NTNU
Norwegian University of
Science and Technology

Automatic Sarcasm Detection in Twitter Messages

Johan Georg Cyrus Mazaher
Ræder

Master of Science in Computer Science
Submission date: June 2016
Supervisor: Bjørn Gamback, IDI

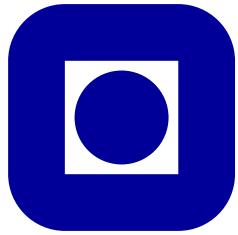
Norwegian University of Science and Technology
Department of Computer and Information Science

Johan G. Cyrus M. Ræder

Automatic Sarcasm Detection in Twitter Messages

Master's Thesis, Spring 2016

Artificial Intelligence Group
Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering



Abstract

In the past decade, social media like Twitter have become popular and a part of everyday life for many people. Opinion mining of the thoughts and opinions they share can be of interest to, e.g., companies and organizations. The sentiment of a text can be drastically altered when figurative language such as sarcasm is used. This thesis presents a system for automatic sarcasm detection in Twitter messages.

To get a better understanding of the field, state-of-the-art systems for detecting sarcasm in Twitter messages are explored. Many such systems already exist, and a common theme among them is the use of automatically annotated data for both training and testing. In addition to presenting a system for detecting sarcasm, this thesis also looks into the use of manually annotated data for testing. To this end, a dataset of tweets manually annotated with respect to the presence of sarcasm was built. The result was very similar to that of a previously made set, and both of them showed considerable deviation from automatic annotation. This implies that using automatically annotated data for the task of sarcasm detection in tweets is a mediocre approximation.

Experiments with both of the manually annotated datasets also gave very similar results, showing that they are well annotated and reasonably representative for sarcasm detection in tweets.

Sammendrag

I løpet av det siste tiåret har sosiale medier som Twitter blitt svært populære, og er en del av hverdagen for mange mennesker. Meningsutvinning av tankene og meningene folk deler med hverandre kan være av interesse for f.eks. selskaper og organisasjoner. Meningen i en tekst kan bli drastisk endret ved bruk av figurativt språk slik som sarkasme. Denne oppgaven presenterer et datasystem for automatisk gjenkjenning av sarkasme i meldinger på Twitter.

For å få en bedre forståelse av fagfeltet har state-of-the-art datasystemer for sarkasmegjenkjenning i Twitter blitt undersøkt. Mange slike systemer finnes allerede, og en vanlig fremgangsmåte blant dem er å bruke automatisk annoterte data til både trening og testing. I tillegg til å presentere et datasystem for sarkasmegjenkjenning, ser denne oppgaven også på bruken av manuelt annoterte data for testing. For å oppnå dette har et datasett av twittermeldinger manuelt annotert med hensyn på sarkasme blitt laget. Resultatet var veldig likt resultatet til et tidligere laget tilsvarende datasett. Begge viser også betydelige avvik fra automatisk annotering. Dette antyder at bruken av automatisk annoterte data er en middelmådig tilnærming til problemet.

Eksperimenter gjennomført med begge de manuelt annoterte datasettene gir også veldig like resultater, noe som viser at de er godt annoterte, og rimelig representative for sarkasmegjennkjenning i twittermeldinger.

Preface

This Master's Thesis has been completed as part of a Computer Science Master's Degree programme at the Department of Computer and Information Science at the Norwegian University of Science and Technology (NTNU). The thesis was supervised by Björn Gambäck.

Johan G. Cyrus M. Ræder
Trondheim, June 13^{th}, 2016

Acknowledgements

I would like to thank my supervisor Björn Gambäck for his guidance throughout the project. I would also like to thank Valerij Fredriksen, Brage Ekroll Jahren, Mathieu Cliche, and Ellen Riloff for providing data.

