedrag leidend, en werd dit getriggerd door een nieuwsbericht (enkele media wakkerden het waargenomen fenomeen aan).

We zien dat laatstgenoemde tweet tegen de eerdere 4 van de media ingaat. Om te achterhalen hoe de verhoudingen hiertussen zijn, zetten we deze tweet af tegen de rest, en zien dan het volgende:

De lichtblauwe lijn geeft het totaal aantal tweets weer, de paarse lijn de tweet die tegen de nieuwsberichten ingaat.

We zien dat dit ene tegenbericht gelijk de heersende opinie overneemt, het de retweets van de nieuwsmedia stillegt.

Later zien we echter dat er opnieuw enkele schommelingen zijn, en zoomen daarom ook op dat tijdinterval in, en halen vervolgens de tweet van @janpaulvansoest uit de selectie.



De tweets die overblijven gaan opnieuw over 'het rekenfoutje', en gezien de bovenstaande trend is dit uiteindelijk 'de winnaar' die de laatste activiteit vertoont.


 **Bart Zuidervaart**
@BartTrouw  

Wetenschappers: windmolens kosten veel meer dan in #energieakkoord staat. Niet 3,7 maar 19 miljard euro. s.trouw.nl/3521997?utm_so...

 View translation

 Reply  Retweet  Favorite  More

<https://twitter.com/BartTrouw/status/386387916110299136>

 **Jan Paul van Soest**
@janpaulvansoest  


Na enig gepuzzel werd duidelijk wie de 'wetenschappers' zijn die betogen dat windenergie te duur is:
pic.twitter.com/IJdUafWkTa

 View translation

 Reply  Retweet  Favorite  More



<https://twitter.com/janpaulvansoest/status/386571556937138176>

 **Bas Eickhout**
@BasEickhout  

Een van die kritische 'wetenschappers' tegen windmolens is good-old klimaatontkenner Hans Labohm. #voorspelbaar #kritischemedia?

 View translation

 Reply  Retweet  Favorite  More

<https://twitter.com/BasEickhout/status/386422758462468096>

Conclusie ongeclassificeerde data

Onvoorziene 'events' zijn te identificeren door naar term-frequenties te kijken van ongeclassificeerde data, en daarbij de termen waarop gezocht wordt weg te laten. Voorwaarde is wel dat de data binnenkomt via gerelateerde zoekwoorden. Dit kan zowel achteraf, als real-time, zolang er gebruik wordt gemaakt van een IR-engine.

Tevens hebben we aangetoond dat we, met deze manier van benaderen, naast 'trending topics' ook snel meer informatie inzichtelijk kunnen maken, zoals welke specifieke tweets populair zijn, hoe populair, en wanneer.

Het retweet-gedrag lijkt daarbij leidend om snel en efficiënt de publieke opinie te peilen.

Interessante markeringen (9/21)

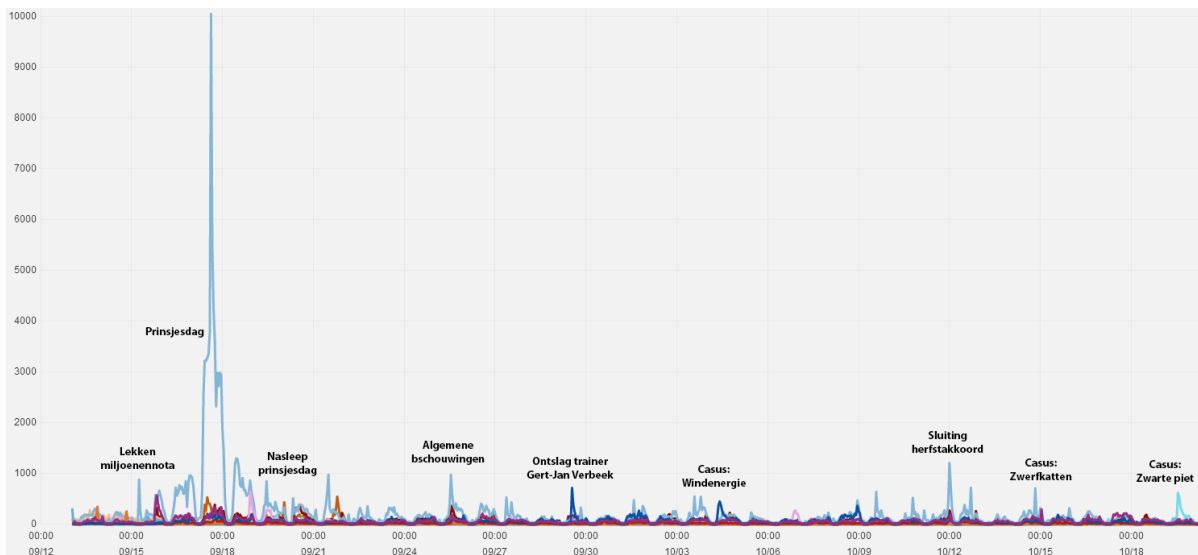
<i>Goed, we hebben de JSF. Laten we er dan in godsnaam ook maar blij mee zijn. Wanneer is de eerstvolgende oorlog?</i>
<i>Leg mij logica uit: we bestellen 37 JSF's voor liefst 4 miljard euro. 4000 kamervragen erover kostte 15 miljoen. En 2300 man defensie er uit</i>
<i>Ik heb zojuist met pijn in mijn hart lidmaatschap van de PvdA opgezegd. JSF krijg ik mezelf niet uitgelegd!</i>
<i>@APechtold met mijn inkomen van 1200 euro leef ik heel krap nu moeten de ouderen nog meer bezuinigen dat kan niet meer</i>
<i>Defensie wil bezuinigen door o.a. de kazerne in #Assen te sluiten. En dan toch besluiten om de #JSF aan te schaffen??</i>
<i>Kabinet weet wat er leeft in Nederland. Schaffen de #jsf aan en burgers leveren weer koopkracht in. #zalwelaanmijliggen</i>
<i>Weten we eindelijk waar al die extra bezuinigingen toe leiden: aanschaf 37 JSF straaljagers voor 4,5 miljard euro #miljoenennota #defensie</i>
<i>In tijden van Crisis waar je amper nog je Brood kan betalen Schaft ONS kabinet 37 JSF vliegtuigen aan! #VVD #PVDa <<<< #wegermee !!!!!</i>
<i>Iets wat ik niet begrijp, en waarschijnlijk nooit zal begrijpen: 6 miljard bezuinigen, maar wél 37 JSF's kopen? #miljoenennota #prinsjesdag</i>

Ten opzichte van 'trending topics' kan innovatief real-time monitorren (Twitcident) meer context aan een topic geven: wat wordt er precies over gezegd, hoeveel, en door wie?

