DETERMINING TRUTH IN TWEETS USING FEATURE BASED SUPERVISED STATISTICAL CLASSIFIERS

BAS JANSSEN

Master's Thesis
Databases group
Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente

July 21, 2016

Bas Janssen: *Determining truth in tweets using feature based supervised statistical classifiers*

Supervisors:
Maurice van Keulen
Mena B. Habib

Voor familie en vrienden

ABSTRACT

Social media is getting more and more important in society. Social media is actively used by 32 percent of the world population and the amount of active users has grown 10 percent in 2015. Social media has changed how people communicate with each other and is taking over the way people obtain information such as (financial) news by replacing newspapers and how companies carry out their market research. Social media can be described as uncurated and uncontrolled and its messages can serve as a real-time propagation of information with an enormous reach. In several popular papers, the usefulness of so called social media mining has been shown and this has attracted other researchers to perform similar experiments with social media data. Next to these success stories with social media data, social media can have negative impact on society in which it, among other popular examples, enables rioters a communication channel and enables users to spread false rumours which causes panic in society and thus will have far-reaching consequences.

By understanding this context, the tremendous opportunities to work with social media data and the acknowledgement of the negative effects, a way of determining truth in claims on social media would not only be interesting but also very valuable. By making use of this ability, applications using social media data could be supported (for example by using this ability as a filter step by discarding the false tweets) or this ability can be used as a selection tool in research regarding the spread of false rumours.

In this thesis, we show that we can determine truth by using a statistical classifier supported by three preprocessing phases; filtering, detecting types of facts and extracting facts. We base our research on a dataset of Twitter messages (including meta-information) about the 2014 FIFA World Cup. We determine the truth of a tweet by using 7 popular fact types (involving events in the matches in the tournament such as scoring a goal) and we show that we can determine truth by using a feature based classifier achieving an F1-score of 0.988 for the first class; the tweets which contained no false facts and an F1-score of 0.818 on the second class; the tweets which contained one or more false facts. We show that we can determine truth for the selected kind of facts by using features which determine which fact type the facts in the tweet belong to in combination with features which determine the popularity of the facts (how many times users have repeated the fact), the reach of the facts (how many people were able to see the fact) and the number of replies on the facts in the tweet.

Our discoveries look promising and we expect that there are several situations, which we describe in this thesis in detail, in which the reliability classifier will perform similarly as good as our obtained results. We expect that the classifier only performs well in situations comparable to the dataset we have used in the thesis and that more

research is needed to provide the same results in incomparable situations, for which we offer some advice.

ACKNOWLEDGEMENTS

Met de afronding van mijn afstuderen eindigt een studie van 7 jaar. Sinds 2009 studeer ik op de Universiteit Twente en heb er met veel plezier gestudeerd.

Ik wil bij dezen een enorm lijst aan mensen bedanken die door mijn studie belangrijk zijn geweest. Allereerst wil ik mijn ouders bedanken; met jullie hulp is studeren een stuk gemakkelijker en aangenamer en wanneer nodig stonden jullie altijd voor mij klaar.

De volgenden die ik wil bedanken zijn mijn studiegenoten, voor het ontelbare keren samenwerken en voor het zorgen dat ik een hele leuke studieperiode heb gehad.

Ik wil mijn huisgenoten bedanken, alle actieve studenten bij studievereniging Inter-Actief bedanken en natuurlijk veel familie en vrienden.

Last, maar zeker niet least natuurlijk mijn begeleiders: Maurice en Mena.
Maurice, hartstikke bedankt voor de begeleiding door het afstudeerproject. Je enorm efficiente manier van werken en problemen oplossen en je kennis in het vakgebied heeft er voor gezorgd dat mijn afstuderen een stuk gemakkelijker werd.
Mena, thank you very much for all of your time, your friendliness and support. You provided me with numerous important sources and we often had very good discussions how to progress in the project. Furthermore, you gave me some very useful peptalks in times of need. I'm very happy that you were my supervisor.

– Bas

CONTENTS

1 INTRODUCTION

1.1 PROBLEM STATEMENT

According to Nielsen, Internet users continue to spend more time with social media websites than any other type of website[15]. Web the help of social media, people all over the world can broadcast messages to everywhere, anytime about anything. Although social media is not always reliable, people do rely a lot on social media. According to Reuters [17], social media appears one of the most important ways for people to find news online. This means that social media influences the sources of the news and therefore the interpretation and content. In 2015, Twitter had over 300 million active users and Facebook over 1550 million active users. [1] These enormous amounts of users produce enormous amounts of messages and a lot of them contain factual information. This factual information may contain anything: opinions, news, remarks about recent events; on social media, everyone can engage in a conversation about anything. Because of these vast number of users and because of the diverse number of topics people discuss on twitter, social media has become a widespread, diverge platform containing a lot of factual information which makes it a very valuable platform for a lot of people.

In several popular papers, the usefulness of Twitter data has been shown and this has attracted other researchers to perform similar experiments with social media data. A couple of popular examples are the generation of accurate reports of occurrences of earthquakes, costumer satisfaction analysis by companies and the usefulness of the early spread of news on Twitter. Because of the popularity of social media and the research done with social media data, the question which automatically comes to ask is: how reliable are the messages on social media and should we trust social media messages? If we could determine the reliability of social media messages, this could result in a very interesting preprocessing step to take for researches who work with social media datasets.

A couple of papers have shown that the credibility of social media messages is low. Although there has been done a lot of research in credibility of social media, research relating to the reliability of social media is lacking. Although there is little research on the reliability, the limited research available does show that a lot of people spread false facts through social media and show a couple of examples where Twitter has lead to false spreadings of misinformation. Apparently, social media is not always reliable, and that will surprise no one, given the open uncontrolled nature of social media.

1 http://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

In everyday life, people process a lot of facts, obtained from numerous sources, such as colleagues, friends, newspapers and internet sources. A part of these facts are factually wrong. People make mistakes, spread facts which they deemed to be true or try to spread lies. A lot of people know this, and create internal mechanisms to cope with this such as creating context around a fact, reflect on findings, check someone's authority on a subject etc. Most of these mechanisms are indicators; they do not determine the truth with an assurance, but rather give someone an idea of the probability of truth. Sometimes, an authority or way of determining truth is simply unavailable or lacking. For example, someone who tweets his daily activities around his house, there is no way to validate these facts. Next to determining the truth of a fact, important step in this process, and therefore an important step in this thesis, is the step of knowing what is meant in a twitter message. A lot of facts are not presented in a straightforward way; tweet's content is often brief, contains mistakes, lacks context and is uncurated [2]. An example of such a tweet is: "The first red card in World Cup". In this tweet, there is no reference to which World Cup the author is referencing, the tweet is very brief, there is no reference to a player nor team nor match, but after some investigation determined as false because at the moment of posting the number of red cards in the tournaments surpassed 3. Another example of a tweet is "Algeria achieved a good result against Germany". How do we have to interpret this opinion? Do we have to assume they won, or that because Germany historically has a better team assume the match resulted in a tie or 'acceptable' loss? The last example we will mention is "Marcelo scored in the 11nd minute BRAZIL 0 - 1 CROATIA". Without any context, this tweet is fairly unmistakeable; there is a score, there is a team, there is a player and there is a minute. However, Croatia and Brazil both have a player called Marcelo in their squad: the full name of Croatia's Marcelo is "Marcelo Brozovic" and the full name of Brazil's Marcelo is "Marcelo Vieira da Silva Júnior". A logical reasoning would be that Croatia's Marcelo scored the goal since the score favours Croatia. However in this case, Marcelo from Brazil scored an own goal.

1.2 GOAL AND CONTRIBUTION

In this thesis, we explore if we can find features which can determine the reliability of the facts in tweets. By doing so, we contribute to research in various ways:

- We contribute to the field of information extraction by putting a combination of rule based and NLP algorithms into practice
- We contribute to the field of reliability and credibility of information on social media
- We contribute to the field of feature discovery

2 https://gate.ac.uk/wiki/twitie.html

Furthermore, it is important to state that in this thesis we do not aim to improve natural language programming algorithms or optimize classifiers by hyperparameter optimization. In this project, we are aiming to optimize results by choosing between various classifier and NLP tool options, but improving them is out of the scope of this project.

1.3 DATASOURCE

Sport is a very popular subject being discussed on Twitter. Twitter users like to inform their followers about what kind of sports they like, which matches they follow, how a particular match progresses and their emotions belonging to all of those. Furthermore, athletes and twitter are an inseparable match as well. Just like in any other entertainment sector, athletes use Twitter to keep in touch with their fans. They tweet about training schemes, their views on sport events and private life. Another important part in the relation between Twitter and sports is the increase of industry to use Twitter as a marketing tool. A lot of TV programs also try to influence the social conversation as described in "Social Networks in a Battle for the Second Screen"[12]. Twitter even provides tools and tips for broadcasters to use Twitter to make shows more appealing and provide tools to use Twitter for advertisements[32]. The popularity of Twitter has led to the creation of lots of official accounts of sport associations on Twitter. For example, the FIFA[3][4], UEFA[5][6] and KNVB[7] have one or even several Twitter accounts which they often fill with up-to-date news.

A popular sport event is the FIFA world cup football, held every 4 years. It is one of the biggest sport events in the world, and consequently, many people tweet about it. After the FIFA World Cup 2014, Twitter Inc. reported [31] that Twitter users have sent about 670 million tweets about the world cup, making it the biggest sport event on Twitter. At the end of the finals of the World Cup, knowing Germany won the FIFA World Cup 2014, Twitter reported that users sent a peak volume of 618 thousand tweets per minute.

Needlessly to say, these tweets contain a lot of information. A big part of these tweets contain information, which does not have to be very meaningful for everyone. A lot of tweets, as is well known, cover private affairs: for example how someone is watching the World Cup. Many tweets cover emotions or cover the basics of "watching something" or are not more than a cheer to a team or the sport event. On the other hand, many tweets cover the World Cup in a detailed and comprehensive way. They cover goals, substitutes and yellow and red cards; important events in a match worthy to be mentioned in a summary about the game. Next to true important and true but unimportant facts, there are also a lot of tweets containing false, untrue

3 https://twitter.com/fifacom
4 https://twitter.com/FIFAWorldCup
5 https://twitter.com/uefacom
6 https://twitter.com/uefaeuro
7 https://twitter.com/knvb

information regarding the World Cup. These tweets may contain misguided information, lies or small errors. Many are copied from false sources, contain reversed facts or are not accurately adopted from true sources.

The University of Twente's Database group is hosting a database containing 64 million tweets about the FIFA World Cup 2014. The World Cup is played during 32 days, involving 64 matches. This database was filled by collecting all tweets which contained one ore more of the following hashtags: #worldcup, #worldcup2014, #fifaworldcup, #brazil2014, #brasil2014 and #fifaworldcup2014. An example of a tweet and the corresponding meta-information about the tweet in JSON format can be found in appendix A. Note that those 64 million tweets are original tweets, retweets and replies combined. Although tweets can originate from a website which already prepopulates a tweet, these tweets are still original tweets.

One of the most important remark we can make about the dataset is that Twitter text differs a lot from normal written text a lot of classifiers are trained on. Most classifiers, such as the Stanford NLP classifier[8], are trained on dataset such as the Stanford NER classifier is trained on: the CoNLL-2003 dataset[9], which contain collection of news wire articles from the Reuters Corpus. Twitter messages of course differ a lot in relation to that training set, making it perform less on Twitter messages. In 'Extracting Knowledge from Twitter and The Web'[26], the authors called these Twitter messages "noisy" and "unique". Noisy, meaning it for example contains a lot of grammatical errors, poor formatted sentences and not using capitals were need to. Unique, meaning it contains a lot of new words and using 'SMS language' (abbreviations and slang commonly used with Internet-based communication[10]).

1.4 ASSOCIATION FOOTBALL

Football is a team sport, played between two teams of 11 players each, by making use of a ball. The objective of the game is to score a goal by getting the ball in the goal of the opposing team, scoring 1 point. The team with the most points wins or, if both teams have the same amount of points, a draw is declared. If a match ends in a draw, the game can go into extra time or a penalty shootout, but in this project we only focus on the first 48 matches which can end in a draw. A match consists of 2 periods, each taking 45 minutes. Both periods can be extended by the referee of the match, making it possible for something to happen in the 45+Xth minute of the game. Beware of the notation; the 45+1th minute is not the same as the 46th minute (the first one takes place in the first half and the second one in the second half), but without extra time, the 90+2nd minute is the same as the 92nd minute. Each goal is always credited to one player, even if the player scores in his own goal. In a match, a player can be punished

8 http://nlp.stanford.edu/
9 http://www.cnts.ua.ac.be/conll2003/
10 SMS language - https://en.wikipedia.org/wiki/SMS_language

by a misconduct by receiving a yellow card (a caution) and a red card (dismissal). A player who received a second yellow card in the same game results into a red card. When a player received a red card, the player has to leave the field, resulting the team to play further with one player less.

1.5 GROUND TRUTH

The FIFA World Cup is FIFA's biggest event and it is documented thoroughly. There are several reports written about every match which contain summarizing match reports, player statistics and heat maps. Although these reports are detailed and extensive, they are not specifically made to be easily extractable. Using the Open football project[11], we received a prepared list of players, end score and score development for every match in the World Cup. Other needed statistics like substitutions, yellow and red cards and country codes, were extracted manually from the FIFA website[12]. In our research, we focused our work on the group stages of the tournament. This gave us the advantage of the match able to result in a draw and not to be extended by extra time or a penalty shootout. Altogether, this resulted in 48 group games, players scoring 136 goals, coaches substituting 279 times, referees giving 124 yellow card and 9 red card bookings. Altogether, 32 countries played against each other each team making use of 23 players.

1.6 INTRODUCTION TO THE ARCHITECTURE

Before we present the research questions, we want to give a brief introduction to the architecture of the system presented in chapter 3 because the research questions are based on knowledge of this architecture. In this thesis we present an architecture of several phases (components) leading to a mechanism which can determine truth in tweets. In phase one, presented in chapter 3 and 5, we use a feature based classifier to identify and classify and extract different types of facts. In phase two, presented in chapter 3 and 6, we use an extraction mechanism to extract facts classified in the first phase. In the third phase, presented in chapter 3 and 7, we designed a feature based classifier to determine the truth in tweets using input of the previous phases and features based on tweets and groups of tweets.

In figure 1, an overview of the system composition can be found. In chapter 3, we will explain the system composition more in detail.

11 Open football - http://openfootball.github.io/
12 www.fifa.com

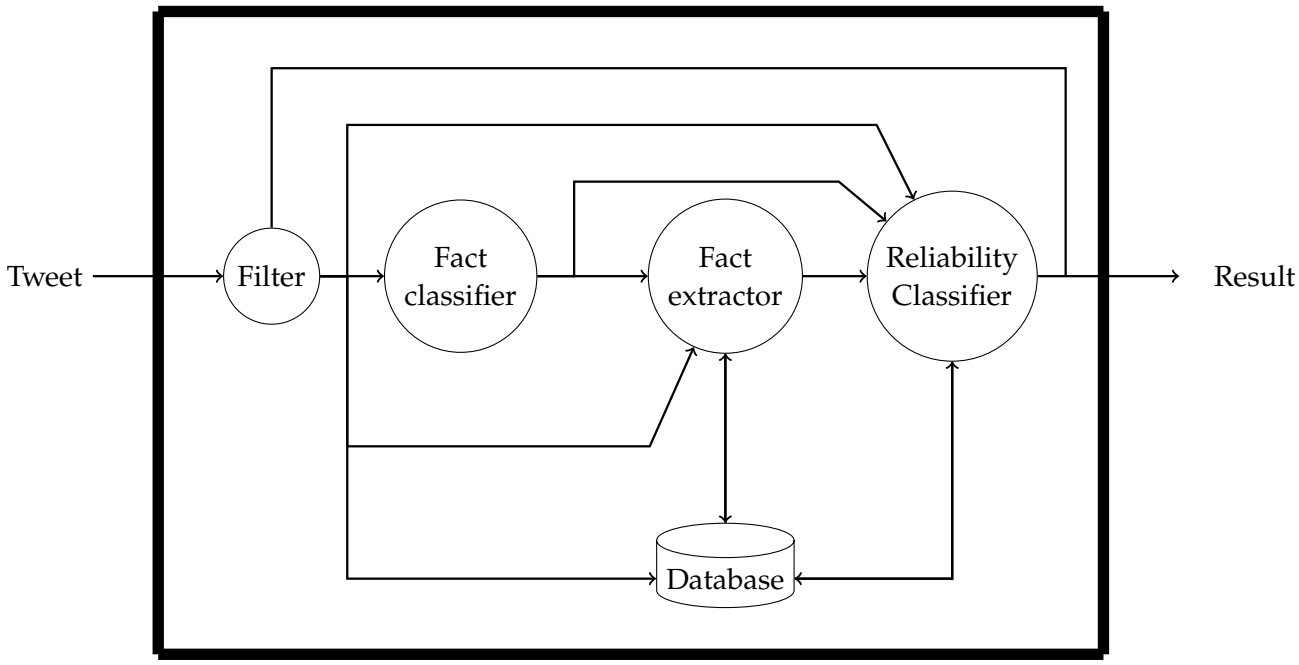

Figure 1: Overview of the system composition.

1.7 RESEARCH QUESTIONS

The main research question of this research project is:
Can we automatically determine truth in tweets using feature based supervised statistical classifiers?

From this, we have gathered the following research questions that need to be answered before we can provide a solution to our problem:

1. What architecture can we compose and implement to test and train features and classifiers to check if facts presented in tweets are true or false?
2. How effective can we extract facts from tweets?
3. What kind of features can we design and implement which determine truth in tweets?

1.8 RESEARCH METHOD

For the first sub-question, we need to design and implement an architecture where it is possible to do the following:

- We want to be able to determine the attributes of messages, attributes of collections of messages and correlations between them.
- We want to define (multiple) steps where we can determine which facts are present in messages and if those facts are true or not

- We want to be able to save and query data about the Twitter messages and FIFA tournament in every part of the architecture

For the second sub-question, we want to design and implement a system where it is possible to extract facts from twitter messages.

For the third sub-question, we analyse if the designed implementations also work in a live environment. For example if the system can be deployed to work as an online service and be able to perform. If the design does not function directly in such an environment, we discuss the specific adjustments it will need to do so.

1.9 STRUCTURE

This thesis is divided into 9 chapters, starting with this chapter containing an introduction to the thesis. In chapter 2, we provide the reader with background information of the topics of this thesis and introduce various concepts which has an essential role in this project. In chapter 3, the architecture of the prototype is laid out and explains how we are going to answer the research questions in this chapter. After this chapter, which gave an introduction to the implementation, each following chapter, from chapter 4 till chapter 7, describes the introduced parts of the system individually. In each of those chapters, we lay out the ideas behind that part of the system, explain the implementation detailedly, provide the evaluation of that part of the system and complete the chapter with a discussion and future research ideas.

2

LITERATURE STUDY

2.1 TRUTH, RELIABILITY, CREDIBILITY AND SOCIAL MEDIA

The truth - the real facts about something, the things that are true
Truthful - telling the truth, containing or expressing the truth
Trustworthy - able to be relied on to do or provide what is needed or right, deserving of trust
Reliable - able to be trusted to do or provide what is needed, able to be believed, likely to be true or correct
Credibility - the quality of being believed or accepted as true, real, or honest
from [1].

Research on truth, truthfulness and trustworthiness are immensely old and have occupied scientist and philosophers for thousands of years. The Roman Governor Pontius Pilate's famous question "What is truth", around 30 AD, was answered by Jesus Christ with "And ye shall know the truth, and the truth shall make you free". Research on truth has been part of research subjects around natural laws, human conscience and peoples desire for justice. People admit using deception in 14% of emails, 27% of face-to-face interactions, and 37% of phone calls. [13] One in four persons lie on their Facebook profile. [25] The opposite of telling the truth, deception, is an everyday aspect of social interaction, and has been researched extensively[24]. Where information is present, deception can be present. Research addressing deception is very broad: ranging from very old communication channels like handwriting[21], nervous system analysis (lie detectors) to new media such as Wikipedia and all kinds of social media platforms.