Contents

1. Introduction **1**
1.1. Project Goals . . . 2
1.2. Defining Sarcasm . . . 2
1.3. Contributions . . . 3
1.4. Thesis Structure . . . 4

2. Background **5**
2.1. Tweet Anatomy . . . 5
2.2. Part-of-Speech Tagging . . . 5
2.3. Algorithms . . . 6
2.3.1. Support Vector Machines . . . 6
2.3.2. Logistic Regression . . . 7
2.4. Tools and Resources . . . 8
2.4.1. Scikit-learn . . . 8
2.4.2. NLTK . . . 8
2.4.3. NRC Hashtag Sentiment Lexicon . . . 8
2.4.4. AFINN . . . 9
2.5. Evaluation Metrics . . . 9
2.5.1. Inter-Annotator Agreement . . . 9
2.5.2. System Performance . . . 10

3. Related Work **13**
3.1. State-of-the-Art . . . 13
3.1.1. SemEval 2015 . . . 13
3.1.2. Feature Selection . . . 15
3.2. Datasets . . . 17
3.3. Preprocessing . . . 19

4. Data **21**
4.1. Datasets 21
4.2. Annotating Tweets for MAD2 23

5. Architecture **25**
5.1. Features 25
5.2. System Design 27
5.2.1. Preprocessing 27
5.2.2. Workflow 27

6. Experimental Setup **31**
6.1. Algorithms and Grid Search 31
6.2. Experiments 33
6.3. Baselines 33

7. Results **35**

8. Discussion and Conclusion **53**
8.1. Discussion 53
8.2. Conclusion 55
8.3. Future Work 56

Bibliography **58**

A. List of Emoticons Converted to ASCII **65**

List of Figures

5.1. An overview of the system architecture. 29

7.1. Normalized confusion matrix for the AAD 10-fold cross validation using all features and the `SVC` classifier. 36
7.2. Normalized confusion matrix for the AAD 10-fold cross validation using lexical features and the `SVC` classifier. 37
7.3. Normalized confusion matrix for the AAD 10-fold cross validation using sentiment features and the `SVC` classifier. . . . 38
7.4. Normalized confusion matrix for the AAD 10-fold cross validation using all features and the `LogisticRegression` classifier. 39
7.5. Normalized confusion matrix for the AAD 10-fold cross validation using lexical features and the `LogisticRegression` classifier. 40
7.6. Normalized confusion matrix for the AAD 10-fold cross validation using sentiment features and the `LogisticRegression` classifier. 41
7.7. Normalized confusion matrix for the testing with MAD1 using all features and the `SVC` classifier. 42
7.8. Normalized confusion matrix for the testing with MAD1 using all features and the `LogisticRegression` classifier. . 43
7.9. Normalized confusion matrix for the testing with MAD1-auto using all features and the `SVC` classifier. 44
7.10. Normalized confusion matrix from testing with MAD1-auto using all features and the `LogisticRegression` classifier. . 45
7.11. Normalized confusion matrix for the testing with MAD2 using all features and the `SVC` classifier. 46
7.12. Normalized confusion matrix from testing with MAD2 using all features and the `LogisticRegression` classifier. . . 47

7.13. Normalized confusion matrix for the testing with MAD2-auto using all features and the `SVC` classifier. 48
7.14. Normalized confusion matrix from testing with MAD2-auto using all features and the `LogisticRegression` classifier. . 49

List of Tables

3.1. Results for Task 11 in SemEval 2015. Taken from Ghosh et al. [2015a]. 14

4.1. Rough quality check of the AAD. 22
4.2. Statistics for the different data sets. 22
4.3. Breakdown over inter-annotator agreement for the 2205 tweets in MAD2. 24

5.1. Words identified by Ghosh et al. [2015b] as good candidates for being used sarcastically. 26

6.1. Parameter values for the coarse grid search. 32
6.2. Parameter values for the fine grid search. 32
6.3. List of experiments. 33
6.4. Performance of theoretical baselines for the different data sets. 34

7.1. Evaluation metrics and raw classification numbers for the AAD 10-fold cross validation using all features and the `SVC` classifier. 36
7.2. Evaluation metrics and raw classification numbers for the AAD 10-fold cross validation using lexical features and the `SVC` classifier. 37
7.3. Evaluation metrics and raw classification numbers for the AAD 10-fold cross validation using sentiment features and the `SVC` classifier. 38
7.4. Evaluation metrics and raw classification numbers for the AAD 10-fold cross validation using all features and the `LogisticRegression` classifier. 39