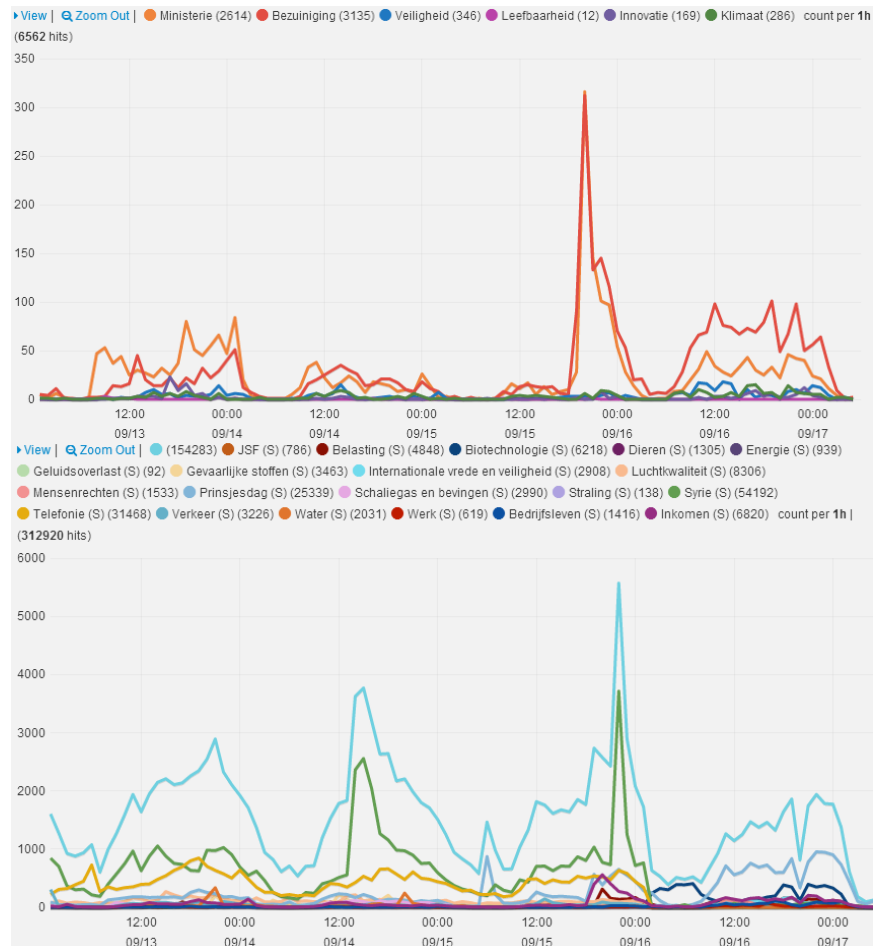
Er heerst veel onbegrip over de samenhang tussen de JSF en bezuinigen. Doel van JSF niet duidelijk?

Zoom-in op specifieke periode's

Timeline overzicht



De aanloop naar prinsjesdag

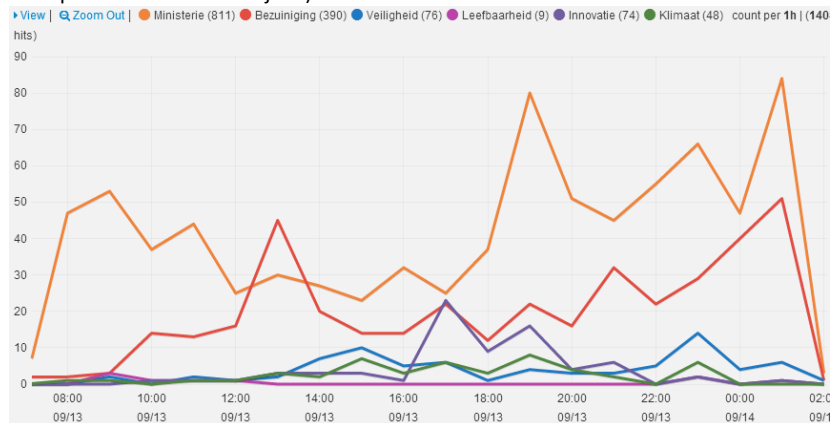


Boven zien we de geclassificeerde data. Onder zien we alle streams, waarbij de bovenste blauwe lijn het totaal aantal tweets omvat. Het eerste dat opvalt is het enorme aandeel dat 'Syrië' heeft. Op basis van dit patroon zullen we achtereenvolgens inzoomen op de volgende momenten, om te zien wat er toen gebeurde (dit kon toen ook real-time):

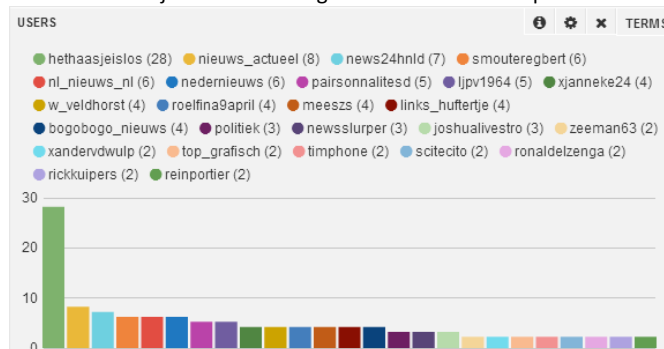
- 13 september 08:00-23:59, op basis van de piekjes in geclassificeerde data
- 14 september 12:00-15:00, op basis van sterke piek in de Syrië-stream
- 15 september 16:30-21:00, op basis van de grote spike
- 16 september 12:00-18:00, op basis van het gerommel in de geclassificeerde data

13 september 08:00-23:59

We zoomen in op deze periode, en filteren op geclassificeerde data (dat gedeelte gaf aanleiding deze periode nader te bekijken).



Als we verder kijken valt ook de gebruikersdistributie op:



Als we deze gebruiker nader bekijken, zien we dat deze actief een demonstratie in Amsterdam aan het aanwakkeren is voor zaterdag 21 september (1 week later):

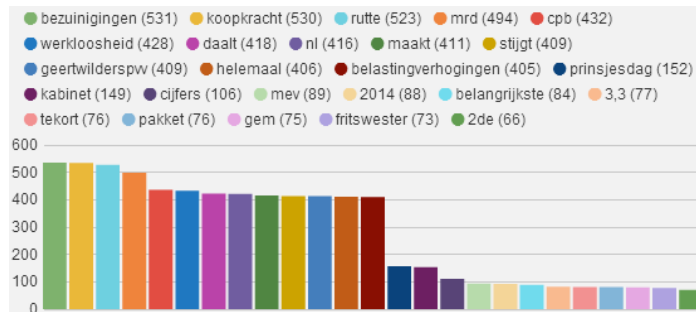
Door het actief real-time monitorren, kunnen we al vroeg inspelen op bepaalde gebeurtenissen (demonstraties).

Na diepere analyse op deze periode zien we dat er veel verschillende uitspraken gedaan worden in reactie op uitspraken van kabinetsleden. Met name Rutte en Pechtold worden aangehaald. Verder wordt er ook veel getweet dat 'het vertrouwen in het kabinet historisch laag is', naar aanleiding van een startend bericht op Geen Stijl dan regelmatig ge-retweet werd.

14 september, 12:00-15:00

Deze piek in de streaming-data werd met name veroorzaakt door veel tweets vanuit Frankrijk omtrent Syrië, chemische wapens en een uitspraak van John Kerry. In Nederland speelde hier niet veel omheen.

15 september, 16:30-21:00



Op basis van de enorme piek, en het patroon in termdistributie (groot aantal gelijken, gevolgd door een significante daling), kunnen we stellen dat deze piek veroorzaakt wordt door een erg populaire tweet, waarbij Geert Wilders en Rutte centraal staan in het kader van bezuinigingen, koopkracht en onderzoek van CPB. Na zoeken op deze termen gaat het concreet om de volgende tweet van Wilders welke erg populair is en leeft onder de bevolking:

Geert Wilders
@geertwilderspvv

CPB: koopkracht daalt, werkloosheid stijgt.
Door 6 mrd bezuinigingen en
belastingverhogingen. Rutte maakt NL
helemaal kapot. Opstappen Mark!