An interesting but simple practical example of the importance of the pursuit for distinction between truth and deception is the work of the police and Ministries of Justice. In a justice system for example, the investigations of prosecutors is an example of search for the truth of an offence on a higher level. The investigations of the police through forensic research, interrogations and for example by making use of lie detectors is an example of search for the truth on on a lower level.

Next to the research on reliability of what persons say in front of you, a lot of research has been done on reliability of printed media. On elementary school, children learn how to judge different types of media. They learn that they should not trust every kind of source, but also learn which kinds of source mostly are trustworthy. Society holds a lot of opinions about reliability of different kind of newspapers, especially the popular ones. A lot newspapers get a label for be-

1 Merriam-Webster.com. 2016. http://www.merriam-webster.com (9 April 2016)

ing 'quality press' or being sensational or opinionated. In NRC Next, a popular Dutch newspaper, a daily column 'NRC Checkt' is dedicated to check recent popular quotes or recent presented results in other newspapers or media articles. Because the media landscape has changed a lot in the recent years, with the rise of online media and the decline of readers of printed newspapers, there has been a lot of debate about the reliability of news and sources of news. A lot of journalists talk about the new ways of news spreading and the credibility and reliability coming along with those. [3]

A good and interesting example around the reliability of information source is the reliability of Wikipedia. Since the founding of Wikipedia in 2001, Wikipedia has grown to a very important source of information in a short time for many pupils, students and the rest of society. There has been a lot of research about the reliability of Wikipedia[34]. Many of the reports differ in outcome but the latest studies say that although everyone should approach Wikipedia with caution, since 13 percent of the articles on Wikipedia contain errors [6], articles were as reliable as other sources comparable with Wikipedia. Next to the reliability of Wikipedia, a lot of papers present research regarding the credibility of Wikipedia. In 'Predicting Trust in Wikipedia Articles'[7], Cheung presents research where so called 'surface features' (references, links, pictures) predict the credibility of a Wikipedia articles. In this research, users are asked how credible they perceive a certain Wikipedia article and Cheung tries to correlate the credibility scores to these surface features. In 'Evaluating WikiTrust: A trust support tool for Wikipedia' [20], a tool is presented which assigns a credibility score to words in a Wikipedia article. By using features based on the accepted and reversed edits, by using the revision history of an article, WikiTrust is able to build up a reputation of an author. By making use this reputation, it can assign a credibility score to a particular piece of an article edited by a specific author. Tools determining the credibility of Wikipedia articles are not unique. In 'Extracting trust from domain analysis: A case study on the Wikipedia project'[9], Dondio et al also present a tool comparable to WikiTrust, building a model using features such as article length, number of edits and the amount of discussion about an article. In another tool presented by McGuinness et al [23], trust in a Wikipedia article is computed by using a 'link-ratio'; a feature based on the number of times an article is linked to by other Wikipedia articles and the number of times that the topic of an article is mentioned but not linked to.

Social media is immensely popular. In January 2016, there were 2.3 billion active social media users and the prediction was an annual growth of 10 percent. [8] Spreading news and opinions via social media is very popular and has played a very important role all around the world. In the "Arab Spring", social media has been very important to spread news, rumours and to serve as communication tool between different parties in several revolutions across the Arab world in 2011. [14] As a communication tool, Twitter has been used and functioned well during natural disasters, like hurricanes, wildfires and floods

[5]. Next to being a communication tool, Twitter has been used as a source of data to create prediction models of earthquakes. In Japan, Twitter is used to detect earthquakes real-time by using keywords in tweets as machine learning features.[27] Social media is one of the most important ways to find news nowadays. [17]. One example is that financial news on social media seems to have a very high impact on consumers. Social media also is important in the development of reputation and brand preference for organizations.[1]

The amount of research regarding veracity in social media is limited. According to Castillo in 'Information credibility on twitter'[5], a lot of Twitter messages are truthful, but Twitter is also used to spread misinformation and false rumours, although most of that is done unintentionally. The amount of research regarding the perceived credibility of online news is higher. In 'Journalisten: social media niet betrouwbaar, wel belangrijk #SMING14'[3], it is indicated that the perceived credibility of social media is rising, now at the same level of television and radio, but that the perceived credibility of newspapers is still much higher. A recent research report indicated that the credibility of online news such as social media is heavily linked to the presentation of the news and not so much correlated to the content of the news. A research report published in 2009 stated that if the same news headlines were shown in various types of online media, participants gave Twitter the lowest credibility score. [28]. According to Castillo in 'Information credibility on twitter'[18], this is caused, among other things, by the increased number of spam on Twitter and many big incidents of misinformation spreading.

A very interesting research project which is heavily related to this thesis is the European funded Pheme project[2]. The Pheme project is named after the goddess of fame and rumours. The Pheme project is a 36 months research project establishing the veracity of claims made on the internet. Two prominent case studies in the Pheme project cover information about healthcare and information used by journalists. Another interesting paper we found which is very much related to this research is presented in 'ClaimFinder: A Framework for Identifying Claims in Microblogs'.[4] In this paper, Lim et al present a system using existing open information extraction techniques to find claims in a tweets, resulting in subject-predicate pairs. Using these claims, tweets are grouped according to their agreement on events, based on the similarity of their claims. In this way, ClaimFinder is able to group opinions on social media; an important prepossessing task as we will show in this thesis. The credibility assessment task is beyond the scope of this work. Two important remarks we want to make regarding the papers are the goals they are aiming for. Both papers signify the adverse effects of untruths and rumours on social media during times of crisis. The Pheme project recalls the 2011 England riots and the corresponding flood of rumours on social media. In ClaimFinder, they work with a dataset with tweets regarding the MH370 disappearance, containing several rumours, true and false,

2 https://www.pheme.eu/

about the current status and cause of the disappearance of the aeroplane. Both papers show the (increasing) relevance of these kinds of research projects.

2.2 CLASSIFIERS

A statistical classifier is an implementation of an algorithm which classifies an observation to a set of classes. In most classifiers, the observations are accompanied by quantifiable properties/attributes called features. A popular example is an algorithm which classifies e-mails to SPAM or NON-SPAM and uses properties within the e-mail (words, length) to classify it to the right class. There exists numerous classifiers models and in this section, we present the very basics of some of them which are popular in the field of natural language processing (NLP) and data mining. Statistical classification is considered an instance of supervised learning, which means that the classifier classifies an observation based on a training set. The unsupervised equivalent of statistical classification is clustering.

2.2.1 *Decision tree learning ID3, C4.5, J48*

Decision tree learning is a method where a decision tree (see figure 2) is used as a model for classification. The output of the algorithm is the generated decision tree in which the values of the attributes of the sample data decide the class of the sample.

C4.5, developed by Ross Quinlan, is a popular implementation of such an algorithm and was the number 1 ranking algorithm in a highly rated paper called "Top 10 Algorithms in Data Mining" published by Springer LNCS in 2009. [35] C4.5 is based on ID3, also developed by Ross Quinlan. ID3 builds a decision tree by making use of entropy (amount of information) of an attribute. In its algorithm, it select the attribute to be a node of the decision tree by calculating its information gain and selecting the attributes with the highest information gain first. C4.5 is an extension of ID3 which, in relation to ID3, is improved on for example accepting both continuous and discrete features (attributes) and handles incomplete data points. J48 is the open-source Java implementation of C4.5.

2.2.2 *Support vector machines*

Support vector machines is a learning model and associated algorithms where attributes are transformed to points in space and computing an optimal separating hyperplane to classify the data two one of the two classes.

As can be seen in figure 3, the attributes (or features) are found on the axis and samples are plotted in the space. In its algorithm, it computes the optimal hyperplane $w \cdot x + b = 0$ which separates the two classes so that the distance between them is as large as possible.

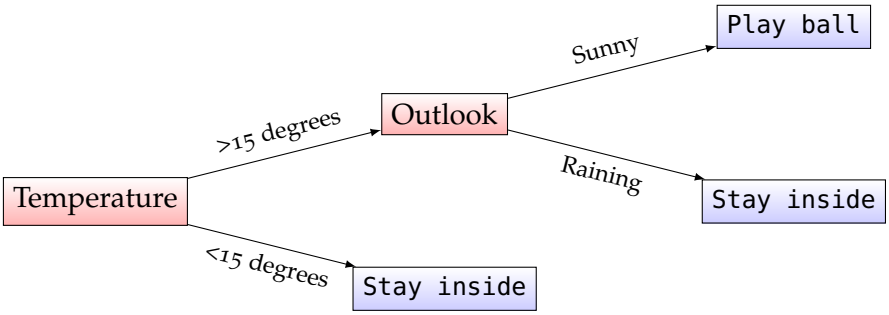

Figure 2: Example of a simple decision tree representing a choice to play ball or stay inside by making use of two decisions: temperature and outlook.

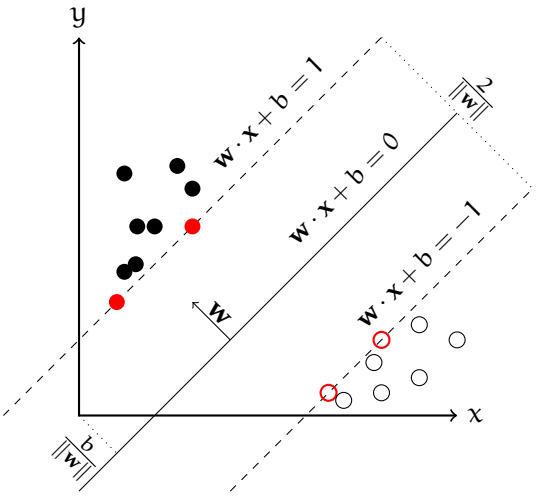

Figure 3: Example of a computed hyperplane of a Support Vector Machine with the support vectors in red.

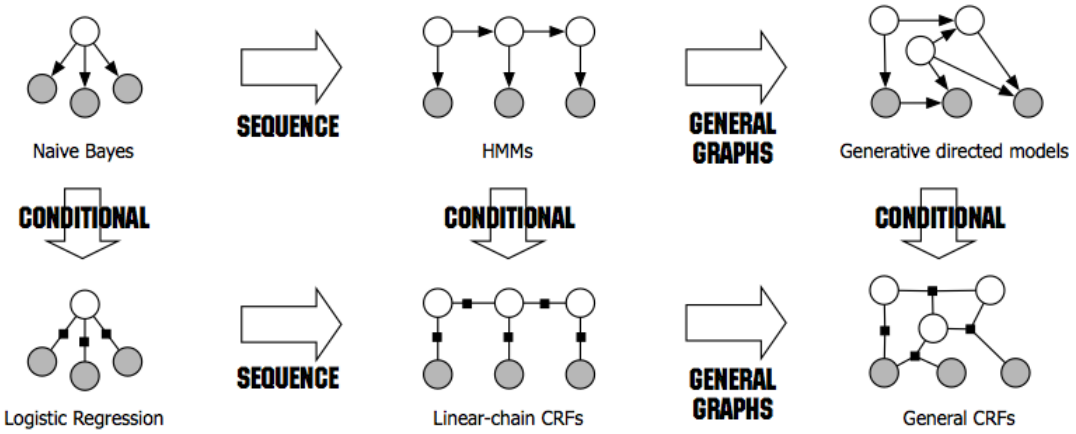

Figure 4: Diagram of the relation between several related classifiers expressed in directed and factor graph (source: 'An Introduction to Conditional Random Fields' [30])

2.2.3 *Hidden Markov model, conditional random field*

A hidden Markov model is a Markov model modelling a Markov process with 'hidden' states which output (emit) visible observations.

As can be seen in figure 4, the observations (in grey) are dependant on the hidden states (white) which are dependant on only the previous state and not on other states before that by therefore can take context into account. In the figure, the relation between the other models is expressed, for example the relation between hidden Markov models and conditional random fields.

2.3 NATURAL LANGUAGE PROCESSING TASKS

In this section, we present several natural language processing tasks. Several of these tasks are used in the implementations in this thesis. Some of them will be presented to give the reader a modest overview of the field.

2.3.1 *Word/sentence segmentation*

Word segmentation is the tasks of splitting up sentences into individual words. In English, the space character is mostly used to divide sentence into words, but not every segmentation implementation uses this way of word segmentation. The Unicode Consortium has published a guideline on text segmentation, applicable for many languages and scripts, using several rule based techniques. [22].

2.3.2 *Part-of-speech tagging*

Part of speech (POS) tagging is the task of assigning words to part of speech categories. In English, there exists 9 parts of speech (nouns,

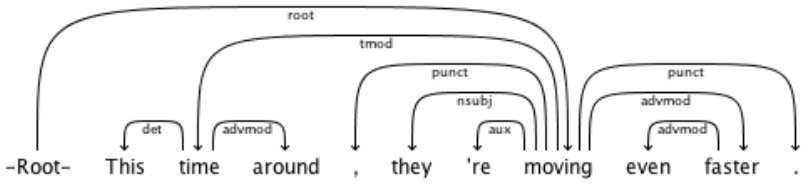

Figure 5: An example of a dependency graph (source: Neural Network Dependency Parser - nlp.stanford.edu)

verbs, adjectives, among others), but most Part of Speech taggers use the Penn Treebank Project's part of speech tag list, consisting of 36 part of speech tags. Part of speech taggers use both the definition of the words and its context among other words to determine in to which part of speech a word belongs.

One of the more classical approaches of part of speech taggers is using (higher order) Hidden Markov models. In this model, the POS-tags are hidden states which must be calculated/optimized using the visible observations which are the words in the sentences. In this way, the part of speech tags are dependant on the previous part of speech tags. Although hidden Markov models are a great way to visualize part of speech tagging, it is just one of the many models used nowadays to perform part of speech tagging.

2.3.3 *Grammatical dependency parsing*

Grammatical dependency parsing is the task of identifying the semantic relation between the words in a sentence. In most NLP tools, the dependency parsing builds a semantic dependency graph of the words in a sentence. An example of such a dependency graph can be seen in figure 5.

2.3.4 *Named entity recognition*

Named entity recognition (NER) is the task of locating and classifying words or word sequences into categories, often being names of persons, locations and organizations. A popular model approach of Named entity recognition is using a Conditional random field (see 2.2.3) and use a combination of features including the words surrounding the word to be classified, the part of speech tags of the surrounding words and local features (of the word itself) such as the word shape (for example the ending characters of the word and if the word starts with a capital). In 'Incorporating non-local information into information extraction systems by gibbs sampling', Finkel et al described a popular named entity recognition model. [11] An example of named entity recognition can be seen in 6

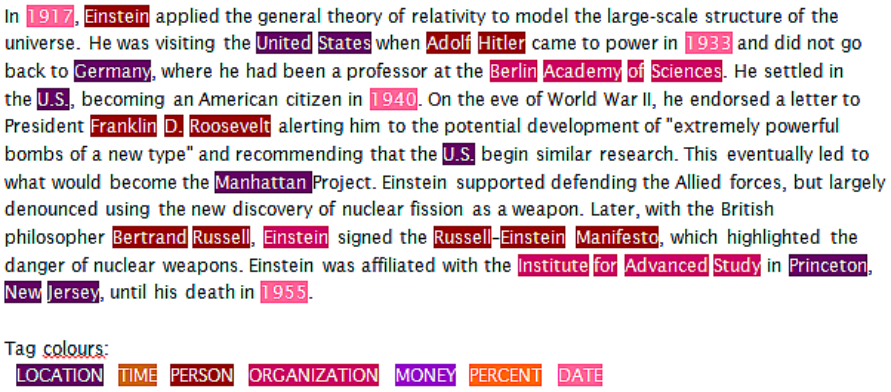

Figure 6: An example of a Wikipedia Article tagged with the Stanford NER tool (source: http://nlp.stanford.edu:8080/ner/)

2.3.5 *Sentiment analysis*

Although sentiment analysis is a broad concept, it is mostly known for the tasks of classifying text to an attitude score, mostly ranging from 'very negative' to 'very positive'. Implementations use, among others, a combination of approaches using a bag of words (words as 'bad' or 'angry' as negative sentiment and words as 'happy' and 'good' as positive sentiment), semantic orientation of combinations of words (n-grams) and machine learning approaches.

2.4 NATURAL LANGUAGE PROCESSING TOOLS

2.4.1 *Stanford natural language processing software*

The Stanford natural language processing software is an open source set of natural language analysis tools from the Stanford University's NLP group. Existing out of multiple packages, the software tools offers word segmenting, POS tagging, Named Entity Recognition, dependency parsing and many other NLP analysis functions.

2.4.2 *Noah's ark Twitter NLP*

Noah's ark Twitter NLP is a set of software tools and dataset including a tokenizer, a part-of-speech tagger, hierarchical word clusters, and a dependency parser for tweets. One of the strengths of this NLP suite is that the tools are trained on a tweet corpus and therefore the POS-tagger for example is able to classify several popular 'social media' words to their correct part of speech, for example "ikr" means "I know, right?", and is tagged as an interjection.

2.4.3 *Natural Language Toolkit (NLTK)*

Natural Language Toolkit (NLTK) is an open-source set of libraries for natural language programming for Python. The toolkit includes a tokenizer, POS-tagger, named-entity recognizer.

2.4.4 *Gate*

The General Architecture for Text Engineering (GATE) is open source software consisting of natural language analysis tools. Within GATE distribution, many plugins are available for use including the Stanford NLP models and Open NLP models. GATE offers various graphical interfaces for creation, measurement and maintenance of software of natural language processing tasks.

3

PROTOTYPE ARCHITECTURE

Fact - a piece of information presented as having objective reality

from [1].

In this chapter, we explain how we are going to answer the research questions by giving a detailed implementation plan. We begin with explaining the relation of the research with the dataset and ground truth which is used to construct the foundations of this research on.

3.1 INTRODUCTION

By making use of these two sources, the tweets about the 2014 FIFA World Cup tournament and the ground truth of the FIFA, we construct the foundations of this research which is the prototype. The aim of the research is to find indicators which can determine the truth in tweets, and by making use of the tweets in the dataset and by using the ground truth we are able to realize a prototype which applies this ideal on this dataset. In this thesis, we present a prototype which applies this ideal by defining several kinds of facts (called 'fact classes' in this thesis) and try to find truth determining features for these kinds of facts. Furthermore, when we have determined the performance of these features in relation to these kinds of facts, we can generalize these to other facts and forecasting the performance of other situations.

This research approach will have several implications. After we have determined a final list of types of facts which we can work with, we will not consider other facts in determining truth in a tweet. Therefore, we will actually look if there are truth indicators which determine if these specific facts are true or false. After we have determined how these features perform on these facts, we can analyse how these (or other features) will perform on other facts and datasets.

3.2 FACTS

The facts we chose to asses in this thesis must satisfy the following characteristics:

- They are verifiable. That means that the fact can be checked to be true or false. Some facts, for example personal ones like 'I ate an apple today' are very hard to verify and therefore we are unable to use them easily.
- The truth of the fact is easily classifiable. That means in this project that we only take facts which we can easily classify with

1 Merriam-Webster.com. 2016. http://www.merriam-webster.com (9 April 2016)

"true" or "false" and not something in between. For example 'Robben played well' can rely on opinions. Of course we could design a model which would assign a fair, well balanced score to each player, but we have determined that this is out of the scope of this thesis and those types of facts will therefore not be used in this thesis.

- There must be a decent amount of tweets containing the type of facts, otherwise we will not be able to experiment with the data.
- There must be a decent amount of training data classified 'true' and a decent amount classified 'false', otherwise we will not be able to train the classifier.

As can be seen in table 1 and in figure 7, each fact class has a fact comparison and a fact scope. The first attribute, the fact comparison, we express the type of data which is compared when we check if the fact is true or not. For example, when we check if the fact "Van Persie scored in the 34th minute", which belongs to the "Score minute" fact class which belongs to the minute category, we check if the minute "34" is the correct minute in which Van Persie scored the goal. The second division we make is the fact scope. As can be seen in figure 7, we have determined three fact scopes namely "Tournament", "Team" and "Match". We expect these fact scopes to have an important relation with the performance of the features.