7.5. Evaluation metrics and raw classification numbers for the AAD 10-fold cross validation using lexical features and the `LogisticRegression` classifier. 40
7.6. Evaluation metrics and raw classification numbers for the AAD 10-fold cross validation using sentiment features and the `LogisticRegression` classifier. 41
7.7. Evaluation metrics and raw classification numbers from testing with MAD1 using all features and the `SVC` classifier. . . 42
7.8. Evaluation metrics and raw classification numbers from testing with MAD1 using all features and the `LogisticRegression` classifier. 43
7.9. Evaluation metrics and raw classification numbers from testing with MAD1-auto using all features and the `SVC` classifier. 44
7.10. Evaluation metrics and raw classification numbers from testing with MAD1-auto using all features and the `LogisticRegression` classifier. 45
7.11. Evaluation metrics and raw classification numbers from testing with MAD2 using all features and the `SVC` classifier. . . 46
7.12. Evaluation metrics and raw classification numbers from testing with MAD2 using all features and the `LogisticRegression` classifier. 47
7.13. Evaluation metrics and raw classification numbers from testing with MAD2-auto using all features and the `SVC` classifier. 48
7.14. Evaluation metrics and raw classification numbers from testing with MAD2-auto using all features and the `LogisticRegression` classifier. 49
7.15. The classification of each class of tweet in percentages, using all features and the `SVC` classifier. 50
7.16. The classification of each class of tweet in percentages, using all features and the `LogisticRegression` classifier. 50
7.17. System performance compared to theoretical baselines for the different data sets. 51

1. Introduction

Natural language processing (NLP) is a field within computer science that deals with making computers able to meaningfully process human language. Some applications of NLP are document topic classification, language identification, and text summarization. Another area of use is sentiment analysis, also known as opinion mining. The goal here is to automatically extract a person's opinion about something from a text they've written, e.g., a customer review. As an example, in 2016, a Norwegian newspaper, developed a system that classified the description of a perpetrator as either favourable or unfavourable [Nipen et al., 2016]. They used this system to investigate how the judicial system treats the genders differently.

In recent times, social media have become an everyday part of many people's lives, and hence play a role in modern society. There are currently 1.65 billion monthly active users on Facebook [Facebook, 2016], and 310 million on Twitter [Twitter, 2016]. The number of people involved leads to a vast amount of data being shared between them, regarding everything from their daily lives to their (often vocal) opinions on a wide variety of matters. A lot of this information can be valuable to different companies, organizations, and other parties. Classic examples are a company wondering what people think of its newly launched product, or a politician interested in seeing people's reactions to a political debate. The amount of data involved in this can quickly become too much to handle manually. These traits make social media an interesting platform to explore opinion mining on. Of special interest is Twitter. Messages on Twitter, called *tweets*, are subject to a character limit, which forces the author to get to the point. Using a feature of tweets called hashtags, it is also easy to organize them by topic. Posting a tweet is an easy process, and people often respond swiftly to topics they are engaged in, resulting in quick feedback and discussion. Big events such as political debates or sports events will

often have a Twitter feed where people can post messages while the event is taking place.

When doing semantic analysis of language, there are several problems to tackle. For instance, people might not express themselves using clear concise language. Instead, they might do something like using figurative language, which is a prominent feature of human communication. Such language can drastically alter the meaning of an utterance compared to the literal interpretation. Because of this, it is necessary to find ways of dealing with figurative language when performing opinion mining.

Sarcasm is a form of figurative language which can have a clear effect on an utterance, i.e., alter or reverse its sentiment. Unfortunately, sarcasm isn't always easy to detect. It becomes even more difficult online than face to face, as cues such as facial expressions and change of vocal pitch are lost. In 2010, a man was arrested for a sarcastic tweet he wrote [Daily Mail, 2010]. To avoid such incidents, the United States Department of Homeland Security expressed interest in a sarcasm detector for Twitter [DHS, 2014].

1.1. Project Goals

G1: Investigate sarcasm and machine learning
Look into linguistic theory on sarcastic language and how machine learning can be used as a tool to automatically detect sarcasm in tweets.

G2: Experiment with different features and algorithms
Experiment with different combinations of features and machine-learning algortihms, and explain why the results turn out the way they do.