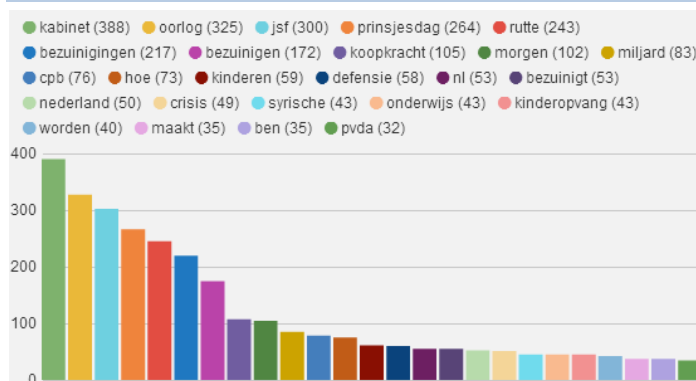
View translation

Reply Retweet Favorite More

763 RETWEETS 55 FAVORITES

9:02 AM - 15 Sep 13

16 september



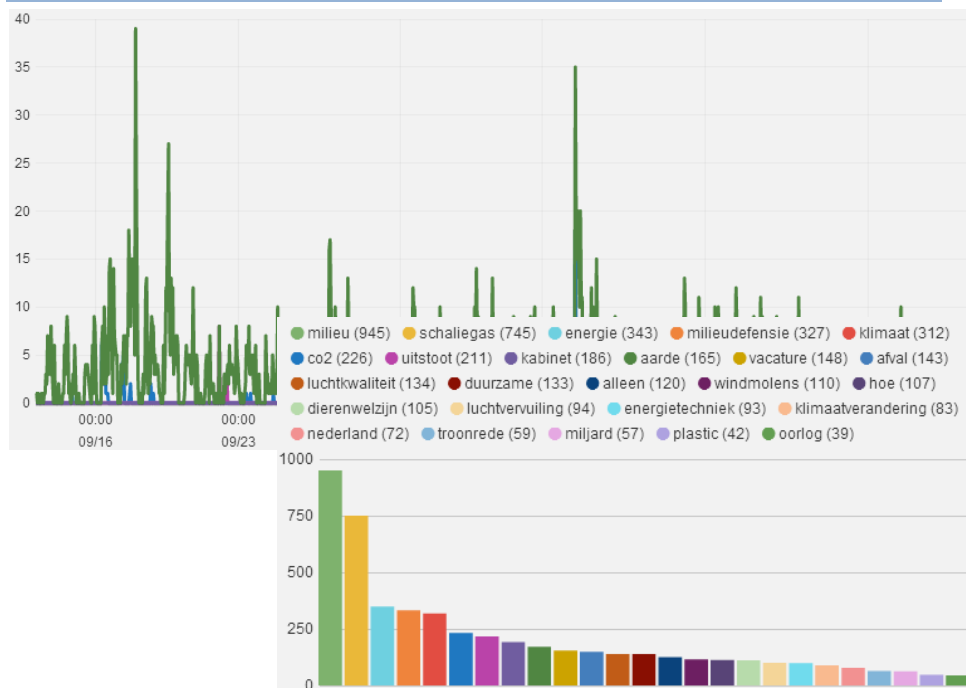
Er blijkt veel getweet te worden over het kabinet in relatie tot oorlog en de JSF, in het kader van prinsjesdag. Na filteren op deze woorden, blijkt er niet een duidelijk verband te zijn. Daarom bekijken we de woorden apart met een filter.

Het woord 'Oorlog' blijkt die dag populair, maar we waren niet in staat een duidelijke oorzaak daarvan aan te wijzen. Het wordt in verbandt gebracht met een boek met oorlog in de titel, de uitkomst van het computerspel GTA5 en wat zich afspeelt in Syrië. Veel verschillende tweets.

Verder zijn er veel vragen rondom de JSF en bezuinigingen in de aanloop naar prinsjesdag.

Zoom-in op relevante dossiers

Klimaat



Op basis van de gerelateerde woorden zien we dat 'schaliegas' centraal staat bij het thema Klimaat. Verder kunnen we zien hoeveel het publiek tweet over klimaat-gerelateerde woorden, en in welke mate.

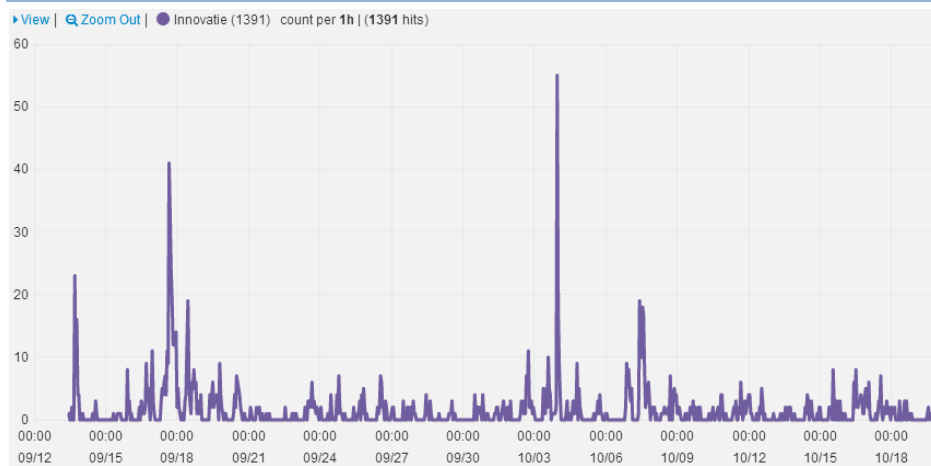
Verder zien we een aantal pieken, waar we op in willen zoomen: 2 rond prinsjesdag, en 1 rond 9 oktober 15:00.

De eerste prinsjesdag-piek wordt veroorzaakt door de volgende populaire tweet:

RT @weermanreinier: *Mogelijke dreiging milieuramp door overstroomde boorputten #schaliegas in Colorado.* <http://t.co/vy9Th7HoL9>

De tweede prinsjesdag-piek wordt veroorzaakt door de volgende populaire tweet:

Innovatie



De eerste piek boven de 20 wordt veroorzaakt door de volgende 2 tweets:

Topinstituut Amsterdam gericht op stedelijke **innovatie**: Het nieuw op te richten internationaal technologisch i... <http://t.co/yjmoVDvKDo>
Denk je aan een tweede inkomen test Free <http://t.co/c44nWVjXFo> Geen \$\$ **investering** wel kortingproducten enveiling

De tweede serie pieken valt samen met prinsjesdag, waar de datastroom groter is. De volgende tweets schetsen het beeld gedurende prinsjesdag:

BREKEND: Koning schijnt hele blz niet uitgesproken te hebben. Ging over perspectief, **innovatie**, duurzaamheid. #troonrede #Prinsjesdag
RT @deVSNU: Per saldo geen **investeringen** in wetenschappelijk onderzoek en onderwijs #Prinsjesdag <http://t.co/oSFSCqMHq6>

De piek op 3 oktober om 23:00 wordt veroorzaakt door de volgende populaire tweet:

RT @APechtold: Heb kabinet gezegd: D66 blijft streven naar samenhang hervormingen, lastenverlichting en **investeringen**. Kabinet nu aan zet.