Next to the implemented fact classes, we have also listed several interesting, but unimplemented fact classes in table 2. When we started exploring several options for fact classes, these fact classes did not meet one of the requirements in the list above or we decided not the implement the fact class because it cost us to much time to implement. One of the most important reason to not implement several fact classes in table 2 is the number of tweets in the dataset which contain such a fact. For 'Yellow card count' and 'Country referee', the number of tweets containing this fact was to low to train the classifiers and implementing them would not have a big impact on the total number of facts gathered. One very interesting and popular fact class is the 'Player name'. This fact class is very interesting because it is very popular and we also found that a lot of player names are misspelled.

One of the first observations one could make when looking at the number of facts, is that most of the time, the number of true facts outnumber the number of false facts several times. That is interesting, because before we had no knowledge about these numbers before we collected the tweets. A very important remark to make, is that we only can base this conclusion on the types of facts (fact class) in table 1. We observe for example that number of true and false facts of the fact class 'Yellow card minute' are distributed a lot more balanced: 56%/44%.

Fact class	Fact comparison type	Factscope	Description and example	Number of collected tweets containing the fact class	
				Classified true	Classified false
Red card count (CRC)	Count	Tournament	The fact stating the count of red cards in the whole tournament, when a red card is received by a player. For example: 'Robben received the 3rd red card of the tournament'.	1037	242
Score final time (FT)	Score	Match	The fact stating the final time score of a match. For example: 'Final time NED - AUS 3-0'.	6826	99
Score half time (HT)	Score	Match	The fact stating the half time score of a match. For example: 'Half time Australia - Netherlands 0:1'.	7123	37
Score other time (OT)	Score	Match	The fact stating the score at a random time of a match. For example: 'Netherlands scored! 1-0'.	100	271
Score minute (MS)	Minute	Match	The fact stating in which minute a goal was scored in a match. For example: 'Van Persie scored in the 34th minute'.	3615	617
Red card minute (MRC)	Minute	Match	The fact stating in which minute a red card was received by a player on the field. For example: 'Robben received a red card in the 23rd minute'.	34	42
Yellow card minute (MYC)	Minute	Match	The fact stating in which minute a yellow card was received by a player on the field. For example: 'Robben received a yellow card in the 25th minute'.	235	184

Table 1: List of implemented fact classes

Fact class	Description and example
Yellow card count (CYC)	The fact stating the count of yellow cards in the whole tournament, when a yellow card is received by a player on the field. For example: 'Van Persie received the 2nd yellow card of the tournament'.
Player name (MN)	The fact stating the name of a player.
Country referee (CR)	The fact stating the country the referee of the tournament is originating from.

Table 2: List of researched but unimplemented fact classes

For most fact classes, the dataset is unbalanced and we would of course rather have worked with a more balanced dataset. One of the first suggestions one could come up with is generating the dataset ourselves, by asking a panel of people to create tweets which contain false facts. Of course, the problem we would face if we would implement this strategy is that the attributes that would translate into the features in the reliability classifier in chapter 7, would not be the same as the attributes which would naturally occur in the dataset. In fact, any interference on the dataset will lead to different results which would potentially harm the results.

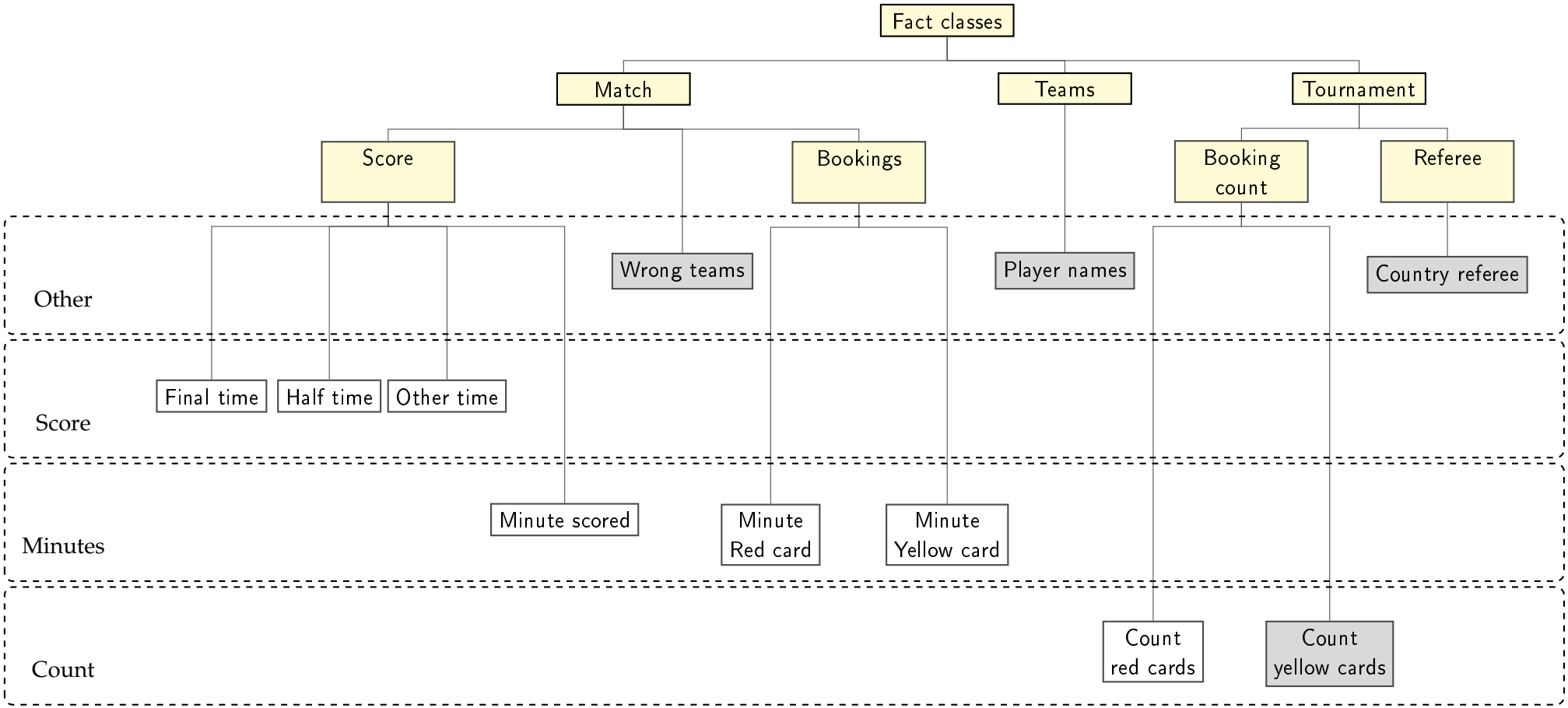

Figure 7: Overview diagram of the facts classes.
The yellow nodes indicate the structure of the fact classes. The white boxes indicate the implemented fact classes and the grey nodes indicate the unimplemented fact classes. Surrounding the facts with a dashed box are the fact comparison types.

3.3 ARCHITECTURAL MODEL

To classify a tweet, we designed a system made up of 5 main parts, as can be seen in figure 8. Each phase of the system mentioned below, except for the filter phase, is supported by the database, which saves and communicates important information to each part of the system. Two examples of functions are saving intermediate results in each system phase and queries in several parts of the system to the ground truth. In the chapters explaining each part of the system, the functioning of the database in relation to that part is described.

3.3.1 *Filter*

When a tweet enters the system, the filter is there to check if the tweet can be classified by the system. There are a numerous amount of tweets which contain advertisements, predictions and spam-like content. This part of the system is made up of a rule-based classifier and is further explained in chapter 4. The input of the filter is the text of the tweet. The output of the filter is the classification if the tweet is used or is filtered out.

3.3.2 *Fact classifier*

If the tweet has passed the filter, the tweet is fed to the fact classifier. The fact classifier consists of several parts; each fact class (type of fact) introduced in section 3.2 has its own trained fact classifier which determines if the tweet contains that particular fact or not. This part of the system is made up of several feature based supervised learning classifiers and is further explained in chapter 5. The input the classifier is the text of the tweet. The output of the classifier is a list of (zero or more) non-duplicate fact classes, listed in table 1.

3.3.3 *Fact extractor*

If the types of facts have been determined by the fact classifier, the tweet is fed to the fact extractor. In this part of the system, when the fact classifier mentions the presence of a particular type of fact in the tweet, the fact extractor tries to extract that type of fact by making use of a numerous amount of techniques, e.g. rule based classifiers and natural language programming. This part of the system is further explained in chapter 6. The input the fact extractor is the text and meta-information of the tweet, the ground truth and the classification output of the fact classifier. The output of the classifier is a list of (zero or more) facts.

3.3.4 *Reliability classifier*

When the facts in the tweets are extracted, all of the results of previous steps are fed to the last part of the system called the reliability

classifier. In this part of the system, the tweet is classified to contain 'false facts' or to contain all 'true facts'. This part of the system is based a feature based supervised learning classifier and is further explained in chapter 7. The input of the reliability classifier is the text and meta-information of the tweet, the output fact classes of the fact classifier and the extracted facts from the fact extractor. The reliability classifier is also able communicate with the database to save and load attributes and data from other tweets. The output of the reliability classifier is false if one of the extracted facts is false or true if all extracted facts are true.

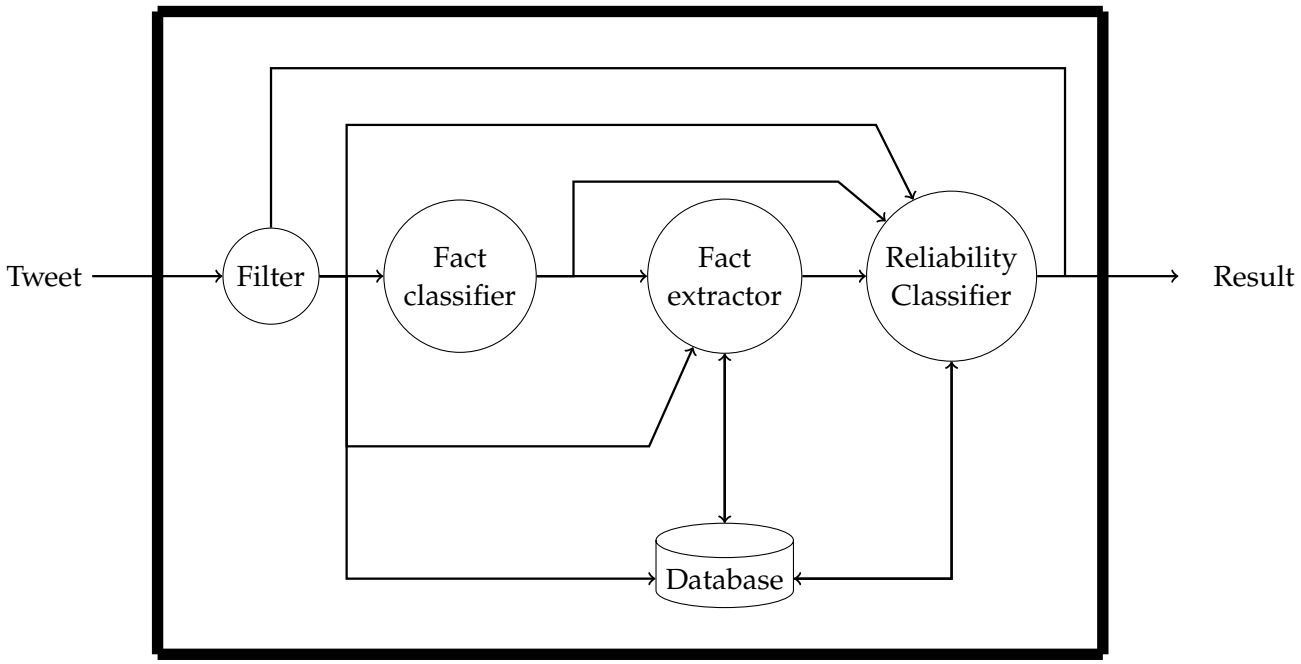

Figure 8: Overview of the system composition. In this diagram, the links between the various components of the system are represented. As can be seen, a tweets enters the system via de filter, which filters unwanted tweets from the dataset. If the tweet has passed the filter, it enters the fact classifier which detects the fact classes in the tweet. The fact extractor, which receives the fact classes from the fact classifier and the tweet from the filter, extracts the facts from the tweets by also making use of the database which contains a lot of domain specific resources. The last component of the system, the reliability classifier, is able to access every part of information in the system. An important part of the performance the reliability classifier is the fact that it can aggregate information of other tweets which entered the system by making use of the database.

4

FILTER

4.1 INTRODUCTION

The central purpose of this part of the system is to filter out tweets which are hurting the performance of the system. Since the start of the thesis, we have investigated the dataset thoroughly by analysing the structure of several messages and querying the dataset numerous times in search for different types of facts. As been said in the previous chapters, we have primarily focussed on the types of facts regarding the FIFA 2014 World Cup tournament. By analysing a lot of tweets, we have come to the conclusion that most of the tweets are about the tournament, primarily because the dataset is build up by making use of keywords belonging to the tournament, but we have also discovered that a lot of tweets do not focus on stating facts regarding the game. Most of those tweets are tweets concerning the daily life of the Twitter user. Those tweets contain messages regarding the experience of watching the tournament, the emotions they have when they do and the things they do between watching the matches. Another big group of tweets contain advertisements or spam-like messages from spam-like Twitter accounts (according to Pear Analytics 4% of the Twitter volume consists of spam [1]). In those tweets, people are invited to gamble on (matches of) the tournament or to visit dubious web pages. Also, there are a lot of tweets regarding predictions of people about the tournament. Those tweets are exceptionally hurting our results because they are very similarly structured as the facts we have presented in section 3.2. Although these tweets will be removed by the filter, these tweets are very interesting to use for other research projects because of their relation to reliability and the opportunities to build models of truth in predictions. A diagram of the filter phase can be found in figure 9.

Input: The input of the filter is the text of the tweet.
Output: The output of the filter is the classification if the tweet is used or is filtered out.

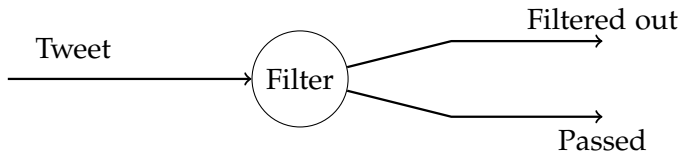

Figure 9: Diagram of filter

1 http://pearanalytics.com/blog/2009/twitter-study-reveals-interesting-results-40-percent-pointless-babble/

Number of tweets in the dataset	+- 63 million
Number of tweets after the filter	+- 61 million

Table 3: Number of tweets filtered by the filter phase

4.2 IMPLEMENTATION AND EVALUATION

The implemented classifier is a rule-based classifier which based on specific expressions in the tweet message, filters tweet from the rest of the system. We have chosen a very minimalistic approach because of two reasons. The first reason is that we only have to filter the tweets which affect the performance in this thesis. It does not have to be a filter for any other purpose. The second reason is that we have chosen for this somewhat 'high precision, low recall' approach is because we want to make sure the tweets which pass the filter, are as 'pure' as possible. In this way, we can more safely work with the dataset after the tweets have passed the filter. This has a direct consequence that we will filter out some tweets which should not be filtered but because we are working with a very big dataset, we expect still enough tweets will pass the filter to ensure a good performance. The number of tweets filtered by the filter phase is shown in table 3.

4.3 IMPROVEMENTS AND FUTURE RESEARCH

One of the most obvious ways of improving the filter is to replace the rule based approach with a statistical a classifier, by building a training set. One of the most hurting tweets in the dataset are the tweets containing predictions about the upcoming matches. A few of them are predictions Twitter users make by themselves, but a lot of them are spam tweets. There has been a lot of research done concerning detecting spam on Twitter. A possible approach could be to implement one of those ideas presented.

5 FACT CLASSIFIER

5.1 INTRODUCTION

The central purpose of this part of the system is to determine which kinds of facts (that is, which fact classes) are present in a given tweet. By introducing this phase in the system, we are aiming for three goals. The first goal is the creation of attributes (features) which other parts of the system depend on to work properly. As will be presented in the upcoming chapters regarding the fact extraction and the reliability classifier, the output of this classifier is directly used by the other parts of the system to work. For example, the fact extractor makes use of the output of this classifier to know which fact classes it has to extract from the tweet. The second goal of the fact classifier is to be able to evaluate the final system more precisely. For example, we are able to draw conclusions regarding certain fact classes or fact comparison types or fact scopes, instead of all fact classes combined together. The third goal of this part of the system is automation. Our goal is to design a completely automatic system which can, given a tweet as input, determine automatically if the facts within the tweets are correct or not. By introducing this part of the system, the system as a whole can operate and be trained more automatically. A diagram of the fact classifier can be found in figure 10.

Input: The input the classifier is the text of the tweet.
Output: The output of the classifier is a list of (zero or more) non-duplicate fact classes.

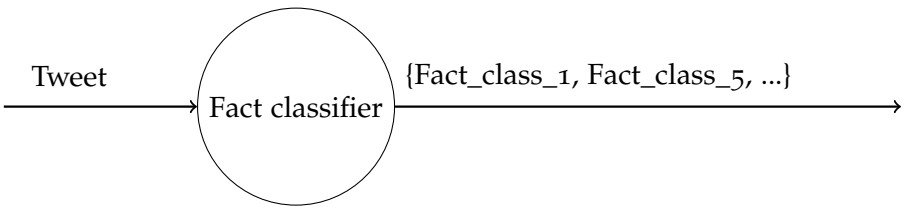

Figure 10: Diagram of fact classifier

5.2 MALLET TOPIC MODELLING

To implement the classifier, we initially implemented a topic model. In machine learning, a topic model is a statistical model for determining "topics" in a collection of documents. When a document is about a certain kind of topic, for example about a particular game like chess, you would expect certain kind of words to appear in the document that can indicate which kind of topic(s) it belongs to, for example the

pieces as "Rook" or "King" or "Pawn". Often, topic models can define several topics and indicate which words belong to that topic, as well as specify for each topic a percentage to a document to point out how much that document belongs to a certain kind of topic, for example 80% Chess and 20% Draughts.

MAchine Learning for LanguagE Toolkit (Mallet) [1] is a package for natural language processing and was mainly developed by Andrew McCallum of the University of Massachusetts Amherst. Mallet contains several NLP tools, including document classification, sequence tagging and numerical optimization and a topic modelling toolkit.

When we conducted the first experiments, we noticed that the classifier has difficulty with classifying different scores. The facts which we introduced in the previous section mostly make use of structure based facts. For example, each fact which belongs to the fact comparison type 'score' makes use of a score entity such as '1-0' (for example: 'Final time Australia - Netherlands: 2-1'). Which particular score-expression (2-0, 1-0, 1-5) is mentioned in the tweet, does not matter to the fact class. But the fact that the tweet contains a score-expression does and the exact location in the sentence does matter. The problem with Mallet is that it sees those different scores as different words and therefore also tries to classify those in different facts. Notations of minutes, players and teams suffer the same problem. To solve this problem, we have replaced various important basic structures such as the score entities with unique words which do not appear in the tweets. An example of such a word, replacing a score entity is '**UniqueMalletScorePattern**'. By doing so, Mallet does not differentiate between those words any more and tweets are not affected by those words. When applying this 'filter', we still have kept it very basic. We did not affect high level structures of entities, for example the ones we would expect at fact level such as replacing a combination of player and score, but only low level entities. By doing so, we have been beware to not introduce preconceptions in the data, in such a way that Mallet is unable to find the relations or in a way that we would force certain relations on Mallet.