1.2. Defining Sarcasm

An exact, universal definition of sarcasm is hard to nail down. The Oxford Dictionary of English (not to be confused with the Oxford English Dictionary) defines sarcasm as[1]:

> "The use of irony to mock or convey contempt"

[1] `http://www.oxforddictionaries.com/definition/english/sarcasm`

Merriam Webster gives several definitions[2]:

> "A sharp and often satirical or ironic utterance designed to cut or give pain"
> "A mode of satirical wit depending for its effect on bitter, caustic, and often ironic language that is usually directed against an individual"
> "The use of words that mean the opposite of what you really want to say especially in order to insult someone, to show irritation, or to be funny"

The last definition is closest to the one that will be used here; for the purpose of this thesis, sarcasm will be defined as follows:

> *"Sarcasm is meaning the opposite of what you say"*

This definition was chosen because it is well defined and allows for relatively precise classification of tweets as sarcastic or not.

1.3. Contributions

C1: A dataset of tweets manually annotated as either sarcastic or not sarcastic.

C2: The implementation of a system classifying tweets as either sarcastic or not sarcastic.

C3: A comparison of different manually annotated datasets for the purpose of testing.

C4: A comparison of different machine learning algortihms and feature combinations for detecting sarcasm in tweets.

[2] `http://www.merriam-webster.com/dictionary/sarcasm`

1.4. Thesis Structure

Chapter 2 contains some background information the reader is expected to know before reading the rest of the thesis. The state-of-the-art analysis undertaken is presented in Chapter 3. The data used for training and testing the system are detailed in Chapter 4, along with information about the creation of a manually annotated dataset. In Chapter 5, the architecture of the system is presented. Experiments that have been performed are given in Chapter 6. Chapter 7 presents the results of the experiments. Finally, the discussion of the results is given in Chapter 8, together with the conclusion, and suggestions for future work.

2. Background

This chapter contains some useful information about tweets, and gives brief overviews of the classification algorithms chosen for the system. It also presents the main tools and resources that have been used in the implementation, as well as the evaluation metrics used later on.

2.1. Tweet Anatomy

In addition to containing text, tweets can contain URLs, hashtags, and mentions. A tweet that begins with "RT" is a *retweet*. A retweet is a tweet that one user has forwarded from another user. A *hashtag* is any word prefixed with the hashmark symbol "#", e.g., "#ThisIsAHashtag". Hashtags are useful for grouping tweets by topic. A *mention* is a reference to another Twitter user. This is done by prefixing the user's name with the "@" symbol, e.g., "@JohnDoe123". A tweet is considered a reply if it starts with a mention. A tweet can contain a maximum of 140 characters, including any URLs, hashtags, and mentions. However, URLs that are longer than a certain length will automatically be shortened. The rules regarding character length are being overhauled shortly after the completion of this thesis,[1] but this does not affect the tweets used here.

2.2. Part-of-Speech Tagging

The act of part-of-speech tagging (POS-tagging) is to assign a part-of-speech-tag (e.g., verb, noun, interjection) to each token (word, punctuation mark, etc.) of a sentence. Doing this is not as simple as having a dictionary where each token is listed with a POS-tag. The role a token

[1] `https://blog.twitter.com/express-even-more-in-140-characters`

plays in a sentence must also be taken into account. For instance, consider the following two sentences:

- "Did you read the text?"
- "Did he text you?"

In the first sentence, the word "text" is used as a noun, but in the second it is used as a verb. POS-tagging is a fairly well researched area, and several algorithms have been developed for doing it automatically. The best results on the English Wall Street Journal corpus are now above 97%, Spoustová et al. [2009] report achieving an accuracy of 97.43%. However, the accuracy for getting all the tokens in individual sentences right is not as high, Manning [2011] states a 56% accuracy for this problem.

2.3. Algorithms

In general, a classification algorithm is an algorithm that classifies objects into one of several classes. To do this, the algorithm must first learn. The straightforward way to do this is doing what is called supervised learning. When doing this, the algorithm is fed object examples whose classification is already known. By looking at predefined features of the examples, somewhat defined clusters should appear, e.g., examples of class X tend to have high values in one feature and low values in another. The algorithm then separates the clusters, and assigns a class to each based on the previously known classification. When presented with an example whose classification is unknown, values for the features are calculated, and the example is classified based on which cluster it ends up in.