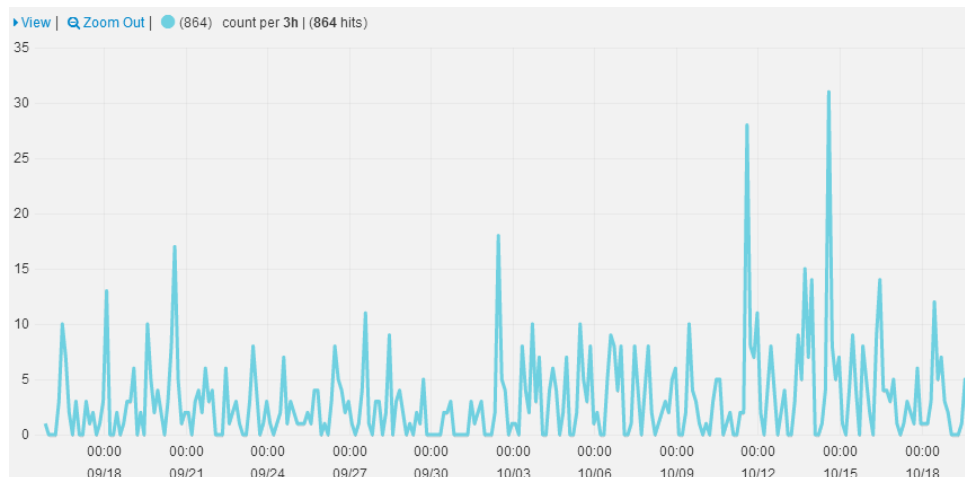
De kleinere piekjes hieromheen worden veroorzaakt door de volgende tweet:

Shell probeert **investering** schaliegas in VS terug te draaien:<http://www.ft.com/cms/s/0/c2298e46-29ae-11e3-9bc6-00144feab7de.html#ixzz2gYFTb5sD> ... (paywall) Goed, nu in EU nog het licht zien #schalieflop

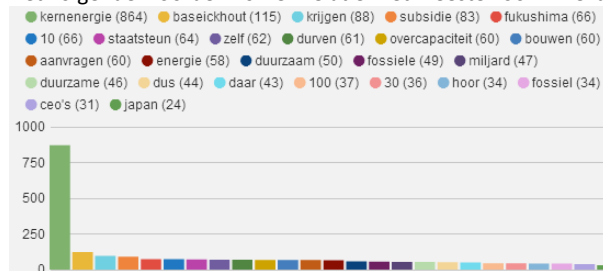
De laatste piek op 7 oktober wordt veroorzaakt door de volgende tweets:

RT @deWindvogel: Met de 24 miljard investering van Shell in schaliegas hadden alle huishoudens in NL zonnepanelen kunnen plaatsen. #keuzes
Shell topman Peter Voser heeft spijt van investeringen in schaliegas in VS #newslocker <http://t.co/5xq3qz44YN>

Kernenergie



Het volgende woorden komen relatief het meeste voor in verband met 'kernenergie':



'baseickhout' blijkt de gebruiker 'Bas Eickhout', met de volgende 2 populaire tweets:

Je moet maar durven: zelf overcapaciteit bouwen, subsidie voor kernenergie aanvragen, staatsteun krijgen en dan dit: <http://ow.ly/2ADpKk>
Deze verklaart de piek rond 11 oktober.

Dus 100 miljard naar fossiel + kernenergie. 30 miljard naar duurzaam. Daar hoor je de 10 fossiele CEO's dan niet over <http://www.sueddeutsche.de/wirtschaft/foerderung-der-energiebranche-oettinger-schoent-subventionsbericht-1.1793957>
Deze verklaart de piek rond 15 oktober.

De andere kleinere piekjes worden veroorzaakt door de volgende tweets:

Niet moeilijk doen over dier&milieu, kernenergie, asielbeleid & opkomend fascisme, dan doet de AIVD ook niet moeilijk. -
<http://www.volkskrant.nl/vk/nl/3184/opinie/article/detail/3522726/2013/10/07/De-AIVD-jaagt-op-anarcho-extremistische-spoken.dhtml>

Goedkope elektriciteit hé, die kernenergie?

10 #energiereuzen tegen steun duurzame energie, maar #kolen en #kernenergie krijgen tientallen miljarden subsidie #telegraaf

 **Bas Eickhout**
@BasEickhout 

Je moet maar durven: zelf overcapaciteit bouwen, subsidie voor kernenergie aanvragen, staatsteun krijgen en dan dit:
ow.ly/2ADpKk

 View translation

 Reply  Retweet  Favorite  More

65 RETWEETS 6 FAVORITES 

3:43 AM - 11 Oct 13
<https://twitter.com/BasEickhout/status/388615951597707265>

 **Jeroen de Haas**
@JFdeHaas 

10 #energiereuzen tegen steun duurzame energie, maar #kolen en #kernenergie krijgen tientallen miljarden subsidie #telegraaf

 View translation

 Reply  Retweet  Favorite  More

16 RETWEETS 

11:32 PM - 15 Oct 13
<https://twitter.com/JFdeHaas/status/390364663453478912>

 **Bas Eickhout**
@BasEickhout 

Dus 100 miljard naar fossiel + kernenergie. 30 miljard naar duurzaam. Daar hoor je de 10 fossiele CEO's dan niet over sueddeutsche.de/wirtschaft/foe...

 View translation

 Reply  Retweet  Favorite  More

 Süddeutsche Zeitung

Energiebranche: Oettinger schönt Subventionsbericht
EU-Kommissar Oettinger streicht Zahlen zur Förderung der Energiebranche aus Subventionsbericht.

[View on web](#) 

41 RETWEETS 1 FAVORITE 

3:44 AM - 14 Oct 13  Flag media
<https://twitter.com/BasEickhout/status/389703262783672320>

Bouwsteen 'Milieu & duurzaamheid'

Must include at least one of: (pure terms, may be extended with wildcards)

"Milieu" "Duurzaam" "Duurzame" "Energie" "Windmolen" "Windmolens" "Windenergie"
"Kernenergie" "Uitstoot" "Schaliegas" "Wind" "Hernieuwbare" "Stroom" "Gas" "Zon"
"Zonnepanelen" "Boringen" "Elektriciteit" "Zonneenergie" "Biomassa" "Bioenergie" "Fossiele"
"Proefboringen" "Winning"

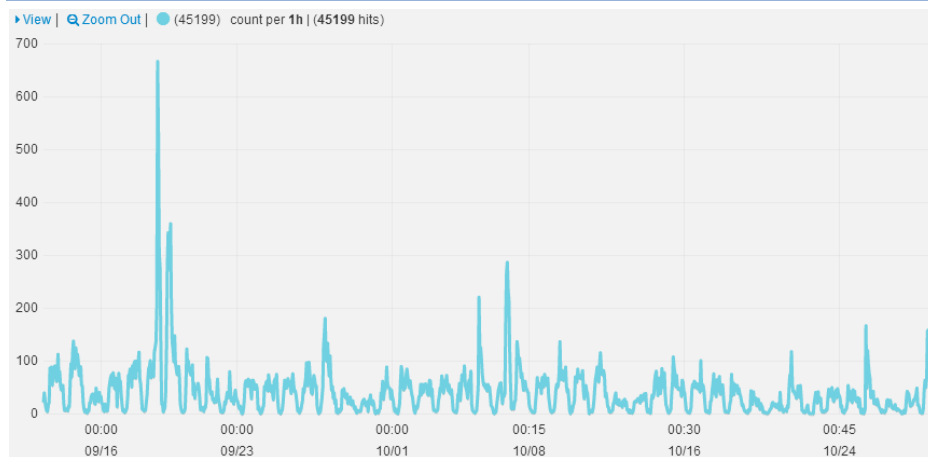
Must exclude any one of: (containing one should filter tweet away)

"braunkohle" "steinkohle" "summe" "gesamt" "windkraft" "der" "infolengkap" "für" "donde"
"hubo" "fuego" "una" "oxidación" "violenta" "combustible" "con" "calor" "vapor" "agua" "el"
"emissor" "emissions" "kerja" "que"
(om andere talen buiten de deur te houden)

Zie:

<http://149.210.130.23/kibana/index.html#/dashboard/elasticsearch/TNO%20Prinsjesdag%20-%20Bouwsteen%20Milieu-duurzaamheid>

Tijdslijn met deze bouwsteen:



De pieken rond 5-6 oktober komen overeen met onze casus windmolens.