After some experiments with the training data, we were discontent about the performance of Mallet. We did not find a proper mapping of words or tweets into fact classes. We have tried to tune Mallet in two ways. In the first experiment, we have asked Mallet to assign as much classes as the number of fact classes, trying for a 1 on 1 match between the Mallet output classes and our fact classes. In the second experiment, we have asked Mallet to assign more classes, trying for a n to 1 match between the Mallet output classes and the fact classes. In the first experiment, we saw that a lot of classes contained the same words which we would expect in different classes, they were far to generic; almost every Mallet output class had a score structure or a 'final time'. When we tried to increase the number of Mallet output classes (to assign one or multiple to one fact class), this afore-

1 MAchine Learning for LanguagE Toolkit (Mallet) - http://mallet.cs.umass.edu/

mentioned problem still remained. Next to this problem, we also still remained with the problem that we were unable to replace every basic entity to a basic entity unique word. Because there are so many different ways of these entities to be expressed (especially players and teams), this steps is far more challenging then expected and because of the poor performance of this part of the filter, Mallet still divides those entities over different Mallet output classes. After those two experiments, we could have chosen to focus more on replacing those entities, which would increase the performance of the filter step and by making use of that step, try to increase the performance of the topic modelling. However, we were not convinced to keep working with Mallet's topic modelling, because we were more and more convinced that the performance was going to rely on the filter step, instead of the underlying model of the topic modelling. Our hypothesis was that several words and structures of characters would form the essence of this classification. Words like 'FT', which is a popular way to represent 'Final time', indicating that a final time of the match has been reached and providing a score to indicate what the score was at that moment. Which kind of players scored a goal does not matter to the type fact, in fact: most words will not matter.

5.3 IMPLEMENTATION

By making use of the conclusions and implementation preparations of the topic modelling experiments, we have designed and implemented the final implementation of the fact identifier. The most important conclusions we have drawn from the previous chapter are the following:

- To identify the facts; important indicators are 'word categories' such as a score indication which can result in several options such as '1-0', '1-2'
- Those combinations of words and words categories will indicate which facts are present in a tweet
- But that we are not sure which combinations of words and which word categories will lead to a good result

By analysing the conclusions, we have chosen to use a feature based classifier, which in our opinion will fit the needs of this problem perfectly. We have decided to implement the classifier by making use of supervised learning. To do so, we have designed 14 features, listed in table 4. A big part of these features target the entities we have presented in the previous sections such as a Minutes, Cards and Counts. Others make use of the ground truth, for example feature 'Player', which uses the list of players from the FIFA.

Adding a new type of fact (fact class) is fairly easy. One has to perform 3 steps:

1. Add a new fact class, by determining a name, description and identifier.

Feature name and description	**Feature implementation (regular expression)**
Feature 'Minute 1'. Targets minute entities. For example '93+3"'.	`(^|[^a-zA-Z0-9\+])([0-9]?[0-9])\s?(\+([0-9]))?["']\s?(\+[0–9])?(\s|$)`
Feature 'Minute 2'. Targets minute entities. For example '32nd minute'.	`\b([0-9]?[0-9]?)(1st|2nd|3rd|[4-90]th)\s(?i)(minute|min(\s|\.|$))`
Feature 'Count'. Targets count entities.	`\b((([0-9]?)(1st|2nd|3rd|[4-90]th))|(?i)(first|second|third|fourth|fifth|sixth|seventh|eighth|ninth))`
Feature 'Yellow card'. Targets parts of yellow card expressions.	`(?i)(((?=.*?\byellow\b))^.*$\|(?=.*?\bbooked\b)^.*$|(?=.*?\bbooking\b)^.*$\|((?=.*?\byellow\b)(?=.*?\bpick(s\|ed)?\b))^.*$)`
Feature 'Red card'. Targets parts of red card expressions.	`(?i)(((?=.*?\bred(card)?\b)(?=.*?\b(red)?card(ed)?\b))^.*$\|((?=.*?\bbooked\b)(?=.*?\bred\b))^.*$|((?=.*?\bbooking\b)(?=.*?\bred\b))^.*$)`
Feature 'Score'. Targets score expressions.	`\b[0-9](\s?)(:|-|v)(\s?)[0-9]\b`
Feature 'Score 2'. Targets score expressions.	`((\#|\b)(?i)(...)\s?[0-9]\s?(\-|\:|\,)?\s?){2}`
Feature 'Score verb'. Targets score verbs.	`(?i)(score(s|d)|(g+o+a+l+s?)+)`
Feature 'Final time'. Targets final time words.	`\b(FT|ft\:|Ft\:|(?i)(final[\-\s]?time)|(?i)(final[\-\s]?(score)?)|(?i)(full[\-\s]?time(\:)?)|(?i)(result\:))(\b|$|\s)`
Feature 'Final time 2'. Targets final time words.	`\b(?i)(beats?|wins?|lose|loss)(\b|$|\s)`
Feature 'Half time' Targets half time words.	`\b(HT|HT\:|ht\:|Ht\:|(?i)(half[\-\s]?time))(\b|$|\s)`
Feature 'Player' Targets player names. The ... in the expression is a list of player name components, for example 'Robben' from the name 'Arjan Robben'.	`(\b|\#)(?i)(...)\b`
Feature 'Player 2'. Targets player names. The ... in the expression is a list of player name components, for example 'Robben' from the name 'Arjan Robben'.	`\(((?i)(...)\s([0-9]?[0-9])(\+([0-9]))?["']](\+[0–9])?\,?\s?)+\)`
Feature 'Other time'. Targets other score expressions. For example (1-0)	`(\(|\[)(\h?)[0-9](\h?)(:|-|v)(\h?)[0-9](\h?)(\)|\])`

Table 4: List of features used in the fact classifier

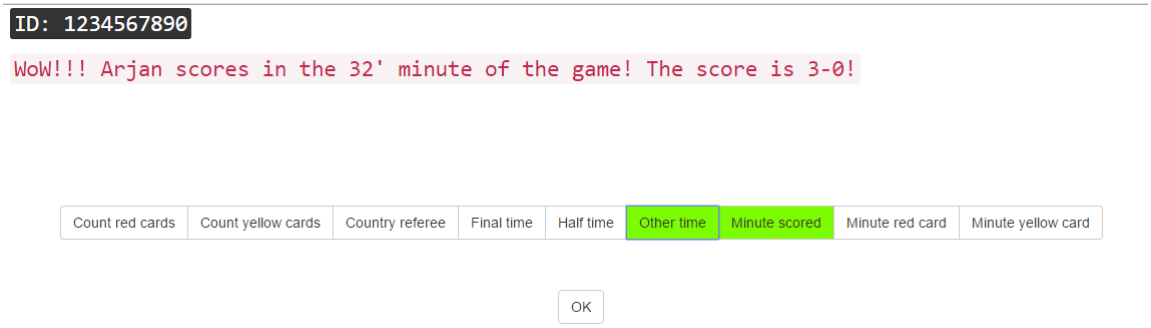

Figure 11: Screenshot of the self-designed tool which we used to annotate tweets which resulted in the training data.

2. Compose features which will determine if a fact class is present in a tweet, and add those to the existing list of features.

3. Prepare a training set which can train the fact class specific classifier and add those to the existing training set.

By making use of the Waikato Environment for Knowledge Analysis 3 (Weka 3) suite, we have experimented with several classifier algorithms. The results of the experiments can be found in the next section.

5.4 EVALUATION

To train and evaluate the fact classifier, we have manually labelled 1883 unique tweets. For each fact class, we have randomly taken a few hundred of tweets (if possible) and manually labelled the fact classes which were presented in the tweets. This resulted in 2341 fact classes being present in 1883 tweets. The results can be seen in table 5 and 6.

Fact class	**Number of tweets containing the fact class**
Red card count (CRC)	267
Score final time (FT)	493
Score half time (HT)	356
Score other time (OT)	531
Score minute (MS)	389
Red card minute (MRC)	68
Yellow card minute (MYC)	203
Total number of tweets	1883

Table 5: List of number of tweets manually labelled tweets

For each fact class, a separate classifier is trained and evaluated. By making use of the Weka suite, we have experimented with several classifier algorithms and achieved the best performance with the

J48 classifier. As presented in the second chapter, the J48 classifier is a Java-based implementation of the C4.5 algorithm and generates a decision tree. By making use of 10-fold cross-validation, we have accomplished the following results:

Fact class	**Confusion matrix** [1]		**Precision**	**Recall**	**F-measure**
Red card count (CRC)	264	3	0.996	0.989	0.992
	1	1614			
Score final time (FT)	457	36	0.991	0.927	0.960
	4	1385			
Score half time (HT)	356	0	0.992	1	0.995
	3	1523			
Score other time (OT)	511	20	0.936	0.962	0.949
	35	1316			
Score minute (MS)	356	33	0.978	0.915	0.945
	8	1485			
Red card minute (MRC)	66	2	0.985	0.971	0.978
	1	1813			
Yellow card minute (MYC)	202	1	0.985	0.995	0.990
	3	1676			

Table 6: List of performance results of the fact classifier. In the confusion matrix column (1), a 2x2 matrix is shown in which, from left to right and top to bottom the following numbers are shown: the number of cases where the corresponding fact class was correctly predicted as 'present' in the tweet, the number of cases where the fact class was incorrectly predicted as 'not present' in the tweet, the number of cases where the corresponding fact class was incorrectly predicted as 'present' in the tweet, the number of cases where the corresponding fact class was correctly predicted as 'not present' in the tweet.

As can be seen in the table, each fact class classifier performs satisfactory. The only fact class which performs below average level is the Score other time classifier. One of the most important reasons for that is that the ways of presenting a Score final time are quite diverse. If we would improve on features detecting Score final time, we would improve the performance of the Score final time (recall) and Score other time (precision).

In the implementation, we are aiming for a satisfactory precision because we believe that a low precision does more damage to the performance of the whole system than a low recall. Because we use a big dataset, a lower recall will still result in a satisfactory amount of results. As you can see in the table, the goal, to attain a good precision while maintaining a decent recall, has been achieved. Because the fact extractor relies heavily on the performance of the fact classifier, we are content with the performance results of this classifier.

The diagram of the fact classifier in figure 10 is updated to the diagram in figure 12.

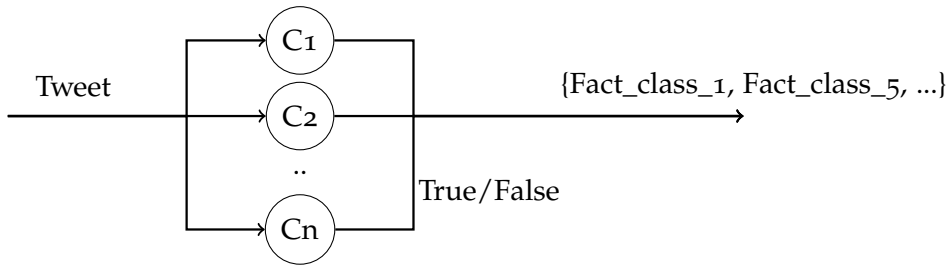

Figure 12: Updated diagram of fact classifier. Each fact class has a dedicated fact classifier with a tweet as input and a boolean as output. The outputs are aggregated and passed on as the final output of the fact classifier.

5.5 IMPROVEMENTS AND FEATURE RESEARCH

One of the most important improvements we can perform on this classifier is the introduction of more refined features. Because of the fusion of several verbs into one feature, the features are getting somewhat coarse. Because of the good results, we have not been tempted to change that, but we think that we can we achieve better results if we introduced some refinement.

Another improvement would focus on the 'Score other time' fact class. This fact class classifier is performing below the average performance of other classifiers. One of the reasons is that the number of false positives is high, which means that the classifier thinks a Score other time is present, but a similar fact class (Score final time) should have been identified. One of the reasons for that is the high diversity of the ways of declaring a Score final time, which makes it hard to identify the fact class. The other reason is that the 'false negative' rate is high, which means that the classifier is unable to determine the presence of a Score other facts in the tweet. One of the reasons we can point out for that are the various ways a score is expressed in the dataset, which if not recognised causes the classifier's inability to classify tweets correctly. A rather obvious suggestion to improve this type of fact is to add more expressions (words, character sequences) to the list of features. We do not think this will be a very sustainable approach. We think that the post promising approach would be to use the context of the fact. Events taking place in the World Cup are often directly posted on Twitter; is little time between the time of Tweeting and the time of the event taking place. A 'Score half time' would likely to be tweeted around half time, a 'Score final time' at the end or a long time after the match. This is also related to the paper "Automatic Extraction of Soccer Game Events from Twitter"[33], in which the authors used the average moment of tweeting to extract the minute of the event in a soccer match. The authors were not entirely satisfied with the result, but the progress they made would certainly improve the fact class detection.

6 FACT EXTRACTOR

6.1 INTRODUCTION

The central purpose of this part of the system is to determine which kinds of facts (that is, which instances of the known fact classes) are present in a given tweet and extract them. The ultimate goal of the system is to determine if the listed fact classes in the previous chapters within a tweet are factually true or factual false. To do so, our aim is to build a classifier (the reliability classifier) which can determine whether this is the case or not. This step, determining which kinds of facts are in a particular tweet, aims for three goals. The first goal is the creation of attributes (features) for the reliability classifier, which will be shown in chapter 7. The second goal of the fact classifier is to be able to evaluate the final system more precisely. For example, we are able to draw conclusions regarding certain fact classes or fact comparison types or fact scopes, instead of all fact classes combined together. The third goal of this part of the system is automation. By making use of the output of this part of the system, we are able to build a training set which can be used to train the classifier models in the reliability classifier, by extracting the facts in tweets automatically and determine their factual truth by making use of the ground truth. In this way, we do not have to annotate the tweets manually, which would otherwise take us an enormous amount of effort.

The general strategy of the fact extractor is to find words or structures of words in tweets. Most of the Twitter messages are written in plain natural language but there are some key differences.

- Twitter users are permitted to Tweet in any language they desire. Because the FIFA World Cup is an international event, the dataset shows a lot of variety in languages used. A lot of Twitter users use different kinds of languages in the same tweet. A specific example is the facts that a lot of users use the words they know; they start their message in English, but use some words from their native language because they are unable to express themselves in English completely.

- Twitter messages are not bound to any structure. Twitter messages are encoded in UTF-8 and that means that a lot of characters become available for people to use. Two examples are:

  1) . . - ; ; - . / ; ; ; ` .:\ |.` : ,، ,| \╲/)\
  /(╲، \´('}// . \ \ / . |،/ / . / / |\ /.／ | | The #WorldCup!

  2) #RETWEET ƑØĻĻØω *´¨) #TFB ¸.•´ #TeamFollowBack¸.•*´¨#TFBJP
  ¸.•´@BioPromotion (¸.•`#VİP¸.•´ ¸.•*´¨) Must(¸.•´Follow (¸.•`…

  Twitter messages are not controlled by an authority, or require the use of proper sentence structures and correctly spelled words.

This means you cannot demand any grammatical structure from the Twitter messages.

- We aim to extract a specific kind of tweet. Those tweets will contain the fact classes we have described and Twitter users use certain kind of structures to express those.

A diagram of the fact extractor can be found in figure 13.

Input: The input the fact extractor is the text and meta-information of the tweet, the ground truth and the classification output of the fact classifier.
Output: The output of the classifier is a list of (zero or more) facts.

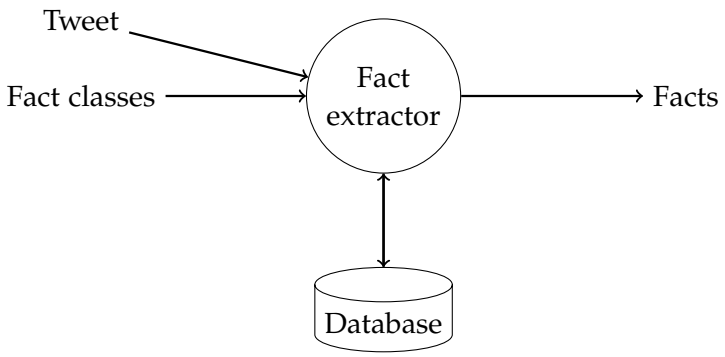

Figure 13: Diagram of the fact extractor

6.2 IMPLEMENTATION

6.2.1 *Introduction*

The extraction of facts works in three stages. The fact extractor receives a list of fact classes from the fact classifier. For each fact class found in the tweet, the fact extractor internally calls an extraction mechanism dedicated for that specific fact class. At the end, all the facts are combined in a list and passed on to the reliability classifier and saved in the database.

The fact extractor is based on several key entities. The most important entity is the one representing the most important concept, the main goal of the fact extractor to output; a fact. In the fact extractor model, a fact is only a very general wrapper concept which serves as a parent for the actual concretization of a fact, which is done by the so called fact classes which we listed in the table 1. The model can be found in figure 14;

Each fact, and therefore each fact class, has a fixed number of pieces of information which makes it a fact in our project. For example, a final score fact always has a match and a score. The match component in a final score match is a unique link to a specific happening (namely the fact that a match has been played) and the score is a piece of information about the happening (namely the fact that the match ended

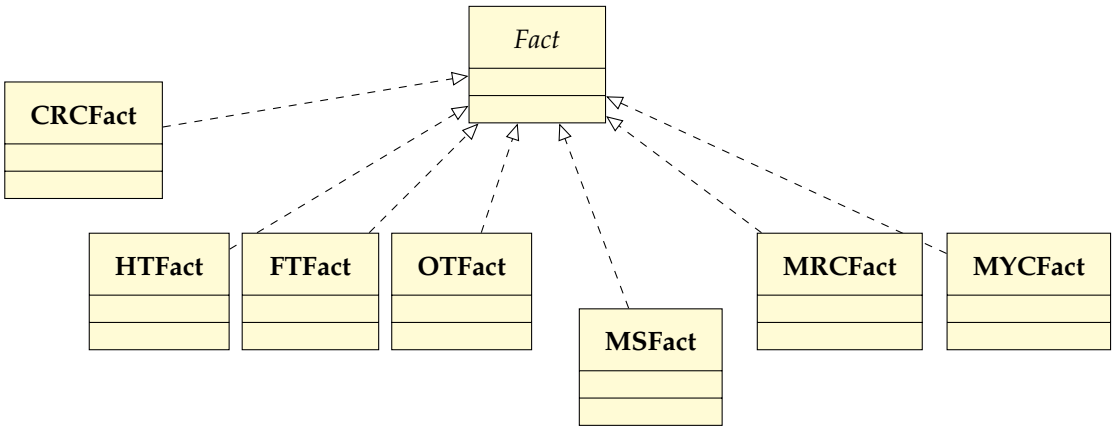

Figure 14: Class diagram showing the relations an abstract Fact and a fact class. As implied in this diagram and described in the upcoming sections, the grouped fact classes (HTFact-FTFact-OTFact and MRCFact-MYCFact) have a very similar implementation and contain the same fact entities.

in 3-2 for example). Because of this fixed combination of information components for each fact, we are able to perform several operations:

- We can point out happenings. Facts are pieces of information about that happening. For example, if that happening existed or when it happened. This is an important step for the other operations, for example, we are able to identify several statements to link to the same happening.
- We can compare facts. For example, a final score fact always has a combination of a match and a score. In this way, we can compare final score facts if identified the matches to be the same.
- We can verify facts. By using the combination of pointing out, comparing and using the ground truth, we can verify if a fact is true or false.

Each fact is made up of one or more of the following entities:

With these entities and a description of the relation between these entities, each fact we implement in this thesis can be described. Therefore, the aim for the fact extractor is to find these pieces of information and find the correct relation between them:

- The fact extractor has to find out how many facts are in the tweet, the fact classifier only determines which fact classes are in the tweet, not how many.
- The fact extractor has to find the information needed for each fact in the tweets. Each fact class needs a certain amount of information to satisfy the conditions of that specific fact class. This information, if available, needs to be extracted from the tweet or meta-information of the tweet.
- The fact extractor has to make sure the information is correctly linked to each fact. Most tweets will contain several facts, even

Match	A match in the tournament, consisting out of two teams. Each match can be identified by an integer linking to a unique match in the tournament .
Team	A team in the tournament, consisting out of a selection of 23 players. Each team can be identified by an integer linking to a unique team in the tournament.
Player	A player in the tournament, belonging to 1 team. Each player can be identified by an integer linking to a unique player in the tournament.
Minute	Every happening in association football is logged by stating the minute in which it happened. Each Minute entity consist of two positive integers. The first minute describes the minute in the normal playing time. If the normal playing time is expired, but the game is still running because of additional time due to for example substitutions or injuries, an additional integer is added in which the minute of the additional time is represented.
Score	The score of a specific match, consisting of two counters counting the number of goals each team scored.
Count	A counter referring to an ordinal number of a specific happening. For example the <1>st yellow card of the tournament.

Table 7: List of fact entities

several facts of the same fact class. If two Score final time facts are present in a fact, the score of one match has to be linked to the right match.