2.3.1. Support Vector Machines

The support vector machine (SVM) classifer was introduced by Cortes and Vapnik [1995]. To learn, it separates classes of examples by creating a linear hyperplane that divides them (in the two-dimensional case this hyperplane would just be a straight line). The hyperplane is created such that it maximizes the margin between the two classes. The examples that end up defining the margin are called the support vectors. New examples can then be classified by which side of the hyperplane they fall on.

However, there is no guarantee that the training examples can be separated this way in the original input space. If this is the case, the examples can often still be linearily separated in a higher-dimensional space. To do this, the examples can be mapped into a higher-dimensional space using a transformation function, linearily separated there, and then the hyperplane is mapped back into the original input space.

In the mathematical formulas that constitute the SVM algorithm, the examples never appear by themselves, but always appear pairwise as inner products with each other. This means that individual examples don't need to be transformed. In fact, nothing needs to be transformed at all. It is sufficient to know what the result of the inner product between two examples in the higher-dimensional space is back in the original input space. The function giving this result is called the "kernel". Two examples are the "linear kernel": $K(\mathbf{u}, \mathbf{v}) = \mathbf{u} \cdot \mathbf{v} + c$, where c is some constant, and the "radial basis function kernal": $K(\mathbf{u}, \mathbf{v}) = exp(-\gamma \|\mathbf{u} - \mathbf{v}\|^2)$, where γ is some constant. Using a kernel is known as the "kernel trick", and can be more computationally efficient than using a transformation function.

2.3.2. Logistic Regression

Logistic regression (LReg) is a classification method that has its origins in linear regression. To train the classifier, each training example is multiplied by a weight vector, and then passed through the so called logistic function, which is a sigmoid:

$$Sigmoid(\mathbf{x} \cdot \mathbf{w}) = \frac{1}{1 + exp^{-\mathbf{x} \cdot \mathbf{w}}}$$

$$\mathbf{x} = [1, x_1, x_2, \ldots, x_n] \quad \textit{is an example.}$$
$$\mathbf{w} = [w_0, w_1, w_2, \ldots, w_n] \quad \textit{is the weight vector.}$$

For a given set of weights, the logistic function gives a number between 0 and 1 for each example in the training data. These numbers can be interpreted as the probability of belonging to a certain class. Comparing these numbers to the actual probabilities of the training data (the examples belonging to that class have probability 1, the others have probability 0), an

error term can be computed. This is done by taking the squared difference, just like in linear regression. The goal is to choose optimal weights so that the error term is minimized. Test examples are classified using the chosen weights and the logistic function.

2.4. Tools and Resources

The following tools and resources are key components of the realized system. They have all been previously employed by others in similar studies.

2.4.1. Scikit-learn

`Scikit-learn` [Pedregosa et al., 2011] is a package for Python. It contains implementations of many state-of-the-art machine-learning algorithms, as well as useful tools for processing data. It is released under the BSD license, and as such is freeware. One of `Scikit-learn`'s focuses is ease of use, thus maintaining one of Python's core ideologies. Although developed in a high-level language, `Scikit-learn` includes both precompiled code and libraries such as the C++ `libSVM` library [Chang and Lin, 2011] to achieve good performance. It is also well documented by its developers.

2.4.2. NLTK

`NLTK` (Natural Language Toolkit) [Bird et al., 2009] is a Python package designed to help build programs that work with human language data. It provides an interface with many corpora and lexical resources, and a wide range of tools for processing language data, e.g., tokenization, stemming, and part-of-speech tagging. `NLTK` is a free, open-source, project.

2.4.3. NRC Hashtag Sentiment Lexicon

The NRC Hashtag Sentiment Lexicon [Mohammad et al., 2013] is a lexicon containing terms and their associated sentiment values (a real number indicating how positive/negative that term is). The terms can be unigrams, bigrams, or pairs containing any combination of the two. The sentiment score is calculated by looking at how often a term co-occurs with a set of positive and negative hashtags. The data used to generate the lexicon is a

set of about 775,000 tweets collected in 2012. It was first developed as part of a twitter-sentiment-evaluation system that competed in a shared task organized by the Conference on Semantic Evaluation Exercises (SemEval-2013).