Appendix H

Setup for Event Detection

Proces:

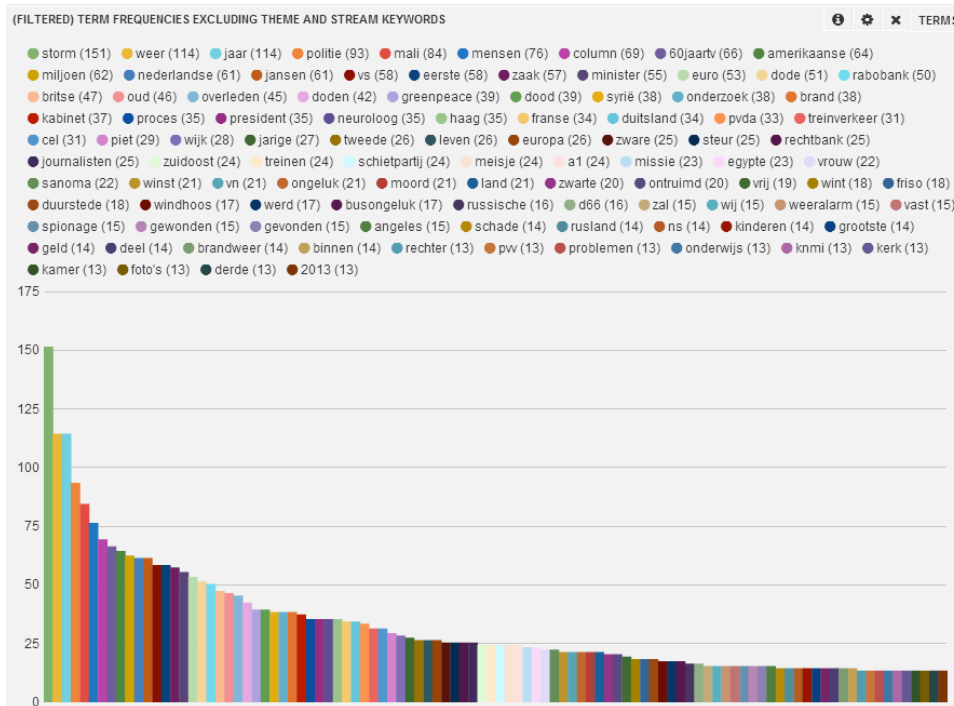
1. Data inladen
2. Dashboard configureren.
 - a. Zie voorbeelden:
 - i. <http://149.210.130.23/kibana/index.html#/dashboard/elasticsearch/TNO%20Prinsjesdag%20-%20Nick%20-%20202>
 - ii. <http://149.210.130.23/kibana/index.html#/dashboard/elasticsearch/Monaco%20FO>
 - b. Grotendeels naar eigen wens. Met name belangrijk: blok met term frequencies.
3. Queries definiëren, nu:
 - a. All tweets, marked tweets, noisy tweets, archived tweets
 - b. Thema's
4. Filters instellen:
 - a. Exclude geclassificeerde data (optioneel)
 - b. Tijdsfilter (optioneel)
5. Configureren van term frequency blok
 - a. Bekijken van terms, en vervolgens ignoreren wat je eruit wilt hebben, in paar deelstappen:
 - b. Algemene en generieke ignore terms toevoegen. Nu voorlopig:
 - i. `http,t.co,de,van,het,een,op,voor,en,bij,nederland,met,nieuws,niet,heeft,fd,anp,aan,zijn,over,naar,dat,die,amsterdam,te,rt,na,om,geen,wil,uit,er,meer,tegen,persbericht,als,cultuurenmedia,ook,nog,hebben,moet,gaat,twee,nu,tot,wordt,man,je,we,al,dan,maar,ik,video,af,den,kan,worden,veel,grote,zo,ze,dit,kunnen,onder,zich,ndnl,door,vandaag,wat,via,maandag,dinsdag,woensdag,donderdag,vrijdag,za,terdag,zondag,krijgt,live,moeten,wat,weg,vanaf,komt,rond,wel,tussen,mogelijk,vanwege,staat,dag,va,terug,een,twee,drie,vier,vijf,zegt,v,1,2,3,4,5,6,7,8,9,0,onze,neemt,los,doet,ziet,uur,s,mee,lees,hier,extra,blijven,volgens,alle,opnieuw,hij,der,willen,toch,gaan,n,sinds,week,zeker,komen,ex,dez,e,tijdens,per,niets,groot,goed,hun,blijft,bijna,mag,steeds,alleen,geeft,t,hoe,e,rtl,zit,ons,noord,oost,zuid,west,nooit,new,even,mogen,maken,zoekt,nieuwe,aantal`
 - ii. Deze woorden zouden in een GUI on-the-fly 'weggeklikt' kunnen worden. Nu is het voorlopig kwestie van zien wat je niet wilt en toevoegen aan ignore-list.
 - c. Monitor-specifieke ignore terms toevoegen. Voor Monaco FO gedaan door alle thema-keywords uit database samen te voegen, en toe te voegen als ignores, dus:
 - i. De volgende queries zijn hierbij handig:
 1. `SET group_concat_max_len=204800;`
 2. `SELECT REPLACE(GROUP_CONCAT('value` separator ','), '*' , ') FROM Theme;`
 - ii. `afgeranseld,afgerosd,afgerost,bossen,afgeslacht,afgesnauwd,afranselen,afrossen,afslacht,afsnauwen,agressie,calamiteit,agressief,agressieve,ak47,appelwek,afslaan,bakka,batten,bedreiging,chieren,bedrijgt,bek,verbouwen,bivakmuts,bivakmuts,blaster,blemmen,diddih,djo,eken,djoeke,doden,dokken,doodschieten,doodschoppen,doodschot,doodsteken,doodt,doowd,elkaar,rossen,ellende,ernstig,opstootje,escalatie,escaleert,escaleren,fighty,fitta,fitti,fitty,flashbanger,ga,je,nakken,ga,je,timmeren,gaat,eraan,gedjoekt,geef,je,klappen,geef,klappen,gepopt,geschoten,gevaar,gevaarlijk,gevaren,gevecht,gewapend,geweerd,geweld,gewelddaad,gewelddadig,geweldplegen,geweldpleger,hak,je,hameren,heibel,hengsten,hoeken,hoofd,afhakken,hoofd,verbouwen,hooligan,iemand,gestoken,ik,boek,je,ik,steek,je,in,elkaar,geslagen,in,elkaar,slaan,incident,incidenten,jonko,kapot,slaan,keel,doorsnijden,klappen,geven,knokken,knokpartij,knuppel,koffoe,kom,vechten,kopstoot,koud,gemaakt,koudmaken,krijgen,tikken,krijgt,tikken,loezoe,maak,haar,kapot,maak,hem,kapot,maak,je,af,matten,mishandeld,mishandeling,mishandelt,mollen,moordaanslag,moorden,nacken,neer,geslagen,neergestoken,neerschieten,neersteken,neerstorten,niffi,noodsituatie,oproer,niffie,nivi,overval,pistool,popo,poppen,popte,ram,je,rellen,ruzie,schieten,schietpartij,schop,je,dood,sgoppen,slaan,smack,smacken,snij,je,keel,snijwond,spikri,steek,jullie,steekpartij,steekwapen,stoffelijk,overschot,stroggel,tantoe,tapanahoni,thopen,tikken,uitdelen,torri,leggen,tuig,tuig,van,de,riggel,van,kant,gemaakt,van,kant,maken,vechten,vechtpartij,verdiende,loon,vermoord,vermoorden,vijandelijk,vijandig,vittie,vuurwapen,wapen,vijand,pistolen,rusland,ontvoer,NSS,activist,269,manco,strooizout,tekort,bevriezing,gladheid,hitte,ijsregen,ijzel,meteoalarm,onweer,orkaan,regen,sneeuwstorm,stormachtig,stormwaarschuwing,waterhozen,watersnoodramp,weeralarm,windhozen,windstoten,windwaarschuwing,zware,storm,overstroming,hoosbuien,dichte,mist,chaos,op,het,spoor,druk,op,de,weg,file,filevorming,geen,trein,geen,treinen,geen,treinen,gratis,koffie,kut,ns,kutns,ns,fail,ns,kut,ns,fail,perron,uitgevallen,trein,rijdt,niet,trein,rijdt,weer,niet,trein,uitgevallen,tunnel,verkeersinfarct,vervangend,vervoer,zet,bussen,in,ontsporen,ontsporen,trein,kettingbotsing,alcyonsecurity,anonops,anonymous,anonyops,antisecl,brenno,c2000,cryptoron,cyber,ddos,digid,gehackt,hack,ictfaal,legosteentje,ncsc,ntisec,operations,opeurope,pastebin,spiritusnl,sql,tango,down,tangodown,trojan,snowden,nsa,cyberspionage,cyberspionage,cyberaanval,cybercrime,3g,4g,acceptgiro,geen,telefoni,afgetapt,elektriciteit,afgetapt,strom,afgetapt,elektriciteit,afgetapt,strom,alvira,atos,worldline,bankieren,bereik,mobiel,bereik,smartphone,bereik,telefoon,geen,verbinding,betalingsverkeer,bic,ccv,chipknip,currence,dataverkeer,een,verbinding,`