A very valuable piece of information of a piece of text in a tweet linking to an entity is the position of that piece of text in the twitter message. Especially in the fact class specific extractors, see section 6.2.3, we will use the position of entities in the tweet to extract information. To do so, we introduce the ObjectLocation, which will be used by most entity extractors in the next section. An ObjectLocation is a combination of a position (in a Twitter message) and an entity. For example, if a player is found in the tweet, an ObjectLocation will indicate the position of the entity and hold a reference to the entity which was extracted. An example can be found in figure 16

6.2.2 *Entity extractors*

Before we going to focus on the specific implementations of the fact extraction of each specific fact class, we first want to focus on a couple of generic functions which we implemented to support the fact class extractors. Because we already know which information we are looking for for each fact type, we can already specify very specific objectives. We already know a Final time score fact needs a score, so

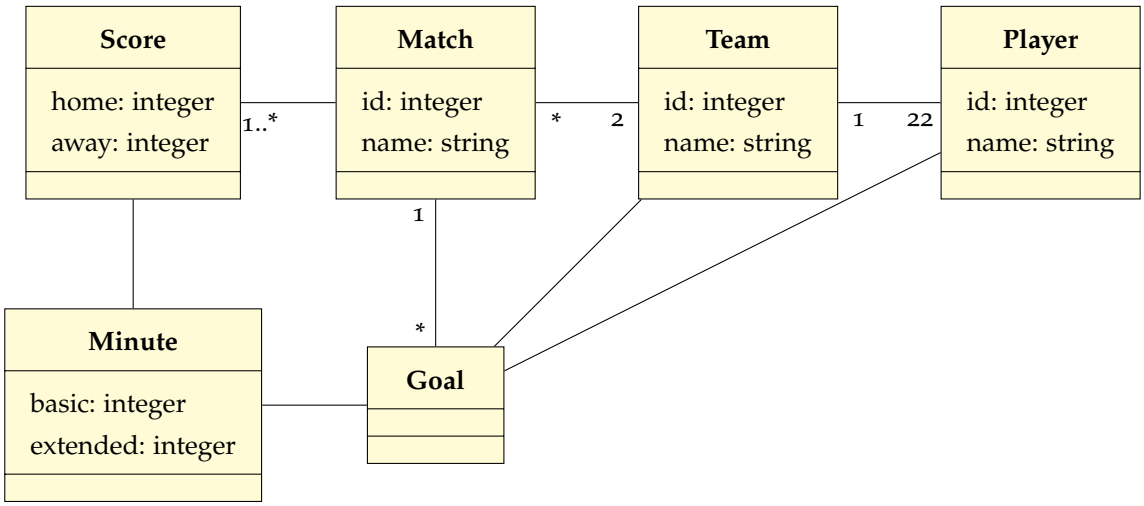

Figure 15: Class diagram showing the relations between the declared entities in the tournament

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
W	O	W		A	r	j	a	n		R	o	b	b	e	n		s	c	o	r	e	s

4 - 16

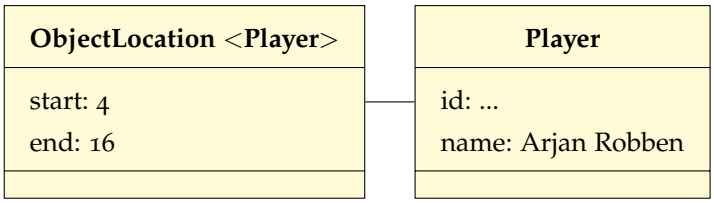

Figure 16: An example of an ObjectLocation

we have to find a way to extract a score from a tweet. We can already implement dedicated extractors for each of the entities (Match, Team, Score, Minute) expressed in the table above. Because of this step, the only thing the specific extractors are concerned with is the interpretation, combination and coordination of those pieces of information. Those are different for each fact class and need to be addressed in the fact class specific extractors.

Player extraction

Player extraction is an important part of the extraction fase. Almost all of the fact class specific extractors use the entity player to extract facts. The player extractor is heavily based on the knowledge of the database about the players participating in the world cup. We make use of two available list of players, one from the FIFA and one from Football.db [1]. We make use of two sources because the spelling of the names which FIFA uses and Football.db use differ a lot. This is because a lot of names are translated from other alphabets (Cyrillic alphabets, Korean alphabet) to the Latin Alphabet and this transla-

1 football.db - Open Public Domain Football Data - http://openfootball.github.io

tion causes different spellings to be used by different authorities. The differences include examples such as "Thomas Müller" vs "Thomas Mueller" or "Hong Jeong-ho" as "Hong Jeongho". This problem leads to the main problem of this task and that is the fact that people make a lot of spelling mistakes when they spell names of football players. A simple example is the misspelling of the name "Marcelo", a Brazilian football player; the name "Marcelo" has been misspelled "Marcello" 3000 times and has been spelled 140k times correctly (resulting in a 2% error rate). These mistakes are made regularly.

The implementation of the player extraction is as follows. Within a tweet, each word gets scanned and compared if it could be part of a name. If there is a match (for example 'Arjan'), this partial match gets saved and the words after the match are checked if they belong to the partial match as well. By doing this we achieve the following:

- We are more sure that a name is found. It could be possible that a word like "rose", which has multiple meanings and is also used as a name, could match a players' name "Rose". If a word sequence "rose jackson" is matched, the probability of "rose" being a name instead of a flower has increased. That is because the name "Rose Jackson" is a popular name and therefore a common occurrence and the bigram "rose jackson" when rose being a flower is less common.

- We can eliminate certain players if for example their first name are similar, but their surnames are not.

- We can achieve a better performance on location determination. If we know more precisely where the names are located and have the knowledge of knowing we are dealing with only one player instead of multiple, we can also do a better job in analysing structures in sentences and therefore receive a better result. For example, because we have

Within the player extractor, to solve some problems matching different spellings of words, the names and tweets are normalized. This means that "Müller" becomes "Muller" and "Mandžukic" becomes "Mandzukic". By normalizing the Tweets and the names, several different spellings are normalized into one.

One of the problems we faced when extracting names was the problem of matching several players because a) names are not as unique as we had hoped and b) in a lot of cases, names are not written out fully, but only a first name or surname is given. We solved a lot of cases by introducing a team filter. Because a lot of tweets mention facts belonging to each other, the team filter first scans the tweet for teams and only extracts names from that tweet belonging to those particular teams, reducing the number of player matches.

Team extraction

When extracting a team entity, the problems we faced were very similar to the problems we faced when extracting players. Like players,

people use a lot of nicknames for their team, ranging from "Holland" when they mean "The Netherlands" to "Oranje", the famous shirt colour of the Dutch team, "Our guys" and many others. Although we faced a lot of similar problems, we also noticed that when referencing teams, Twitter users seem to use a lot less spelling variations. One of the possible reasons is that it is very popular to use hashtags in combination with the abbreviation of the team to represent teams. The most popular abbreviations used is the official FIFA country code for each team which is very similar (but not completely similar) to the official "ISO 3166-1" country codes. For example, Brazil is references by "#BRA" and the Netherlands as "#NED", making it very easy to extract these kinds of notations. One of the reasons to use these hashtags is that the Twitter messages are more accessible to the Twitter search engine and therefore can more easily be found and participate as message in the discussion and in trending topics.

To extract teams, we use a set of words consisting of full names, partial names and abbreviations of teams. We combine these words in a regular expression so we can also detect variations and combinations of the words, but also make sure the words found are truly referencing the teams instead of other entities.

Match extraction

Match extraction has a lot of similarity with team extraction. The main way of representing a match is to refer to the two teams playing against each other. The difference, obviously, with team extraction is that two teams have to be extracted and that the combination of the two teams has to be found. For example, a lot of Twitter messages report results of a specific group in the tournament's group stage (the FIFA World Cup always begins with a group stage, where teams are divided into multiple groups, which play separate round-robins in parallel). Therefore the combination of each team in those tweets are likely (they will all play against each other eventually) and finding the correct combination is more difficult.

In the same way the team extractor does, the match extractor also makes use of several regular expressions (which are mostly combinations of the team extractor's expressions) which make sure the team combinations belong together and refer to a match.

Another type of match extractor we use, which is not part of the match extractor in the previous section, bases its extraction of matches on the time the tweet was sent, using the meta-information of the tweet. When tweeting about the World Cup, as described in section 1.3, it is very popular to tweet updates of the game while the game is still in progress. Because the tournament is spread out over a one month period, most of the matches do not overlap and therefore a tweet can be linked to a unique match. Some others matches are played parallel to each other, with a maximum of two. Tweets in this period of time are linked to a match by also making use of words in the tweet. For example the team in a tweet can determine a match

because a team cannot play two matches at the same time. In this way, also these tweets can be matches to a unique match.

Minute extraction

In relation to the other extractors mentioned above, the minute extractor works pretty simple. As stated in the introduction of this chapter, in Association football, all the happenings in a match are logged using the minute in which the happening happened. By analysing the dataset, we have determined the most popular ways of people to express the minute and use regular expressions to extract them.

Score extraction

The score extraction works pretty simple as well, because the way to present a score is fairly limited. Two of the most used expressions both use the number of goals the teams scored which most sports use. The most used expression combines the scores and uses a separator, for example '1-1' or '1v1' or '1vs1' and many other variants. Another popular expression of a score is locating the number of goals near the team entity. For example like 'Nederlands 1 Italy 0'. We use several regular expressions to detect and extract the various variants. The regular expressions can be viewed in table 8.

`\b[0-9](\s?)(:|-|v)(\s?)[0-9]\b`
`(\(|\[)\h?([0-9])\h?(:|-|v)\h?([0-9])\h?(\(|\[)`
`([A-Za-z\\#]+)\\s+([0-9])\\s?(:|-|v)\\s?([0-9])\\s+([A-Za-z\\#]+)`
`(\#|\b)(?i)(countryPattern)((\h?(v|vs|versus|against|beats?|vs.)?\h?\#?)(?i)(countryPattern))?\b`
`\#?(?i)(countryPattern)\h*(\(.*\)|\[.*\])?\h*(\*?([0-9]))\h*(:|-|v)\h*(([0-9])\*?)\h*(\(.*?\)|\[.*?\])?\h*\#?(?i)(countryPattern)`
`((\#|\b)(?i)(countryPattern)\h?[0-9]\h?(\-|\:|\,|v|vs)?\h?)+`
`(\#|\b)(?i)(countryPattern)\s?([0-9])\s?(\-|\:|\,)?\s?\#?(?i)(countryPattern)\s?([0-9])(?!\s?(-|:|,)\s?[0-9])`
`(\#|\b)(?i)(countryPattern)\h?(\-|\:|\,|beats?)?\h?\#?(?i)(countryPattern)\h?:?\h?(\*?([0-9]))\h?(:|-|v)\h?(([0-9])\*?)(?!\h?(?i)(countryPattern))`
`((\*?([0-9]))\h?(:|-|v)\h?(([0-9])\*?))?\h?\#?(?i)(countryPattern)(vs|v)?(?i)(countryPattern)\h?((\*?([0-9]))\h?(:|-|v)\h?(([0-9])\*?))?`

Table 8: List of regular expressions to find scores (often in combination with teams). Note that 'scorePattern' is replaced with a list of counties participating in the tournament. Each country has multiple multiple expressions, for example England has 'england' and 'eng'.

6.2.3 *Fact class specific extractors*

By making use of the entity extractors, we are ready to focus on the fact class specific extractors.

Score final time (FT)

To extract a 'Score final time' fact we need to extract a combination of a match and a score. The ways to extract a match and a score are presented in the previous sections. The goal of this section is to be sure the combination of the entities belong to each other, such as that a score is not linked to another match in the same tweet (also a Score final time fact) but also that it is not linked to another fact with the same entity, which in this case is a Score half time or Score other time which could exist in the tweet.

As indicated in the previous sections, a Score final time fact is only extracted if it has been found by the fact classifier. Therefore, we know that certain entities are part of a tweet. To extract Score final time facts, the fact extractor begins obtaining an overview to check which entities are present in a tweet. The type of the entities and the number of entities determine which extraction strategy is chosen. The extractor begins determining how many scores, matches and teams the tweet contain. Because of this first step, the extractor can determine if the number of entities can lead to a direct extraction of a fact without making an effort to check which entities belong to each other. If only one score can be found and if only one match can be found, with the condition that the order of the parts of the scores can be linked to the team, the fact can be extracted.

In the second step, different patterns are tested on the tweets. These patterns, sequences of combinations of indicators for entities and series of characters, are step for step applied on the tweet. Each found match is added to a list of Score final time extractions. After each pattern is tested on the tweets, a final check is performed on the list of extracted facts. The patterns we have tested on the tweets are sorted on probability of extracting the correct match. In association football, only one final score exists for each match, so if two scores are extracted pointing to the same match, the first extraction, and therefore the first fact in the list, is the most probable and will be returned as the extracted Score final time fact. Other extracted scores are removed from the list and discarded.

Score half time (HT)

The extraction of a 'Score half time' is extremely similar to the extraction of the 'Score final time'. To extract a 'Score half time', we also need to extract a combination of a match and a score. The three steps which we introduced in the extraction of a 'Score final time', we also use with the extraction of a 'Score half time'. Also the patterns which we use with the extraction of a 'Score final time' are very similar to the the extraction patterns of a 'Score half time'. Only the indicators

which determine if a 'Score final time' is present in the tweet are different from the indicators of a 'Score half time'. This is mostly caused by the fact that a tweet almost never contains both a 'Score final time' and a 'Score half time'. Therefore, the step to distinct these two facts does not have to be executed which makes this extraction very similar to Score final time.

Score other time (OT)

The extraction of a 'Score other time' appears to be similar to 'Score final time' or 'Score half time', but the strategy of extraction is a lot more complicated. Comparable to the previous fact classes, to extract a 'Score other time', a combination of match and score has to be extracted. However, because a 'Score other time' co-exist together with other fact classes within the same fact group (facts containing a score), the extraction strategy is very different to the other fact group extractors.

In the first step of the extraction, the extractor checks if a 'Score final time' or a 'Score half time' is found in the tweet, because the extraction strategy differs for each case. In the first case, if no Final time or Half time is found in the tweet, the extraction of an Other score is almost the same as the extraction of a Half time or Final time. The biggest difference with extracting an Score other time, no matter if a Score final time or Score half time is present, is that the patterns(/structures) around a score entity which are present around a score in a Final time or Half time are not always present around a score belonging to a Score other time. These patterns are used to link entities to each other and make it possible to extract a fact effectively. Several patterns which we use to extract Score final time scores and Score half time scores can be used for Score other time as well, but a big part of the Score other time facts does not have the structure which makes it easy to link the teams to the correct score parts, which makes it challenging to extract. A part of the 'Score other time' facts, namely the ones with a tied score, do not suffer from this increased difficulty, and are pretty common in association football. For these scores, only the relation between the match and the score has to be extracted and not the order of the score. For the scores which are not tied, the score order cannot be extracted instantly. To determine the order, we try to extract the order from other sources in the tweet. Two examples of those sources are hashtags which contain two teams (such as #NetherlandsBrazil) or other facts which already have determined the order of the score (Netherlands 3-0 Spain). Very often, the order of all those sources are identical in a tweet which makes it possible for us to extract the order of a score. Similarly to the extraction of a Score final time and Score half time, the extraction methods are based on a correctness probability. For this fact class, because several scores can belong to a match in contrary to a Final time and Half time where the score is unique, the filter method is not only based on the match. In this filter method, each score-expression is checked to which extracted fact it belongs by making use of the location of the

score-expression using the ObjectLocation. Each score-expression can belong to only one fact, and therefore each duplicate less probable fact is removed from the extraction list. In the second case, in case that a Score half time or a Score final time is found in the tweet, another extraction strategy is used. The main reason for using another extraction strategy is that the way of expressing a Score other time fact is very different when a Half time score or Score final time is present in a tweet. To extract a Score other time, first the Score final time extractor is used to extract every Score half time and Score final time. By making use of the ObjectLocations of those facts, we know which score-expression could be linked to a Score other time, and which score-expressions are linked to Score final/half time facts. In the second step, we check if we can perform the third step. In the third step, we iterate over each score-expression which is not used by the already extracted Score final/half time facts and extract these scores as Score other time (which is differs in regards to the other extraction methods, where we iterate over fact patterns). We can only extract a score-expression as a Score other time fact in step three by making sure we know the order of the teams in a score-expression. In step two, we try to extract this order by using the techniques presented in the first half of this section (we made use of patterns such as hashtags with contain teams).

Minute scored (MS)

The extraction of a 'Minute Scored' is quite different than the extraction of a Final time, Half time or Other time. Firstly the entities which have to be extracted are different. For the 'Minute scored' fact, we try to extract a combination of a Match, Player and Minute. Because the number of entities increases from two to three, the extraction automatically becomes more challenging.

Comparable to the extraction of a Score final time, the 'Minute scored' extractor also checks the entities in the tweet to determine the extraction strategy. If only one Match, Player and Minute is found, using the entity extractions listed in the previous sections, a Minute scored fact can be extracted. After this initial step the main strategy of the 'Minute scored' extractor is initiated. In this strategy, the tweets gets broken down into logical smaller components (how is explained below) and for each smaller component, the same 'main strategy' is applied again. In this way, the tweet is broken down up to three times. Each time the tweet is broken up into smaller parts, parts of the tweet enter a new so called 'extraction phase'. Although the same main strategy is applied on each smaller portion of the tweet, each extraction phases differ a little, which will be detailedly explained below as well. The main strategy consists of the following steps:

- Check the types of entities and number of the entities in the current part of the tweet.

- If an indication of an other fact class with the same fact comparison type (Minute) is detected, do not extract any fact and skip to the tweet breakdown in the last step.
- If only one triple of Match, Minute and Player can be found, return this as an extraction of a 'Minute scored'. If one match is found in the current part of the tweet (or one match was found in previous extraction phases) ánd one player is found in this current part of the tweet, each Minute which is found can be extracted with the combination of the Match and Player as a 'Minute scored' fact.
- If multiple player entities are found in the current part of the tweet, the current part of the tweet is broken up in smaller parts. The found entities of the current part of the tweet are saved because the smaller parts can use these in the previous (third) step. For each of the smaller message parts, each extraction step in this list is performed again.

As stated in the last step of the extraction strategy, the current part of the tweet is split up in smaller parts, leading to different extraction phases.

- Extraction phase: original tweet. The main strategy is applied on the original tweet as well. This extraction phase is similar to every other fact class extractors, which all check if a fact can be extracted without analysing the tweet for a particular fact class relation.
- Extraction phase: lines. In the first split, the original tweet is broken up into individual lines. We have noticed that it is fairly popular for Twitter users to state each Minute scored fact on a different line, which makes it easy for use to utilize this way of Twitter message layout. The tweet is broken up using end-of-line characters.
- Extraction phase: sentences. After the original tweet is broken up in to lines, the tweet gets broken up into sentences. In the dataset, it is common to express different facts in different sentences and by making use of the Stanford NLP software, we can effectively separate each sentence from each other. Note however that the performance of this sentence splitting tasks is not as effective on this dataset as it is on other non-Twitter text. Twitter messages are not written in the same way other text would be written, for example not using the correct punctuation which the sentence splitter is reliant on.
- Extraction phase: part of sentence. After the tweet is broken up into sentences, the most interesting extraction phase is used in which we split the sentence into multiple smaller part by making use of the most common way of expressing the Minute Scored fact in the dataset. Often, when a Minute score fact is

expressed in a tweet, the message looks something like this: "Robben 31', 41 Messi 21'" or '21' Messi, Robben 31'". Often, these expressions are surrounded by parenthesis or brackets, which is a good indicator for us and is used to split the sentence into parts. For each part, a dedicated function is used to extract the Minute scored fact.