2.4.4. AFINN

The AFINN lexicon [Årup Nielsen, 2011] is a smaller lexicon containing 2462 words and 15 phrases, and their associated sentiment values. The terms have been manually selected and annotated by AFINN's author. The lexicon focuses on obscene words, internet slang, and emotional words. It was initially developed for analyzing tweets posted in relation to the COP15 United Nations Climate Conference.

2.5. Evaluation Metrics

Certain metrics are needed to evaluate system performance. This section presents metrics that are common in the field of study of this thesis, and which are used later on.

2.5.1. Inter-Annotator Agreement

In order to judge the agreement between two annotators that are classifying items into distinct classes, it is informative to take into account the hypothetical scenario of chance agreement between them. A measure that does this is Cohen's kappa, given by the formula below:

$$\kappa = \frac{p_0 - p_e}{1 - p_e}$$

Here, p_0 is the observed agreement between the annotators, and p_e is the hypothetical probability of them agreeing by chance. p_e is calculated by assuming each annotator answers randomly with the same distribution as their observed answers.

2.5.2. System Performance

A common way to evaluate the performance of a classification systems is by looking at the system's *precision*, *recall*, and *accuracy* scores. These scores are defined as follows:

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

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

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

Where tp is a true positive (an item correctly classified as belonging to a certain class), fp is a false positive (an item incorrectly classified as belonging to a certain class), tn is a true negative (an item correctly classified as not belonging to a certain class), and fn is a false negative (an item incorrectly classified as not belonging to a certain class).

As a simple example, let's say that we have a bunch of red M&Ms and a bunch of blue M&Ms, and a system that classifies them as either red or blue. Let's also say that we are currently regarding "red" as the positive class. If the system has a precision of 0.80, a recall of 0.67, and an accuracy of 0.90, then we know the following:

- 80% of all the M&Ms that are classified as red are indeed red, whereas 20% of them are blue.
- 67% of all the red M&Ms are correctly classified, whereas 33% of them are incorrectly classified.
- 90% of all the M&Ms are correctly classified, whereas 10% of them are incorrectly classified.

It is common to combine precision and recall into a single score known as the F-score, which was introduced by van Rijsbergen [1975]. The F-score is the weighted harmonic mean of the two, and the general formula is:

$$F_\beta = \frac{1+\beta^2}{\frac{1}{precision} + \beta^2 \frac{1}{recall}} = \frac{(1+\beta^2) \times precision \times recall}{(\beta^2 \times precision) + recall}$$

where recall is weighted with β^2, and precision is weighted with 1. It is common to use $\beta = 1$, so that the F-score becomes:

$$F_1 = \frac{2 \times precision \times recall}{precision + recall}$$

In the M&M example above, if we had regarded "blue" as the positive class instead of "red", the scores might have been different, even though the classification stays the same. To evaluate the system performance, it is therefore necessary to combine the two results. This is usually done by either micro-averaging or macro-averaging. If tp_{red} and fp_{red} denote true and false positives when "red" is the positive class, and tp_{blue} and fp_{blue} denote true and false positives when "blue" is the positive class, the different averages for precision are computed as follows:

$$Macro = \frac{\frac{tp_{red}}{tp_{red}+fp_{red}} + \frac{tp_{blue}}{tp_{blue}+fp_{blue}}}{2}$$

$$Micro = \frac{tp_{red} + tp_{blue}}{(tp_{red} + tp_{blue}) + (fp_{red} + fp_{blue})}$$

The averages for the other scores are computed analogously. Using micro-average will bias the result towards the largest class(es), macro-average will bias towards the smallest.

3. Related Work

The findings of the state-of-the-art study in Twitter sarcasm detection systems is presented here. Firstly, the SemEval-2015 workshop is presented. The chapter then looks at different ways of identifying sarcasm in tweets, and finally datasets and preprocessing are covered.