elektriciteit, geenstroom, energie storing, energiestoring, geen bereik, geen telefonie, geen data, geen euri, geen euro, geen internet, geenelektriciteit, geen licht, geen ligt, geen stroom, geen stroom, geskimd, geskimed, geskimmed, getapt elektriciteit, getapt stroom, gsm bereik, gsm ontvangst, gsm storing, iban, icp companies, ideal, ik zie niks, incasso, inkomen, internet verbinding, internetbankieren, internetverbinding, kabelbreuk, kortsluiting, lampen vallen uit, lening, licht uit, licht valt uit, machtigen, machtiging, machtigingen, meterkast, mobiel ontvangst, mobiel storing, netwerk bereik, pas ingeslikt, pasje ingeslikt, pik donker, pin automaat, pin automaten, pinautomaat, pinautomaten, pinnen, pinpas, pinpas ingeslikt, rabopas, scherm valt uit, sepa, sepapay, skim, skimmen, slecht bereik, slecht bereik, slechte verbinding, smartphone ontvangst, smartphone storing, sparen, stoppe liggen eruit, stoppen gesprongen, stoppen liggen eruit, stoppengesprongen, storing energie, storing pin, storing telefonie, stroom is eraf, stroom uitgevallen, stroom uitval, stroom weg, stroomnet, stroomstoring, stroomuitval, stroomvoorziening, studentlening, swift code, tapt af elektriciteit, tapt af stroom, tarief payroll, telebankieren, telefonie ontvangst, telefoon ontvangst, telefoon storing, telefoonmasten, telefoonpalen, trage verbinding, viel het licht uit, werd het donker, zie geen steek, zie niks, betal probleem, betalings probleem, gaslek, alexander pechtold, andre elissen, arie slob, barry madlener, bram van oijk, co verdaas, danai van weerdenburg, dick schoof, diederik samsom, dion graus, edith schippers, emile roemer, fleur agema, fractievoorzitter cda, fractievoorzitter christenunie, fractievoorzitter d66, fractievoorzitter groenlinks, fractievoorzitter pvda, fractievoorzitter pvv, fractievoorzitter sgp, fractievoorzitter sp, fractievoorzitter vvd, frans timmermans, frans weekers, fred teeven, geert wilders, geertwilders, halbe zijlstra, harm beertema, henk kamp, ino van den besselaar, ivo opstelten, jeanine hennis-plasschaert, jeroen dijssebloem, jet bussemaker, jetta klijnsmma, johan driessen, joram van klaveren, karen gerbrands, kees van der staaïj, leon de jong, lillian helder, lillianne ploumen, lodewijk asscher, louis bontes, machiel de graaf, mark rutte, martin bosma, martin van rijn, melanie schultz, melanie schultz van haegen maas geesteranus, minister algemene zaken, minister binnenlandse zaken, minister buitenlandse handel, minister buitenlandse zaken, minister cultuur, minister defencie, minister defensie, minister economische zaken, minister financiën, minister financin, minister infrastructuur, minister koninkrijkrelaties, minister koninkrijksrelaties, minister milieu, minister onderwijs, minister ontwikkelingssamenwerking, minister rijksdienst, minister sociale zaken, minister sport, minister van algemene zaken, minister van binnenlandse zaken, minister van binnenlandse zaken en koninkrijkrelaties, minister van buitenlandse zaken, minister van bzk, minister van cultuur, minister van defencie, minister van defensie, minister van economische zaken, minister van financiën, minister van financin, minister van infrastructuur, minister van koninkrijkrelaties, minister van koninkrijksrelaties, minister van milieu, minister van onderwijs, minister van ontwikkelingssamenwerking, minister van sociale zaken, minister van sport, minister van veiligheid en justitie, minister van volksgezondheid, minister van welzijn, minister van werkgelegenheid, minister van wetenschap, minister veiligheid en justitie, minister volksgezondheid, minister voor buitenlandse handel, minister voor economische zaken, minister voor infrastructuur, minister voor milieu, minister voor ontwikkelingssamenwerking, minister voor rijksdienst, minister voor sociale zaken, minister voor sport, minister voor veiligheid en justitie, minister voor volksgezondheid, minister voor welzijn, minister voor werkgelegenheid, minister voor wonen, minister welzijn, minister werkgelegenheid, minister wetenschap, minister wonen, ministerpresident, minpres, raymond de roon, reinette klever, roland van vliet, ronald plasterk, sharon dijksma, sietse fritsma, staatssecretaris sander dekker, stef blok, sybrand van haersma buma, tony van dijck, viceminister, viceminister president, viceministerpresident, vicky maeijer, wilma mansveld, actievoerders, adbust, anarchiel, belastingstaking, betoging, bezetting, boycot, calamiteitenroute, dds, demonstratie, dutchrevolution, freecb3rob, geen schoning, geen woning, graffiti, huurstaking, nlrevolution, leve de republiek, levederepubliek, picketing, project x, protest, protestkanaal, protestlied, publiciteitsactie, publiciteitsstunt, publiciteitstund, raasta roko, republiek, revolutie, samizdat, sitin, staking, stakingen, teachin, vlagverbranding, vredeskamp, terreur, anarchist, extremist, diein, aardbeving, aardshok, drone, schutter, volkert, cbrn, ammoniak, ammoniumnitraat, bacillus antracis, biologisch, borsele, borsele, botuline toxine, cesium, chemisch, cobalt, cyanide, dengue, ebola, explosie, fosgeen, fukushima, gifgas, iridium, kerncentrale, kernreactor, kernreactor petten, lewisiet, marburg, mosterdgas, neutronengif, nitromethaan, nucleair, nucleair transport, nusog, pokken, radiologisch, ricine, sarin, straling, strontium, tabun, waterstofperoxide, yersinia pestis, chemische aanval, yperiet, NSS, verdacht pak, verdacht pak, mexicaanse griep, pandemie, q-koorts, sars, virus, mers, griep, H5N1, vogelgriep, H7N1, ziekte, Zoãnose, bacterie, schimmel, prion, mrsa, pathogenen, virulentie, influenza, H7N9, corona, MERS-CoV, aanrijden, aanslag, ak 47, al kaida, noodlot, al-qaida, al-queda, bom bij, bom in, bom op, bomaanslag, bomb, bommelding, da bomb, de bom, oerknal, doodslag, doodt, doowd, een bom, extremist, geweer, pistool, ill shit, karst t, terrorisme, knal, koets aanrijden, koud gemaakt, krijgen wat ze verdienen, oranje aanrijden, moordaanslag, nctv, scarp, six feet, slagveld, terrorisme, terrorist, van kant gemaakt, van kant maken, verdiende loon, moslimrebelln, Al-Shabaab, vliegtuigkaping, kapen, kaping, krijgt wat ze verdient, krijgt wat hij verdient, Volkert, van der G.