To make the Minute score extraction more clear, we will now show an example of a tweet extraction:

```
1 Netherlands 3 - 1 Argentina
2 (Messi 31')
3 (Robben 21', 29' Van Persie 79')
```

In the first phase, the extractor detects a match entity, 2 player entities and 4 minute entities. It cannot make any decisions now, because the tweet is far to complex and it does not know which minute belongs to which player. In the second step, it splits the tweet into lines, which makes it possible for the extractor to extract the first Minute scored fact. It sees that there is only one player and one minute in the second line, which in combination with the detected match in the first extraction phase can be extracted as a Minute scored fact. The third line however is still too complex to extract and moves to the third extraction phase. In this extraction phase, the line will not be split into smaller sentences because the sentence splitter does not detect more than one sentence in this line. In the last extraction phase. The parenthesis are detected which will result in three smaller parts in which the second smaller part will be "Robben 21', 29' Van Persie 79'". This expression is given to a dedicated function which detects that it has to use a left-to-right (player left, minute right) approach to extract a player and then one or more minutes. In combination with the found match, the three Minute scored facts are extracted.

Minute red card (MRC)

The extraction of a 'Minute red card' is similar to the extraction of a Minute Scored. The entities which need to be extracted are the same; a Match, Player and Minute. This extractor also works with a main strategy and extraction phases, but both implementations are a little bit more simple than the Minute scored extractor. The main strategy of Minute red card is:

- Check the types of entities and number of the entities in the current part of the tweet.
- If one Match, one Minute and one Player and a red card indicator can be found in the current part of the tweet or in combination with higher fact extraction phases, return this as an extraction of a 'Minute red card'. In addition to the standard extraction of the required entities, we add a special extraction in this fact class extraction which does not need every required entity to extract a fact. Because a Minute red card fact has special characteristics, such as that every player received not more

than one red card in the tournament, or that the number of red cards in one match did not exceed one, we are able to extract a Minute red card fact by extracting less than the required entities and using the ground truth to add the missing information after the initial extraction. We have adopted this strategy because of two reasons: this fact is very suited for this kind of fact because of the special characteristics of this fact class and because in the dataset, the required entities are often missing, which requires us to modify our strategy to a better performing one.

- If multiple player entities are found in the current part of the tweet, the current part of the tweet is broken up in smaller parts. The found entities of the current part of the tweet are saved because the smaller parts in the next extraction phase can use these in the previous (third) step. For each of the other extraction phases, each extraction step in this list is performed again.

The extraction phases of Minute red card are the following:

- Extraction phase: original tweet. The main strategy is applied on the original tweet as well, similarly to all the other fact class extraction methods.
- Extraction phase: lines. In the first split, the original tweet is broken up into individual lines, using end-of-line characters.
- Extraction phase: sentences. After the original tweet is broken up in to lines, the tweet gets broken up into sentences by making use of the Stanford NLP software. After each step in the main strategy is carried out, a special dedicated Minute red card function is applied to each sentence. In this function, the Stanford NLP's dependency parser is used to construct a dependency graph for each sentence. If a sentence with a Minute red card is found which is still hard to parse because for example multiple player entities are in the same sentence, the dependency graph will provide us information about which player belongs to the Minute red card fact. In figure 17, three examples are provided of dependency graphs leading to an extraction of a Minute red card fact. The first two examples show two different graph structures which both will lead to an extraction of a fact. A special type of structure is shown in the third graph which shows that if a verb is found in the root of the fork of the graph, a Minute red card fact is extracted.

Minute yellow card (MYC)

The extraction of a 'Minute yellow card' is implemented in exactly the same way as the extraction of a Minute red card. The only difference in the implementation is that the keywords changed from targeting red cards to targeting yellow cards.

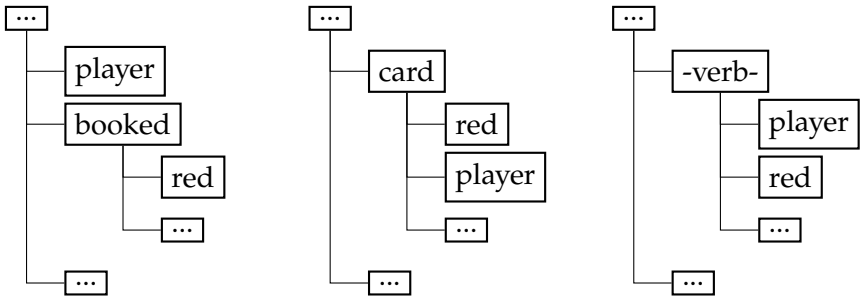

Figure 17: Example of three dependency graphs leading to an extraction of a Minute red card

Count red cards (CRC)

The extraction of 'Count red cards' is implemented by looking for the following entities; a Count, a Player and a Match. These entities will result a unique identification for each red card in every FIFA tournament, because one player can only receive one red card per game, but multiple per tournament. One of the difficulties we face when extracting the entities is that in most cases, the tweet does not contain all of the required entities, and therefore, the standard extraction method does not perform well. One of the adaptations we have made in this implementation is trying to extract a fact by making use of less entities and using knowledge from the ground truth to accomplish a unique identification. For example, a popular way of tweeting this fact class is "2nd red card of the world cup!!!". To extract a fact, we have to extract a Count, Player and Match. One of the entities, the Count, can be recognized instantly in the tweet: "2nd". The other entities however are missing. One of the extraction methods we use in other fact class extractions as well is by making use of the tweet's meta-information. By using the time the Tweet was sent and the time a match was started, we are able to extract a match which serves as the second extracted entity. By making use of these two entities, we are now able to possibly extract the last required entity, by making use of the ground truth. Because in the 2014 World Cup only one player was fined a red card in each match, if the match and count match a record in the ground truth, we can match that record as the correct extraction.

6.3 EVALUATION

To evaluate the fact extractor, we have manually annotated 578 facts in 412 tweets.

In total, the fact extractor extracted 461 facts of which 442 are correct. More performance details can be found in table 10 below.

In general, we are satisfied with the results we have achieved with the main strategies we have introduced in the fact class extractors. One of the benefits of these strategy is that we evade most of the extraction of complex relations and focus entirely on the extraction of the presented entities. Only in the last steps of the algorithms, we

Fact class	Number of tweets containing the fact class	Number of facts
Red card count (CRC)	30	30
Score final time (FT)	134	141
Score half time (HT)	98	102
Score other time (OT)	102	104
Score minute (MS)	96	117
Red card minute (MRC)	31	31
Yellow card minute (MYC)	51	53
Total	412	578

Table 9: List of manually labelled tweets

Fact class	# facts	#retrieved	#correct	Precision	Recall	F-measure
Red card count (CRC)	30	22	22	1	0.72	0.84
Score final time (FT)	142	122	118	0.97	0.83	0.89
Score half time (HT)	102	89	86	0.97	0.84	0.90
Score other time (OT)	104	71	66	0.93	0.63	0.75
Score minute (MS)	121	98	94	0.96	0.78	0.86
Red card minute (MRC)	26	25	25	1	0.96	0.98
Yellow card minute (MYC)	53	34	31	0.91	0.58	0.71
Total	578	461	442	0.96	0.76	0.85

Table 10: List of results from the fact extractor.

perform relation extraction techniques, which mostly are supported by natural language processing tools. A direct disadvantage of such a strategy is that we run into problems very quickly if the entities are difficult to extract. Because the fact classifier in chapter 5 and the fact extractor are heavily based on the detection of entities, the failure of detecting those in a tweet directly leads to a bad performance. A requirement for this strategy to work, is that the messages fit this way of extraction. But that is of course the analysis we made when designing these extraction methods.

6.4 IMPROVEMENTS AND FUTURE RESEARCH

Player extraction

A challenging and important entity extractor is the Player extractor. In section 6.2.2, we have introduced the player extractor implementation and showed how the player extractor tries to solve the problem of different spellings (often because of translations), different notations of names and how the extractor made sure the most probable player was extracted. There are still numerous different ways to write the names of players, which are not supported by the extractor. For

example the use of the initials and surname if not supported by the extractor; it will find the surname, but cannot distinguish names by making use of the initials. Another problem is the various ways of misspellings of names. This is a frequent problem and the current implementation of the player extractor does not address this problem explicitly. Of course, when only a first name is written wrong, the surname can still lead to a match, but only if the surname is mention as well. If only a surname is present and misspelled, the player extractor is unable to find the correct player. An interesting solution to this problem would be to use the web to solve this problem. A lot of popular search engines already can cope extremely well with the fact that their users misspell queries. One could for example try to find "Marcelo" by making use of the query "Marcelo". Because popular football player names queries are popular on the most popular search engines (Google, Bing) and those engines are extremely trained to deal with misspellings, those search engines are still able to lead the user to the result they desired. By making use of their API's (for example, Google Custom Search API), we can integrate those results and use them to find the player mentioned in the tweet. Another possible advantage would be that the search engines can cope with several nicknames of national teams. For example 'Oranjeleeuwen' (Orange lions) is a popular way for the Dutch to describe their athletes at international sport events. By searching for this keyword, we would easily get transferred to a Dutch fan-site with the name 'Nederlands elftal' which would give us a reference to the Dutch national team. Another way to improve the player entity extractor is to use a Named Entity Tagger, as introduced in chapter 2. One of the reasons we did not use a NER in the current implementation has two reasons. The first reason is that we are working with a Twitter dataset in which the natural language processing tools, as explained in previous sections, do not perform that well in relation to the training set they were trained on. A possible solution would be to use a combination of classifiers trained on Twitter data, for example Ritter's Twitter NLP tools integrated named entity entity extractor. Although we would improve the performance of the player extraction on misspelled names, we still had to implement a way to link the names to the correct players. The second reason for not directly implementing a named entity extractor, is that we already have a list of player names to our disposal. In a possible other scenario, where we are unsure about which names to extract, this solution will not be possible.

Another possible solution to several ambiguity problems is by using additional context in which the facts are presented. A specific example is the situation at the 29th match at the tournament, when Ghana played against the United States. As presented in the player extraction section, players are often referred to with their surname. Ghana however has a "André Ayew" and a "Jordan Ayew" in their squad and both players were selected to be part of the starting lineup. Jordan Ayew was substituted in the 59' minute. In the 82' minute, André scored a goal, which meant that a lot of tweets referred to a player named "Ayew" scoring a goal. By making use of the context,

namely that his namesake was substituted, we could determine the sought-after player directly. In a broader sense; we can use the starting lineup and substitutions to extract players.

Information extraction

One of the post promising approaches would be to use a natural language information extraction approach such as presented in "TwitIE - An Open-Source Information Extraction Pipeline for Microblog Text"[2]. In this paper, the authors present a plug-in for the General Architecture for Text Engineering (GATE) in which they added a Twitter plug-in in which several NLP tasks within the GATE processing chain are enhanced to perform better on Twitter data. These enhancements include for example Social-media specific language footprints for enhanced language detection and Twitter adapted models for POS-tagging. By making use of this enhanced pipeline, we can output a series of annotated documents which would be used by an information extractor, which should perform better in relation to an annotated document, by a standard model. The information extractor we would choose would be independent of the annotation documents we have produced in the previous step. Very often, information extractors make use of the same type of annotations which among others are produced by the TwitIE annotation pipeline. A possible subsequent step would be to use the build-in information extractor from GATE. Another possible option would be to use the information extractor module from Stanford NLP. This information extraction module from Stanford NLP is trained on particular relations such as 'Live in', 'located in', 'organisation based in' and 'work for'. The Stanford NLP software also enables users to train their own relation model by making it possible to supply your own annotated training set. An example of a training set can be found in figure 18. In 'Customizing an Information Extraction System to a New Domain'[29], parts of the function of the Stanford relation extractor are explained. Indicated is that when training data exists, the best performance in information extraction is generally obtained by supervised machine learning approaches. Important to note that research involving relation extraction is primarily focussed around binary relations. Facts which require a tertiary relation need another processing step to transform multiple binary relations to a higher order relation. Several examples of those transformations can be found in literature.

Another popular type of information extraction is open information extraction. Open information extraction systems extract relations between entities from massive corpora without requiring a pre-specified vocabulary. A popular example of an open information system is Reverb. In Reverb, an algorithm is used which uses a POS-tagged and NP-chunked sentence and returns a set of (x,r,y) triples in which x and y are noun phrases and r is a relation verb. To extract these triples, the algorithm searches for verbs by making use of the POS-tags and looks for the nearest two noun phrases around the location of the verb. My making use of various specialistic constraints on the

```
O       O       0       O       IN      In      NOFUNC  x       O
O       O       1       NP      CD      1969    NOFUNC  x       O
O       O       2       O       ,       ,       NOFUNC  x       O
Arg1    B-Peop  3       NP      NNP/NNP/NNP     James/Earl/Ray  NOFUNC  x       O
O       O       4       O       VBD     pleaded NOFUNC  x       O
O       O       5       O       JJ      guilty  NOFUNC  x       O
O       O       6       O       IN      in      NOFUNC  x       O
O       B-Loc   7       NP      NNP/,/NNP       Memphis/,/Tenn. NOFUNC  x       O
O       O       8       O       ,       ,       NOFUNC  x       O
O       O       9       O       TO      to      NOFUNC  x       O
O       O       10      NP      DT      the     NOFUNC  x       O
O       O       11      NP      NN      assassination   NOFUNC  x       O
O       O       12      O       IN      of      NOFUNC  x       O
O       O       13      NP      JJ      civil   NOFUNC  x       O
O       O       14      NP      NNS     rights  NOFUNC  x       O
O       O       15      NP      NN      leader  NOFUNC  x       O
Arg2    B-Peop  16      NP      NNP/NNP/NNP/NNP Martin/Luther/King/Junior       NOFUNC  x       O
O       O       17      O       .       .       NOFUNC  x       O
```

Figure 18: An example of a part of a training set used for relation extraction. In this file, formatted in a popular way using columns for each type of annotation, the first column indicates the two entities which hold a relation. In the second column, the NER tags, the third column the word counter and fifth column the POS-tag.

types of verbs and noun phrases, Reverb is able to achieve a very good performance. More information about Reverb can be found in 'Identifying Relations for Open Information Extraction'[10].

7 RELIABILITY CLASSIFIER

7.1 INTRODUCTION

The central purpose of this part of the system is to determine truth in tweets by making use of the sources of information provided to the system. There are different ways of implementing the reliability classifier. One possible option would be to let the classifier determine if a tweet contains any untruths. In such implementation the classifier aims to classify the tweet as false if one of the facts is false and classify the tweet as true if all the facts are true. This option is chosen in our implementation of the reliability classifier, presented in this thesis. Another implementation choice could be that the classifier only targets a specific set of facts. If one would for example only wants to target facts which someone deems to be very important to recognize (for example, detecting untrue final score facts would be important, but recognizing a wrong minute would not be important), one could train the classifier for only that purpose. Note that we only classify original tweets in the reliability classifier. We use replies and retweets to build features, but we do not determine the truth of those tweets. A diagram of the reliability classifier can be found in figure 19.

Input: The input of the reliability classifier is the text and meta-information of the tweet, the output fact classes of the fact classifier and the extracted facts from the fact extractor. The reliability classifier is also able communicate with the database to save and load attributes from other tweets.
Output: The output of the reliability classifier is false if one of the extracted facts is false or true if all extracted facts are true.

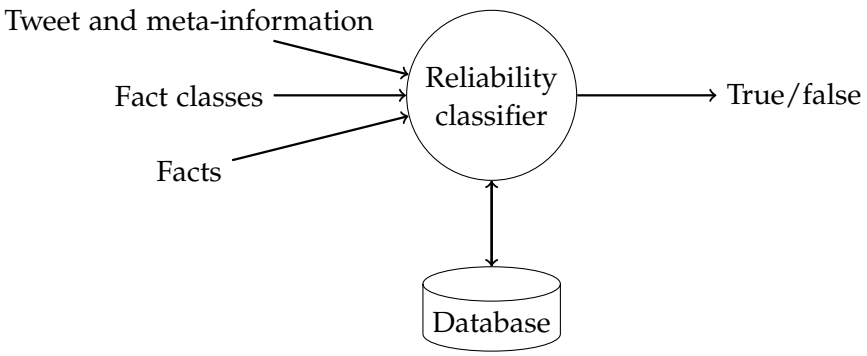

Figure 19: Diagram of the reliability classifier

7.2 IMPLEMENTATION

The reliability classifier has four sources to establish the attributes on: the tweet, which means the text of the tweet and the corresponding meta-information, the fact classes extracted by the fact classifier in chapter 5, the facts extracted by the fact extractor in chapter 6 and the database, which stores the information of every step in the process and enables the reliability classifier to combine the data.

When designing the features, we have decided to keep an open mind about which features to implement and which features not to implement. In several papers we have seen that the researchers rather chose an approach of gathering as much features as they could with the information they have gathered instead of dismissing features which are not deemed to work before implementing them. In table 11 till 16, the features the reliability classifier uses are presented and sorted in categories, which ranges from a category of features based on the Twitter message's text to features based on aggregated fact statistics throughout the dataset.

The three data types we use for the features are Integer, boolean and category. The only type that needs explanation is category which values are integers but they do not possess relationship which each other such that 4 'is bigger than' 3; they only reference to a category which can be the same or not the same. Note that not every feature is available for each tweet; for example 'country sent' is not always available because Twitter users are not obligated to declare their location in their tweets or user profile. If those attributes prove to perform well, the classifier has to use other ways to classify the tweets, which is done with observed non-missing attributes in some classifier implementations.

As the reader is shown in the tables, the total number of features identifiers is 59, but feature 56 and 59 result into 7 fact class specific features each which bring the total number of features to 71. The description of the tweets is available for every tweet, and most of the features reasoning is therefore sufficiently explained. Several of them, however, are part of a feature strategy we have laid out in this thesis as a foundation to achieve a good performance. The hypothesis of the strategy is the following. There is a relation between the popularity of a fact and the truth of a fact. True facts are claimed more often than false facts. True facts have a bigger reach, measured by the amount of users following the user posting the tweet and the follower count of users retweeting the original tweet. Twitter users with a bigger reach are inclined to pay more attention to their messages and make less mistakes. Our reasoning behind that thought is that those tweets are more important because those popular Twitter accounts are often from professionals, official institutions such as the FIFA or UEFA, or from big sites which report news. If people make mistakes or spread disinformation, people will react to those facts. The chance of people reacting to facts is a lot higher when the number of followers is higher. Therefore, it is very important to transition the attributes from one tweet to other tweets by using the knowledge that they contain

the same fact. This part of the hypothesis, the reply to tweets, can serve as a counter-balance to the first part of the hypothesis which claims that true facts are popular. If a false facts gains popularity, for example because a popular twitter user claimed the fact, the replies to that claim will counter the popularity. An important part of this hypothesis is to check what popularity means, some facts for example will automatically be more popular than other types of facts and therefore need another popularity 'score' to be true or false. This is the same with the number of replies.