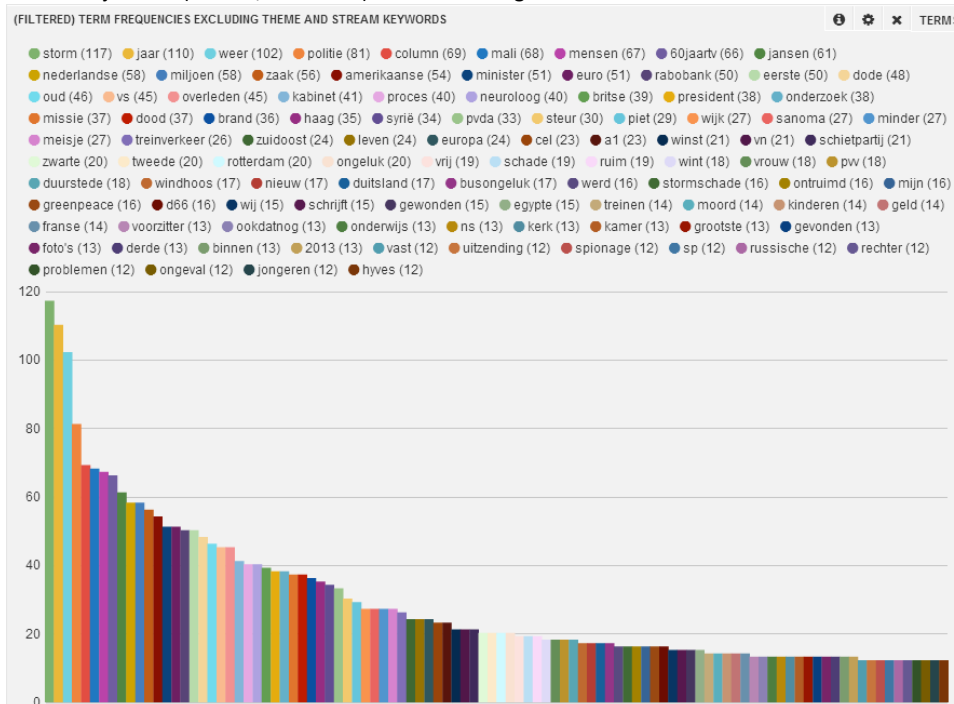
- d. Eventueel stap 4b herhalen met andere parameters (tijdsfilter, classificatie-flag).

6. We krijgen het de volgende beelden:

a. Alleen tijdsfilter (27-10 t/m – 05-11), dus inclusief al geclassificeerd:



b. Tijdsfilter (27-10 t/m – 05-11) en exclusief al geclassificeerde data:



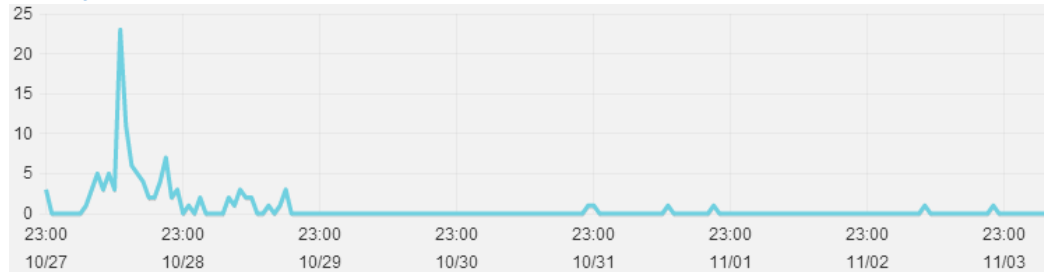
c. Met deze gegevens kunnen we 2 dingen:

- i. Overwegen als mogelijke zoekwoorden / thema's
- ii. Dieper inzoomen

Setup for Event Detection

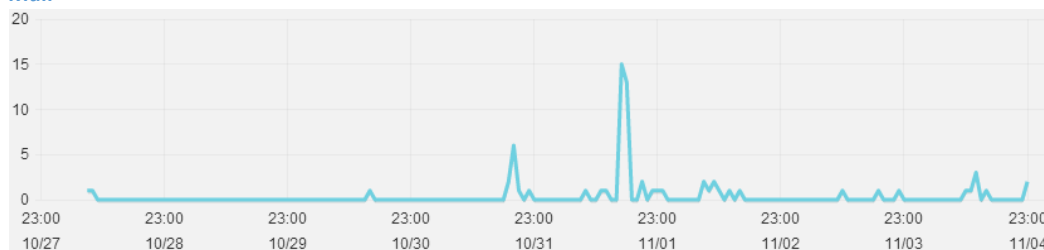
Om dieper in te zoomen, kunnen we het betreffende woord als filter toevoegen, en vervolgens de tweets bekijken. Voor enkele voorbeelden:

Storm of weer



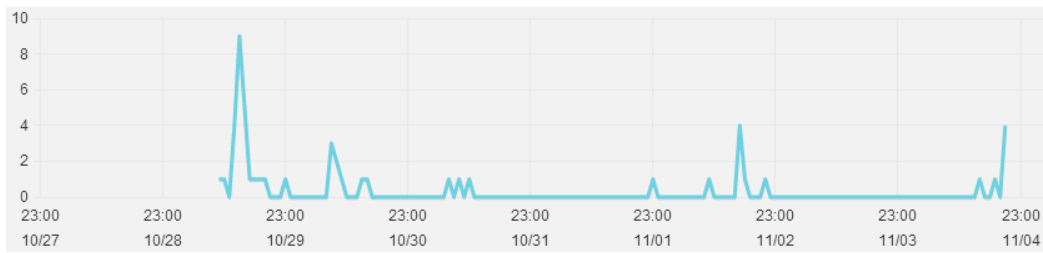
fromUser	text
Mekst	Storm Zeeland,Zuid-Holland geluud: In het zuidwesten van Nederland is de storm over zijn hoogtepunt heen. Het ... http://t.co/ySBjm8is0n
Het_Parool	'Trainingshal Ajax ingestort door storm' http://t.co/FPFpHGIPpu
snel_nieuws	'Stormschade zeker 95 miljoen euro': De storm die maandag over Nederland raasde, heeft een schade van zeker 95... http://t.co/7oMMY1ayo0
telegraaf	Storm verwoest attractie Duinrell http://t.co/jaHntVvacE
nieuws247	[anp] Hulpdiensten ruim 10.000 keer op pad om storm http://t.co/73FPy5osn7 [nederland nieuws]
tekst	Veel Drenten zonder water na storm: In Drenthe zitten ruim duizend mensen zonder water na de storm van gistere... http://t.co/dyOC918EZ9
volkskrant	Tweede dode door storm; jongen in Veenendaal overleden nadat hij afgewaaid tak tegen zijn hoofd kreeg http://t.co/cldaxqBZH4
RTLNieuwsnl	Treinverkeer Amsterdam CS komt langzaam weer op gang: http://t.co/m8Sb3JfIs #storm #amsterdam
NedDagbl	Zwaarste storm sinds 1976: De storm die maandag over Nederland raasde is de zwaarste sinds 1976. De st... #NDnl
RTLNieuwsnl	Storm eist derde leven http://t.co/e59lIQ9n3k
tekst	Dode in A'dam;advies binnen blijven: Een vrouw in Amsterdam is omgekomen toen ze in de storm werd getroffen do... http://t.co/i1MbFlkZe7
Het_Parool	De storm heeft inmiddels twee levens geëist in Amsterdam, de brandweer adviseert iedereen binnen te blijven. http://t.co/dqzXbnuk8

Mali



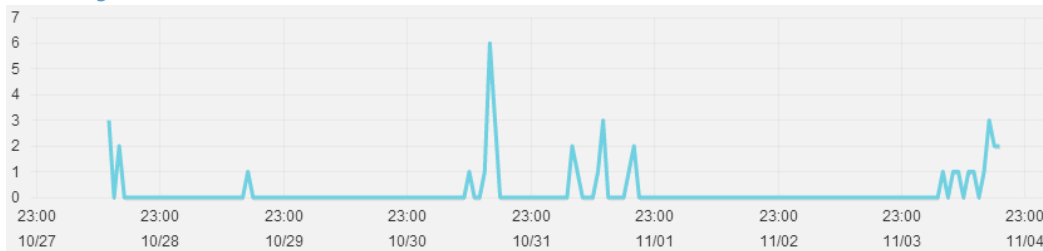
RTLNieuwsnl	Nederland op missie naar Mali: waar gaat het conflict over, wie zijn de strijdende partijen? Lees het hier http://t.co/UR7mhF7MaY
telegraaf	Rutte ziet direct belang bij missie Mali http://t.co/1ZDPea1Wtg
volkskrant	Rutte: 'Missie Mali in belang van veiligheid Nederland en Europa' http://t.co/cYJhBPqaVr
volkskrant	Lees hier de Artikel 100-brief die het kabinet naar de Kamer heeft gezonden over de missie in Mali http://t.co/SbqjaKhtsM
RTLNieuwsnl	NU LIVE: De persconferentie van minister-president Rutte en aansluitend de toelichting over de missie naar Mali. http://t.co/2aH85bEnQX
Mekst	Kabinet neemt besluit Mali-missie: Het kabinet heeft besloten om een militaire missie naar Mali te sturen. De ... http://t.co/0UVvKqWC
NUnl	Kabinet stemt in met Nederlandse deelname militaire missie Mali: http://t.co/DQgi715iDw
volkskrant	Kabinet stuurt militairen naar Mali: dit gaan ze er doen http://t.co/LKq7v9SdMx
nieuws247	[anp] Militairen in beginsel tot eind 2015 in Mali http://t.co/vajDWf1f4P [nederland nieuws]
Trouw_Online	Volgens Koenders is het niet zonder risico's, maar de Nederlandse deelname is zeer gewenst. #Mali http://t.co/1iZxm3wH6j
RTLNieuwsnl	We gaan naar Mali. Het kabinet heeft de knoop zojuist doorgehakt. Nederland doet mee aan een VN-redesmissie. http://t.co/Vbl1HgU2q
NI.nl	Kabinet stemt in met deelname aan militaire missie in Mali: http://t.co/58KA3t167

Rabobank



Teletekst	Megaboete Rabobank voor Libor-zaak http://t.co/D74Nj4Uacw
telegraaf	Rabobank schikt in Libor-affaire voor 774 miljoen http://t.co/hoNoRGIXQU0
nrc	Rabobank krijgt boete van 774 miljoen euro voor Liboraffaire http://t.co/Q3gwt9VLR
RTLNieuwsnl	Topman Moerland stapt op bij de Rabobank: #libor
tftekst	Dijsselbloem:schaamteloze fraude: Minister Dijsselbloem noemt de boetes die zijn opgelegd aan de Rabobank tere... http://t.co/mcvxxH9eFg
tftekst	Libor-boete Rabobank vandaag bekend: Vandaag wordt bekend hoe hoog de boete is die de Rabobank moet betalen in... http://t.co/wuH18BWgh
NU.nl	Rabobank krijgt 774 miljoen euro boete voor Liborfraude: http://t.co/G6XAZCCmOI / Drie vragen over Liborrente: http://t.co/O9a0XCaw1B
volkskrant	Rabobank treft schikking van 774 miljoen euro in zaak Liborfraude http://t.co/ucVFIJZYwE
RTLNieuwsnl	Nu live, extra uitzending rond Rabobank. Mobiel: http://t.co/figmxVzP1g en desktop: http://t.co/HozmVQCJBq
NedDagbl	Rabobank neemt maatregelen na Libor-schandaal: Rabobank heeft tal van maatregelen genomen om te voorkome... #NDnl

Neuroloog



tftekst	AMC tikt neuroloog op de vingers: Neuroloog Rien Vermeulen mag in het AMC in Amsterdam geen patiënten meer beh... http://t.co/F3qqOqA60U
Het_Parool	Neuroloog #AMC geschorst na 'onhandige' uitspraken http://t.co/XKsYFsVCVo
Teletekst	AMC schorst neuroloog per direct http://t.co/wzc8Buj8QR
tftekst	AMC schorst neuroloog om uitspraken: De neuroloog Rien Vermeulen mag in het AMC in Amsterdam per direct geen p... http://t.co/945yBOajTB
snel_nieuws	Neuroloog stopt bij AMC na steun Jansen Steur: Neuroloog Rien Vermeulen stopt bij het Academisch Medisch Centr... http://t.co/F19z1DXKn
telegraaf	Oud-neuroloog verwees naar drugsdealer http://t.co/ndZXQMIaCQ
nrc	Een neuroloog verbonden aan het AMC is geschorst na steun aan ex-neuroloog Ernst Jansen. http://t.co/GJ57MjSPHx
NedDagbl	Neuroloog AMC moet patiënten overdragen: Neuroloog Rien Vermeulen mag niet langer patiënten behandelen i... #NDnl
nieuws247	[anp] Neuroloog geschorst na 'onhandige' uitspraken http://t.co/TvFvixYnZW [nederland nieuws]
NU.nl	'Neuroloog AMC geschorst na steun Jansen Steur': http://t.co/2k6tWVFaNZ
snel_nieuws	'Neuroloog AMC geschorst na steun Jansen Steur': Neuroloog Rien Vermeulen werkzaam bij het Academisch Medisch http://t.co/sWCVGRS1dd