#	Feature name identifier	Feature description	Data type
1	Length in characters	Length of the text of the tweet in number of characters	Integer
2	Count words	Number of words in tweet	Integer
3	Contains question mark	Tweet contains a question mark	Boolean
4	Contains exclamation mark	Tweet contains an exclamation mark	Boolean
5	Contains multi quest or excl.	Tweet contains multiple question or exclamation marks	Boolean
26	Contains emoticons	Tweet contains emotions **To implement this feature, we have listed several combinations of characters leading to emoticons, as well as a list of Unicode Emoji	Boolean
6	Contains positive emoticon	Tweet contains a “smiling” emoticon e.g. :-) ;-) as well as "positive" Unicode Emoji	Boolean
7	Contains negative emoticon	Tweet contains a “frowning” emoticon e.g. :-(;-(as well as "negative" Unicode Emoji	Boolean
8	Count uppercase letters	Fraction of capital letters in the tweet	Integer
9	Contains URL	Tweet contains an URL	Boolean
10	Number of URLs	Number of URLs contained on a tweet	Integer
11	Contains user mention	Mentions a user by using an at sign, e.g. @sneijder101010	Boolean
12	Count user mentions	Number of user mentions	Integer
13	Contains hashtag	Tweet contains a hashtag	Boolean
14	Count hashtags	Number of hashtags	Integer
15	Count spelling mistakes	Number of spelling mistakes. This feature is implemented by using Language Tool, an open source language tool available via https://languagetool.org/	Integer
27	Count sentences	Number of sentences, as given by the Stanford NLP toolset	Integer
28	Count lines	Number of lines in the tweet	Integer
29	Complexity tweet	The complexity of the tweet message, calculated by the Flesch–Kincaid test. The Flesch–Kincaid readability tests is a test to indicate how difficult a text is to understand. We use the Flesch–Kincaid implementation from GitHub user ’troywatson’ available at https://github.com/troywatson/Lawrence-Style-Checker/	Integer
38	Contains media	Tweet contains a link to a media item entity. (https://dev.twitter.com/overview/api/entities-in-twitter-objects#media)	Boolean
39	Count media	Number of media items in tweet	Integer
41	Same language user	If the tweet is written in the same language as the user said he would in its Twitter profile	Boolean
37	Count existence	The number of days the Twitter account is existing	Integer
53	Source of tweet	The source of the tweet (Twitter Web Client, Twitter for iPhone, Ubersocial for Blackberry, ...)	Category

Table 11: List of tweet text features

#	Feature name identifier	Feature description	Data type
16	Day of week user	The day of the week in which the tweet was written.	Category
17	Local time user tweet sent	The local time the tweet was sent by the user, rounded down to the nearest whole hours	Category
18	Country sent	The country in which the tweet was sent. Each Twitter user can set a location to a tweet and can set a location to its Twitter account. First we try to extract the location set to the tweet, if not available, we try to extract the location of the Twitter account.	Category

Table 12: List of tweet time and place features

#	Feature name identifier	Feature description	Data type
19	Sentiment positive words	The number of positive words in the text, using the Stanford NLP software.	Integer
20	Sentiment negative words	The number of negative words in the text, using the Stanford NLP software.	Integer
22	Sentiment score NLP	The sentiment score of the whole tweet, using the Stanford NLP software.	Integer

Table 13: List of sentiment features

#	Feature name identifier	Feature description	Data type
23	Count retweets	The number of retweets	Integer
24	Count replies	The number of replies on the tweet	Integer
25	Count replies positive sentiment	The number of replies with a positive sentiment	Integer
21	Count replies negative sentiment	The number of replies with a negative sentiment	Integer
30	Followers reach	The number of Twitter users seeing this tweet. That is: the number of followers the Twitter user has, plus the number followers each user which retweets this message has.	Integer

Table 14: List of retweets and replies features

To try this hypothesis, we try to find and implement features which relate as much as possible to the hypothesis. Notice that we only have to implement individual parts of the hypothesis; the combination of the features which will possibly result in a good performance will be organised by the classifier itself.

To explain the features regarding facts, we make use of a custom designed fact and tweet notation. A tweet is denoted by using the letter T, and different kinds of tweets by using a subscript identifier;

#	Feature name identifier	Feature description	Data type
31	Count tweets user	The number of tweets a user has sent	Integer
32	Count followers user	The number of followers the user has	Integer
33	Count following user	The number of users the user is following	Integer
34	User verified	If the account of the user has been verified	Boolean
35	Count favourites	The number of favourite tweets the user has	Integer
36	Count listed	The number of listed tweets the user has	Integer
51	Count tweets user dataset	Number of original tweets in the dataset from the user	Integer
52	Count retweets user dataset	Number of retweets tweets in dataset from the user	Integer
54	User has URL	If the description of the Twitter user contains a URL	Boolean

Table 15: List of user features

for example T_1, T_2, T_3 are different tweets, but could still contain the same facts. Fact are denoted by the letter F and, in the same way tweets are, different facts by using subscript: F_1, F_2, F_3. A tweet contains one or more facts, which is denoted by $T_1 = F_1 + F_2$. If the fact class of a fact needs to be denoted, we use superscript. For example, a fact identified by 1 with the fact class 2 is denoted by F_1^2. A reaction to a tweet is denoted by T_R. A reaction to a tweet T_2 is denoted by $T_{R@2}$. If a reply contains a reaction on a fact with a fact comparison entity 'score', it is denoted by $T_{R@2} = R_{score}$

One of the pillars of the strategy is finding the reach of a tweet and the facts in a tweet. We target these types of attributes in the following features:

- Feature 23 'Count tweets' and feature 24 'count retweets'. These features only target the tweet itself.
- Feature 42, 'Count tweets least popular facts'. This feature counts the number of times the facts in the tweet appear in other tweets. The feature returns the number of times the least popular fact appears in other tweets. For example if the dataset consists of the tweets $T_1 = F_1 + F_2$ and $T_2 = F_1$, this feature for tweet T_1 returns 1.
- Feature 55 'Reach least popular fact'. This feature counts the number of times Twitter users could see this fact by calculating the reach of a fact. If a tweet contains a fact, its reach is calculated by the number of followers of the author of the tweet plus the number of followers of all the Twitter users which have retweeted the original tweet.
- Feature 56, 'Count tweets least popular fact category x', is a collection of features with the same name. The last number, denoted in the feature name identifier with an x, is referring to the fact class. Every fact class we have implemented in the system has a corresponding feature which only target the facts which

belong to that class. Every feature, like feature 42, counts the number of tweets that contain the fact, and return the number of times the least popular fact in that fact class appears in the dataset. For example, if a dataset would consists of $T_1 = F_1^2 + F_2^2$, $T_2 = F_3^2 + F_2^2$, feature 'Count tweets least popular fact category 2' would return 1. One of the reasons behind this feature is that the fact classes differ a lot in popularity. Some fact classes are mentioned very often, while some are only mentioned a couple of times throughout the dataset. Now say that a particular fact class, for example 5, is mentioned 500 times on average and fact class 2 has an average of 50. Then, a tweet which would contain a fact from fact class 5 which is false and has therefore has a small count of tweets containing that fact, for example 60, still would not get 'noticed' by feature 42. By implementing this feature, we take the popularity of each class fact into account and we would notice that the popularity of the false fact is below average.

Another pillar of the strategy is finding a reaction and finding the number of reactions on a tweet or fact. We target these types of attributes in the following features:

- Feature 47 'Has reply regarding facts' and feature 48 'Count replies regarding facts'. With these two features, we try to find a way to link comments on a tweet to a fact, by checking if a fact contains a fact comparison entity which is the same as the facts in the tweet. For example if the original tweet contains a Score final time, the fact comparison entity is a score entity.

- Feature 49, 'Has reply facts tweet dataset' and feature 50 'Count replies facts tweet dataset'. With these two features, we build upon the idea of feature 47 and 48, but increase the search scope to the whole dataset. We implement this feature by using each reply to each tweet containing a fact which is part of the original tweet.

- Feature 58 'Has reply facts one fact group tweet dataset' and 59 'Count replies facts on fact group tweet dataset' are two features which build upon the idea from feature 49 and 50, but refine both features in a way we will show by making use of an example. Say that the dataset consists of $T_1 = F_1^1 + F_2^2$, $T_2 = F_1^1$, $T_3 = F_1^1 + F_3^1$, $T_{R@2} = R_{score}$, $T_{R@3} = R_{score}$. Notice that tweet 3 has two facts with the same fact class. So if feature 49 and 50 calculate the number of replies on fact F_1^1, the reply $T_{R@3}$ would be included because T_3 contains the fact F_1^1. However, although the reply has the fact comparison entity 'score', this reply could also be a reaction on the fact F_3^1. Therefore, feature 58 and 59 do not include the reaction on T_3. The reaction on T_2 is included by these two features, because T_2 does not contain a fact with the same fact comparison type.

- Feature 60 'Highest individual count replies facts one fact group tweet dataset'. This feature is implemented in the same way as

feature 59 is, but only returns the number of replies on one fact in the tweet, namely the fact which has the most replies.

#	Feature name identifier	Feature description	Data type
42	Count tweets least popular fact	Returns the number of times the least popular fact in the tweet is mentioned all other tweets in the dataset.	Integer
43	Count facts	Number of facts (extracted by the Fact Extractor) in the tweet	Integer
44	Count minutes	Number of minutes in the tweet	Integer
45	Count scores	Number of score expressions in the tweet	Integer
46	Count count indicators	Number of count indicators in the tweet	Integer
47	Has reply regarding facts	Returns if a tweet has a reaction regarding a fact. We have implemented this by checking if a reply contains a fact comparison entity which is the same as the facts in the tweet.	Boolean
48	Count replies regarding facts	Number of reactions regarding a fact, implemented the same way as feature 48 'Has reply regarding facts'	Integer
49	Has reply facts tweet dataset	Returns true if the dataset has a reaction on a fact also present in this tweet	Integer
50	Count replies facts tweet dataset	The number of reactions to on facts in the tweet in the dataset	Boolean
55	Reach least popular fact	The reach (number of followers which see the fact) of a specific fact	Integer
56	Count tweets least popular fact category x	The number of tweets of the least popular fact in each type of fact in the tweet	Integer
57	Reach least popular fact category x	The reach of the least impactful fact for each type of fact in the tweet	Integer
58	Has reply facts one fact group tweet dataset	Checks if there is a reply in the dataset regarding a fact in the tweet. This feature regards a fact reaction 'valid' if the tweet on which the reaction is based contains only one fact from the fact comparison type on which the fact reaction is based on.	Boolean
59	Count replies facts on fact group tweet dataset	Counts the number of reactions, implemented the same way as feature 58 'Has reply facts one fact group tweet dataset'	Integer
60	Highest individual count replies facts one fact group tweet dataset	This feature is implemented in the same way as feature 59 is, but only returns the number of replies on one fact, instead of the summation of every fact in the tweet.	Integer

Table 16: List of fact features

7.3 EVALUATION

In contrast to the evaluation of the other system parts, we cannot create a manual test set for the reliability classifier. A lot of features,

Class	Precision	Recall	F-measure
Class A: tweets with 0 false facts	0.983	0.993	0.988
Class B: tweet with >1 false facts	0.923	0.818	0.867

Table 17: Performance results of the reliability classifier

Class	Classified as A	Classified as B
Class A: tweets with 0 false facts	15590	102
Class B: tweet with >1 false facts	274	1228

Table 18: Reliability classifier's confusion matrix

most of them in table 11, are based on only one tweet, but most of the features which are part of the strategy we have set out in the reliability classifier are based of groups of tweets. To evaluate these features, we would need to extract a lot of facts by hand. To test the reliability classifier, we make use of the results we have achieved with the fact classifier and fact extractor. By making use of the dataset in table 1, we have created a dataset containing 17194 tweets which contain 21367 facts. By making use of these tweets, we collect every Retweet and Reply on these original tweets in the University's dataset. By making use of the ground truth, we can now determine the truth of each fact and classify which tweets contain untruths and which tweets contain only true facts.

Similar to the previous evaluations, we have again tried out several classifiers to maximize the performance of the features. Again the J48 classifier performed the best on our dataset. By using a combination of wrapper based Correlation Feature Selection (CFS) feature selection and the J48 classifier, we have received an F1-score of 0.988 for class A; the tweets which contained no false facts and an F1-score of 0.818 on class B; the tweets which contained one or more false facts. The set of features which combine to the best performance can be found in table 19.

Feature identifier	**Feature name**
42	Count tweets least popular fact
44	Count minutes
45	Count scores
50	Count replies facts tweet dataset
55	Reach least popular fact
56-8	Count tweets least popular fact category 8
57-8	Reach least popular fact category 8

Table 19: Set of features which resulted in the best performance by the reliability classifier

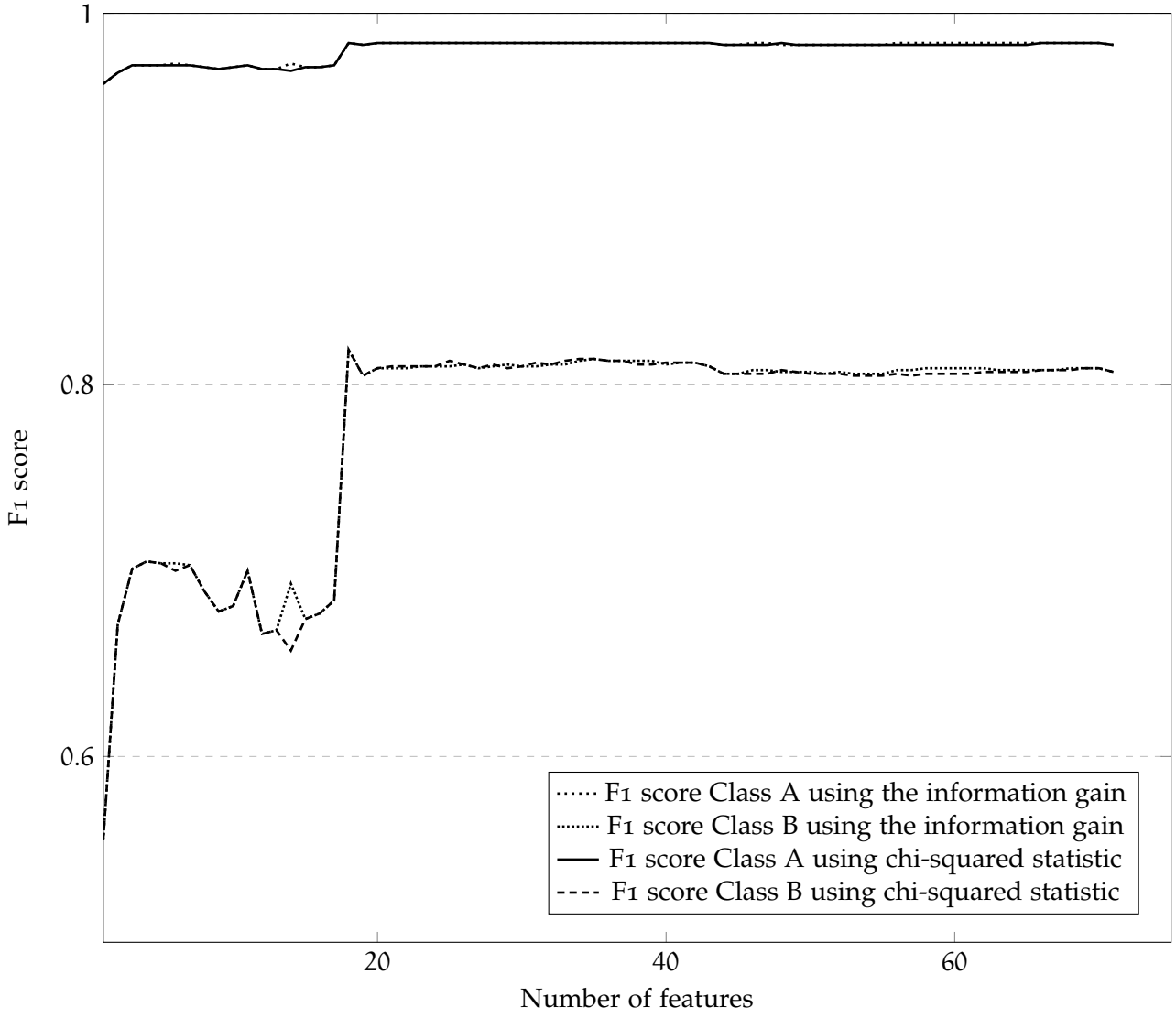

Figure 20: The performance of the classifier using two filter feature selection methods; information gain and the chi-squared statistic. On the x-axis, the number of features available for the classifier is shown, where the features are sorted and added to the feature set based on the feature score of the feature selection method.

Rank	Feature based on the Chi-squared Statistic	Feature based on the Information Gain
1	42 Count tweets least popular fact	42 Count tweets least popular fact
2	55 Reach least popular fact	55 Reach least popular fact
3	50 Count replies facts tweet dataset	50 Count replies facts tweet dataset
4	60 Highest individual count replies facts one fact group tweet dataset	59 Count replies facts on fact group tweet dataset
5	59 Count replies facts on fact group tweet dataset	60 Highest individual count replies facts one fact group tweet dataset
6	49 Has reply facts tweet dataset	58 Has reply facts one fact group tweet dataset
7	58 Has reply facts one fact group tweet dataset	49 Has reply facts tweet dataset
8	57-5 Reach least popular fact category 5	57-5 Reach least popular fact category 5
9	56-6 Count tweets least popular fact category 5	56-5 Count tweets least popular fact category 5
10	43 Count facts	43 Count facts
11	57-8 Reach least popular fact category 8	57-8 Reach least popular fact category 8
12	56-6 Count tweets least popular fact category 6	56-6 Count tweets least popular fact category 6
13	57-6 Reach least popular fact category 6	57-6 Reach least popular fact category 6
14	53 Source of tweet	56-7 Count tweets least popular fact category 7
15	56-7 Count tweets least popular fact category 7	53 Source of tweet
16	56-8 Count tweets least popular fact category 8	57-7 Reach least popular fact category 7
17	57-7 Reach least popular fact category 7	56-8 Count tweets least popular fact category 8
18	44 Count minutes	44 Count minutes
19	18 Country sent	18 Country sent
20	45 Count scores	45 Count scores

Table 20: Top 20 best performing features determined by the Chi-squared Statistic and Information Gain.
The combined performance can be viewed in figure 20.

A very powerful tool in our feature design is the way of connecting tweets by making use of the facts and combine the information acquired in several places in the dataset and apply it as attributes for every tweet. This strategy has worked well and is shown in the resulting set in table 19. One of the regrettable observations we can make is that the features we have hoped for, 'Highest individual count replies facts one fact group tweet dataset' (60) and Count replies facts on fact group tweet dataset (59), are not part of the result set. Although these features are not part of the final feature set, they do score high on the feature selection methods with a 4th and 5th place on both measures. One of the possible reasons of the absence of the features in the final set is that the implementation of the features is not refined enough. One of the surprising and interesting features in the set is the 'count minutes' and 'count scores'. Both features do not score high individually but are performing very well in combination with other features. A possible reason for that is that 'count scores' and 'count minutes' are really good indicators for which fact class are present in a tweet. By making use of that, the classifier can make a link between the count of the facts and the reach of the facts, just like we tried to in feature 56 and 57. We think that a decision tree model, the concluding model we use as reliability classifier, is very applicable in this situation.

An interesting discussion is how we should interpret the two F1-scores of the two classified classes. There are several valid options, but the most important condition is the fact that the choice has to be applicable for the goal of the application. For example, if one would use the classifier to search for false facts in a dataset, the F1-score of class B is leading. On the contrary, for an application which would filter out false facts in order to obtain true facts, the F1-score would be leading. It is also possible to assign a weight to the F1-scores (with weight w ranging from 0-1 resulting in $F1 = w * F1a + (1 - w) * F1b$) which would fit the use case of the application.

There are several ways to rank the performance of each feature. Two popular ways to rank features are two filter methods; the information gain and the chi-squared statistic. Both methods are filter methods which determine the performance of a single feature in respect to the class. The resulting list, the features sorted on the score of both methods, is used to plot the graph in figure 20 which shows the F1 score of the classification using n number of features starting from the best features according to the two feature selection methods. As is shown in the graph, both feature selection methods are unable to detect the best combination of features, which is quite obvious because although each individual features do not have to result in a good performance, a combination of features could. An important remark to make is that we 'cheated' a little bit when we generated the graph is figure 20, because in a normal setting, the test set, which would result in the F1-scores presented in the figure, would not be touched in the evaluation. However, the attribute selection tool already 'saw' the set because it used that set to determine the performance of each feature by calculating the information gain and the chi-squared statis-

tic. The graph is only used to give the reader an indication about the performance development of the classifier throughout the addition of features, and is not used in the evaluation of the performance of the classifier.

7.4 IMPROVEMENTS AND FUTURE RESEARCH

Evaluation per fact class

An important part of the evaluation is the evaluation of different types of facts. Our hypothesis is that the reliability classifier performs differently on different kind of facts because Twitter users behave differently around different kind of facts. People are more inclined to react to facts which they regard as very wrong. For example when they see that a 'Minute scored' fact is off one minute, they are less likely to react on that instead of a score being flipped ('4-0' instead of '0-4'). They see the 'Minute scored' fact as 'true enough' or are not aware that the minute is one off. We think the performance of these types of facts is lower, but we still need to validate our hypothesis by another evaluation.

Reply features

By making use of the replies in the whole tweet set and using the powerful tool of combining the facts, we can also classify unpopular tweets, which do not have reactions or replies. At this moment, we use a rather primitive method to find a reply on a fact, by making use of the fact comparison type. A very powerful improvement would be to design a way to improve the extraction of a reply to a fact:

- We could determine if a reply says something positive or negative about the statement of the fact. For example, the reply could state that a statement made in the original tweet is positive and true or a reply could underline that a statement is false.
- We could improve the link between a reply and a fact. Most tweets contain multiple facts and by improving the reply extraction, we could more accurately pinpoint to which fact a reply is replying. By making use of the improvement, we could improve the the features in the category fact features which now struggle to see to which fact a reply is pointing to (see explanation of feature 58 and 59). Therefore, we throw away a lot of available data which could be used in this possible improved scenario.

Propagation of facts

A very interesting way of looking at facts is how they propagate through Twitter. What are the first sources of the fact, how quick is the fact picked up by others, are there multiple sources and who are the sources who mention the fact? In this way, we could build a model which could serve as a foundation to compare facts with each other. A somewhat comparable example is the detection of facts using tweet

volume, presented in 'Using Twitter to Detect and Tag Important Events in Live Sports' by James Lanagan and Alan F. Smeaton.[19] In this paper, they present a technique which automatically detects events using the detection of an increase of (or spike in) tweet volume for specific keywords relating to events. In this way, the paper showed they could detect and extract sport related events very accurately, but indicated that this technique only worked well for very specific types of facts.

Context of facts

One of the most important concepts of understanding a claim is to understand the context of the claim. We have shown in the fact extractor that by making use the time context, we have been able to extract a the match which belongs to the facts mentioned in the tweet. Very interesting features would use the context of other facts to determine the possibility of other claims. In this way, we could outweigh contradicting claims against each other in order to retrieve the correct facts. An example of such an approach is used in "Sport-related fact extraction in social media"[16], the author's earlier study to offer background information for this thesis. The approach in this project was to use the popular vote as the determining factor for truth. By using this approach, facts which contradicted the most popular vote were discarded and another extraction based on the popular vote was performed. A related example could be that by using the certainty of truthfulness of a claim, another fact can be extracted as well. For example, if the half-time score of a match is certain and is set to be 2-0, the only possible other score in the first half of the match would be 1-0. By using this simple way of creating context around a fact, by using the certainty of truthfulness in one place in combination with knowledge about rules of the game, the performance of an extraction system or truth determination system can be enhanced. This idea of conflicting claims can be used in a broad sense and in more complex situations as well; it is not limited to this situation we showed in this thesis.

8

DISCUSSION & FUTURE RESEARCH

Performance of the fact extractor

The performance of the reliability classifier is based on the other systems modules like the fact classifier and fact extractor. Improving the precision and recall of those systems will improve the reliability classifier. The improvement of the recall is heavily related to the performance of the entity extractors. In chapter 6, we have summed up various ways to improving the extraction of entities, which could improve the recall a lot. Interesting techniques to improve the extraction of facts are the machine learning information extraction techniques.

Performance of the features in other datasets

In this thesis, we have shown that we can achieve a good performance on the dataset and fact classes we have introduced in this thesis. A very interesting, and maybe the most important question after this conclusion is how these features and performance would relate to other datasets and other types of facts. Our hypothesis is that our features work well on datasets similar to the dataset. Our features aim for datasets were facts are repeated and originate from different (independent) sources. In this way, the false facts are countered by a lot of independent other 'correct' sources. Because there are many sources, and the facts can be verified by many sources, the false facts can be countered by replies. Criticising falsities can be universal, but we think there is a lot of difference in when people react to fact and when they do not. For example, in this dataset we have seen that reactions on false scores are a lot more common than reactions on incorrect minutes. A reason for that could be that people see this errors to be too insignificant to react on. Another important observation is that people are more likely to react to authoritative and popular twitter users. For example, a lot of unknown users could spread false facts without getting a reaction from their small group of followers. In contrast to the popular and authoritative users which, in the eyes of their follower base, should be right. If they are incorrect, they have a lot of users which potentially could react to an error. Another important property of the facts is their verifiability. If facts are easily verifiable, a lot more people can discuss and react to the fact, which increases the engagement of users from independent sources. If a false fact is claimed by one user, other independent sources are not likely to spread this because they have the tools to claim otherwise. In contrary, facts which are only verifiable by a few users cannot be rectified by other users because they do not have the sources to do so. This scenario with verifiable facts and independent users is important for our features to perform well. As claimed, a scenario in which

facts are hard to verify or are still to know to have the engagement of multiple users, the features will likely perform worse.

Performance of the features in a real-time situation

A very interesting scenario is how this prototype, if minimally altered, would perform in a live situation. Because the features of the reliability classifier focus on the whole dataset, an index needs to be build for the data sources of the features. Not every tweets has to be saved fully; only small parts, like the links between a reply and a tweet or which facts are contained by which tweet or a list of tweet identifiers for each fact. This index would significantly reduce the amount of data which needs to be saved. It will also increase the performance of the calculations. A very important change in the system will be when the reliability classifier can determine if a fact is true or false. The features which determine this are based on the popularity of the tweet and the fact of people replying to tweets. The number of tweets containing a fact and the number of replies a tweets gets is correlated to the event; for example a FIFA World Cup would get more Twitter volume than a sport tournament on the University of Twente. Furthermore, the Twitter volume is related to the type of fact; there are preferences to which facts and errors people reply and what type of fact they tweet about. In an initial situation, for example when a tweet enters the system which contains a still unknown fact, the reliability classifier often is unable to determine the truth of that statement in this situation. The features in this thesis are based on a period of time: the amount of time the dataset is based on. As we have mentioned in the thesis, most tweets about events are tweeted in the same period, very close to moment the event was happening. Retrospective tweets, a couple of hours after an event, are very popular as well. So the features which are used in this thesis need development time. A certain amount of time or a certain amount of Twitter volume needs to have passed the system to give the reliability classifier time to 'make up its mind'. In this period, the output of the classifier could for example be different kinds of statuses like 'waiting' or 'in progress'. The classifier does not have to make up its mind directly, it could also give a probability to the current classification it can make. It could happen of course that 2 out of 3 facts can be validated, but that the classifier has to wait for 1 other fact to determine the classification for the whole tweet. This could also influence the probability of a classification 'still in progress'. It could also happen that the classification of a tweet changes, because enough tweets have passed the system to decide otherwise. The follow up question then would be how long it would take the classifier to reach a valid classification. This is not an easy question to answer because this very related to the types of facts in the tweet and the event to which those claims belong. In an international event, facts are mentioned a lot more often, and therefore, the classifier needs to be trained differently for small national events. Ways to calculate which 'type' of classifier has to be used can be done using by calculating the number of hashtags or keywords mentioning

an event or by calculating the number of languages involved in the event or from how many countries Twitter users are mentioning the event. By making use of such a model, an estimate can be calculated how many tweets are expected and which model the reliability classifier should use. When deploying the system, the developer could also indicate to the system how many users or tweets it expects to make the reliability classifier perform better in an earlier stage. Note that if we could design features which would use the context of the other claims (see the section 'context of facts' in the previous chapter), we could design a reliability classifier which could use the outcome of one fact to classify another. If we would have classified a score of '1-0' to be true, we could classify an unseen fact '0-1' to be false because those two claims can not coexists together.

9 CONCLUSION

Social media is very popular. Besides the fact that you can follow the lives of your friends, social media today has become a very important tool for companies, associations and media. Although the credibility of social media is still low, partly because of the open, uncontrolled, uncurated nature of social media, different types of research have shown that social media can serve as an important information source for different applications. Furthermore, social media has proven itself as an important source of news; social media is known for its ability to spread news very rapidly. This can serve as an important aid, but also as a source of false rumours, which can be an instigator in a time of crisis and turmoil. From these several perspectives, research on veracity in social media extremely important. By making use of this research, systems can be designed which can serve as a tool to filter out misinformation in times of crisis, or serve as filter applications for systems who make use of social media messages as a source of information. Research relating to this is still very scarce, but recent research done such as ClaimFinder[4] and the Pheme project, shown in the second chapter in this thesis, show the increasing interest in this field.

In this thesis, we have shown a system consisting of four parts which are trained specifically on a dataset consisting of tweets about the 2014 FIFA World Cup. The first part of the system consists of a filter which deletes undesirable tweets (which for example interfere in parts of the system with wanted tweets). The second part of the system consisting of a fact classifier, which is able to recognize which types of facts, from the set of facts we are targeting, are contained in the tweet. The third part of the system is the fact extractor, which is able, with help of the output of the fact classifier, to extract the facts in the tweet. The fourth and final part of the system is the reliability classifier; a feature based classifier which can determine if a tweet contains a false fact. Generally, we are very satisfied with the performance of the systems and system components presented in this thesis. The fact classifier scores an F1-score of 0.96, the fact extractor an F1-score of 0.85 and the reliability classifier an F1-score on class A, tweets with zero false facts, of 0.988 and an F1-score on class B, tweet with 1 or more false facts, of 0.867. As shown in various parts of this thesis, there is much room for improvement, especially an improved entity extraction can give the recall of several systems a big performance increase. In this thesis we have already mentioned several options that can ensure a performance boost.

An important remark is that we have focussed our attention fully on the dataset containing tweets about the FIFA World Cup. It would be very interesting and important to see how this system performs in other scenarios. We expect that there are enough scenarios were it,

with minimal adjustments, also performs very well. We have talked about those scenarios in the discussion chapter, were we have indicated that we expect our features to perform well in scenarios where facts are repeated and originate from different (independent) sources. We of course also envision a lot of scenarios where the mentioned conditions do not hold and therefore, the reliability classifier is expected to perform worse, which leaves plenty of room for future research.

In order to answer the research questions of the thesis, we first will start answering the three sub-questions.

1. What architecture can we compose and implement to test and train features and classifiers to check if facts presented in tweets are true or false?
This research question we have answered by building the system architecture presented in chapter 3. We are satisfied with the choice to divide the functions of identifying the types of facts and extracting them, since the step of identifying the types of facts has proved to be less difficult than extracting them and helps us in several stages in the system. The database enables us to save individual features and effectively combine them into other (higher order) features, which makes training more efficient. Each part in the system can individually be trained and tested because of the modularity.

2. How effective can we extract facts from tweets?
This research question has been answered in chapter 6. In this chapter, we have shown an achievement of satisfying result; the ability of extracting 7 different types of fact classes which performed with an average F1-score of 0.85. By analysing the Twitter messages in this specific dataset, using sport and tournament specific knowledge and using several natural language programming techniques, we are able to effectively extract different types of facts.

3. What kind of features can we design and implement which determine truth in tweets?
In chapter 7, we have tested 87 features which resulted in a result set of 7 features which in combination resulted in the best performance. As we have showed in chapter 7, a combination of features which counted the number of tweets which contained a specific fact, the number of replies to a tweet and the number of users the facts could have reached (our own implementation of engagement) in addition to an indication of the fact classes which are present in the tweet.

Can we automatically determine truth in tweets using feature based supervised statistical classifiers?
By using popularity and reach based features in combination with fact type indicating features and indicators for the number of replies to a fact, we are able to determine truth in a very specific dataset. By using a combination of wrapper based Correlation Feature Selection and the J48 classifier, we have achieved an F1-score of 0.988 for class

A; the tweets which contained no false facts and an F1-score of 0.818 on class B; the tweets which contained one or more false facts. Our discoveries look promising and we expect that there are several situations in which the reliability classifier will perform very well. In the discussion section in chapter 7, we have listed the requirements for a good performance of the classifier, as well as indicated when we expect a worse performance and the corresponding enhancements we propose for every phase of the system.

Part I

APPENDIX

A

TWEET AND CORRESPONDING META-INFORMATION

Below we have listed the JSON representation of the tweet and meta-information of a tweet in the dataset.


```
{
   "filter_level":"medium",
   "contributors":null,
   "text":"RUS 1-2 KOR #WorldCup",
   "geo":null,
   "retweeted":false,
   "in_reply_to_screen_name":null,
   "possibly_sensitive":true,
   "truncated":false,
   "lang":"en",
   "entities":{ },
   "in_reply_to_status_id_str":null,
   "id":479021573169754113,
   "source":"<a href="https://mobile.twitter.com" rel="nofollow">
       Mobile Web (M2)</a>",
   "in_reply_to_user_id_str":null,
   "favorited":false,
   "in_reply_to_status_id":null,
   "retweet_count":0,
   "created_at":"Tue Jun 17 22:03:37 +0000 2014",
   "in_reply_to_user_id":null,
   "favorite_count":0,
   "id_str":"479021573169754113",
   "place":null,
   "user":{
      "location":"Prishtine,KOSOVA",
      "default_profile":false,
      "statuses_count":13585,
      "profile_background_tile":false,
      "lang":"en",
      "profile_link_color":"821010",
      "profile_banner_url":"https://pbs.twimg.com/profile_banners
          /460110010/1397846627",
      "id":460110010,
      "following":null,
      "favourites_count":991,
      "protected":false,
      "profile_text_color":"333333",
      "verified":false,
      "description":"Little things...",
      "contributors_enabled":false,
      "profile_sidebar_border_color":"FFFFFF",
      "name":"Netah",
      "profile_background_color":"709397",
      "created_at":"Tue Jan 10 11:53:54 +0000 2012",
```

```
      "default_profile_image":false,
      "followers_count":499,
      "profile_image_url_https":"https://pbs.twimg.com/
          profile_images/476782495979933696/B5sFTGzN_normal.jpeg",
      "geo_enabled":true,
      "profile_background_image_url":"http://pbs.twimg.com/
          profile_background_images/447040412750868480/qti7S6nn.
          jpeg",
      "profile_background_image_url_https":"https://pbs.twimg.com
          /profile_background_images/447040412750868480/qti7S6nn.
          jpeg",
      "follow_request_sent":null,
      "url":null,
      "utc_offset":7200,
      "time_zone":"Amsterdam",
      "notifications":null,
      "profile_use_background_image":true,
      "friends_count":89,
      "profile_sidebar_fill_color":"A0C5C7",
      "screen_name":"albioneta",
      "id_str":"460110010",
      "profile_image_url":"http://pbs.twimg.com/profile_images
          /476782495979933696/B5sFTGzN_normal.jpeg",
      "listed_count":0,
      "is_translator":false
   },
   "coordinates":null
}
```

BIBLIOGRAPHY

[1] InSites Consulting Anke Moerdyck. *What is the impact of financial news reporting in social media?* Nov. 2012. URL: http://www.insites-consulting.com/what-is-the-impact-of-financial-news-reporting-in-social-media/ (visited on 04/10/2016).

[2] Kalina Bontcheva, Leon Derczynski, Adam Funk, Mark A Greenwood, Diana Maynard, and Niraj Aswani. "TwitIE: An Open-Source Information Extraction Pipeline for Microblog Text." In: *RANLP*. 2013, pp. 83–90.

[3] Marketingfacts Bram Koster. *Journalisten: social media niet betrouwbaar, wel belangrijk #SMING14*. June 2014. URL: http://www.marketingfacts.nl/berichten/journalisten-social-media-niet-betrouwbaar-wel-belangrijk-sming14 (visited on 04/10/2016).

[4] Amparo E Cano, Daniel Preotiuc-Pietro, Danica Radovanović, Katrin Weller, and Aba-Sah Dadzie. "# Microposts2016: 6th Workshop on Making Sense of Microposts: Big things come in small packages." In: *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee. 2016, pp. 1041–1042.

[5] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. "Information credibility on twitter." In: *Proceedings of the 20th international conference on World wide web*. ACM. 2011, pp. 675–684.

[6] Thomas Chesney. "An empirical examination of Wikipedia's credibility." In: *First Monday* 11.11 (2006).

[7] KL Cheung. "Predicting Trust in Wikipedia Articles." In: (2011).

[8] Smart Insights Dave Chaffey. *Global social media research summary 2016*. Apr. 2016. URL: http://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/ (visited on 04/10/2016).

[9] Pierpaolo Dondio, Stephen Barrett, Stefan Weber, and Jean Seigneur. "Extracting trust from domain analysis: A case study on the wikipedia project." In: *Autonomic and Trusted Computing* (2006), pp. 362–373.

[10] Anthony Fader, Stephen Soderland, and Oren Etzioni. "Identifying relations for open information extraction." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2011, pp. 1535–1545.

[11] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. "Incorporating non-local information into information extraction systems by gibbs sampling." In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics. 2005, pp. 363–370.

[12] Vindu Goel and Brian Stelter. *Social Networks in a Battle for the Second Screen*. Oct. 2013. URL: http://www.nytimes.com/2013/10/03/technology/social-networks-in-a-battle-for-the-second-screen.html.

[13] Jeffrey T Hancock. "Digital deception." In: *Oxford handbook of internet psychology* (2007), pp. 289–301.

[14] Philip N Howard, Aiden Duffy, Deen Freelon, Muzammil M Hussain, Will Mari, and Marwa Mazaid. "Opening closed regimes: what was the role of social media during the Arab Spring?" In: *Available at SSRN 2595096* (2011).

[15] Nielsen Holdings Inc. *State of the media - The social medai report 2012*. 2012. URL: http://www.nielsen.com/us/en/insights/reports/2012/state-of-the-media-the-social-media-report-2012.html (visited on 04/10/2016).

[16] Bas Janssen. "Sport-related fact extraction in social media." In: (2016).

[17] Reuters Institute for the Study of Journalism. *Reuters Institute Digital News Report 2013*. 2013. URL: http://reutersinstitute.politics.ox.ac.uk/publication/digital-news-report-2013 (visited on 04/10/2016).

[18] Minjeong Kang. "Measuring social media credibility: A study on a Measure of Blog Credibility." In: *Institute for Public Relations* (2010), pp. 59–68.

[19] James Lanagan and Alan F Smeaton. "Using twitter to detect and tag important events in live sports." In: *Artificial Intelligence* (2011), pp. 542–545.

[20] Teun Lucassen and Jan Maarten Schraagen. "Evaluating WikiTrust: A trust support tool for Wikipedia." In: *First Monday* 16.5 (2011).

[21] Gil Luria and Sara Rosenblum. "Comparing the handwriting behaviours of true and false writing with computerized handwriting measures." In: *Applied Cognitive Psychology* 24.8 (2010), pp. 1115–1128.

[22] Laurentiu Iancu Mark Davis. *Unicode Standard Annex 29 - Unicode text segmentation*. May 2016. URL: http://unicode.org/reports/tr29/ (visited on 05/06/2016).

[23] Deborah L McGuinness, Honglei Zeng, Paulo Pinheiro Da Silva, Li Ding, Dhyanesh Narayanan, and Mayukh Bhaowal. "Investigations into Trust for Collaborative Information Repositories: A Wikipedia Case Study." In: *MTW* 190 (2006).

[24] Stephen Porter and Leanne Brinke. "The truth about lies: What works in detecting high-stakes deception?" In: *Legal and Criminological Psychology* 15.1 (2010), pp. 57–75.

[25] Consumer Reports. *Facebook & your privacy - Who sees the data you share on the biggest social network?* June 2012. URL: http://www.consumerreports.org/cro/magazine/2012/06/facebook-your-privacy/index.htm (visited on 04/09/2016).

[26] Alan L Ritter. "Extracting Knowledge from Twitter and The Web." In: (2013).

[27] 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*. ACM. 2010, pp. 851–860.

[28] 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*. ACM. 2009, pp. 42–51.

[29] Mihai Surdeanu, David McClosky, Mason R Smith, Andrey Gusev, and Christopher D Manning. "Customizing an information extraction system to a new domain." In: *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*. Association for Computational Linguistics. 2011, pp. 2–10.

[30] Charles Sutton and Andrew McCallum. "An introduction to conditional random fields." In: *Machine Learning* 4.4 (2011), pp. 267–373.

[31] Twitter. *Insights into the WorldCup conversation on Twitter*. July 2015. URL: https://blog.twitter.com/2014/insights-into-the-worldcup-conversation-on-twitter.

[32] Twitter. *TVxTwitter*. July 2015. URL: https://biz.twitter.com/twitter-tv.

[33] Guido Van Oorschot, Marieke Van Erp, and Chris Dijkshoorn. "Automatic extraction of soccer game events from twitter." In: *Proceedings of the Workhop on Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE 2012)* 902 (2012), pp. 21–30.

[34] Wikipedia.org. *Reliability of Wikipedia - Comparative studies*. Apr. 2016. URL: https://en.wikipedia.org/wiki/Reliability_of_Wikipedia\#Comparative_studies (visited on 04/10/2016).

[35] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. "Top 10 algorithms in data mining." In: *Knowledge and information systems* 14.1 (2008), pp. 1–37.