3.1. State-of-the-Art

3.1.1. SemEval 2015

SemEval is an international workshop on semantic evaluation that has been held in recent years. Each time, a number of different tasks are announced, and participants submit the system(s) they have developed to solve one or more tasks. The tasks usually cover a variety of topics, and for the 2015 edition of SemEval, the overarching topics were: Text Similarity and Question Answering, Time and Space, Sentiment, Word Sense Disambiguation and Induction, and Learning Semantic Relations. Of particular interest is Task 11, "Sentiment Analysis of Figurative Language in Twitter". The goal was to determine whether a tweet expresses positive, negative, or neutral sentiment. The crux of the task is that the data contains a large amount of figurative language, so the systems must be able to handle this to get accurate results. The training data used consists of 8000 tweets, out of which 5000 contain sarcasm, 1000 contain irony, and 2000 contain metaphor. The test data used consists of 4000 tweets, where 1200 contain sarcasm, 800 contain irony, 800 contain metaphor, and 1200 are classified as "other". The tweets have been manually annotated by humans (i.e., given a score on a sentiment scale). The systems were scored in two different ways. The first is the cosine similarity score. To calculate this score, the sentiment scores output by a participating system, and the manually given sentiment scores, are represented as vectors. The cosine

Team	Cosine	MSE
CLaC	0.758	2.117
UPF	0.711	2.458
LLT_PolyU	0.687	2.600
elirf	0.658	3.096
LT3	0.658	2.913
ValenTo	0.634	2.999
HLT	0.630	4.088
CPH	0.625	3.078
PRHLT	0.623	3.023
DsUniPi	0.602	3.925
PKU	0.574	3.746
KELabTeam	0.552	4.177
RGU	0.523	5.143
SHELLFBK	0.431	7.701
BUAP	0.059	6.785

Table 3.1.: Results for Task 11 in SemEval 2015. Taken from Ghosh et al. [2015a].

of the angle between them is then used as a measure of their similarity. A score of 0 means no similarity and a score of 1 means a perfect match. The second scoring method is the mean squared error (MSE). Systems were penalized if they didn't give a score to all the tweets in the test data. The organizers made three very simple systems to use as a baseline. These systems achieved cosine and MSE scores of: [0.390, 5.672], [0.426, 5.450], and [0.547, 4.065]. Further information can be found in Ghosh et al. [2015a] and on the webpage `http://alt.qcri.org/semeval2015/`. The results for the task are reproduced in Table 3.1. All systems except the bottom three beat the baselines. The MSE ranking order is not exactly the same as the cosine ranking order, but is generally pretty close (with HLT being the notable exception). The systems were also scored for each of the four different types of figurative language (irony, sarcasm, metaphor, other) that were present in the data. The results for irony and sarcasm were a lot better than metaphor and other across all systems.

3.1.2. Feature Selection

When designing a system for sarcasm detection in tweets, it is necessary to choose certain characteristics, called features, of the tweets to look at. For example, one might look for the presence of certain phrases, or give the tweet a score based on the number of punctuation marks used. Well chosen features will hopefully give a strong indication of whether a tweet is sarcastic or not. Feature selection is therefore an important part of the system development process.

Punctuation: Punctuation symbols are any symbols used to help ease the readability and help convey the message of a text. These include commas, colons, semi-colons, full stops, exclamation marks, question marks and more. Using punctuation as an indicator involves looking for several things. One is the amount of punctuation used. Heavy use of punctuation can be a good indicator of sarcasm [Carvalho et al., 2009]. Another is looking for certain combinations, such as a bunch of exclamation and question marks together, or three consecutive dots (commonly used to mark ellipsis), which can be a strong indicator of sarcasm [Davidov et al., 2010].

Emoticons: Emoticons are sequences of characters used to convey emotions, such as the smiley ":)". Davidov et al. [2010] found that emoticons (and onomatopoeic expressions for laughter) are helpful for detecting sarcasm. It was also shown by González-Ibáñez et al. [2011] that emoticons are important when humans try to identify sarcasm in tweets, so it is not surprising that some of these can be used by an automatic system. However, Wang and Castanon [2015] showed that while some emoticons are used very consistently (i.e., always conveying the same emotion), others are used inconsistently. This must be taken into account when using emoticons.

n-grams: An n-gram is simply a unit consisting of n adjacent words (n = 1,2,3...). When n = 1 it's called a unigram, n = 2 a bigram, and n = 3 a trigram. For n greater than 3 it's referred to by its numerical value (e.g., seven-gram). For the sentence *"I like cake"*, the unigrams are "I", "like", "cake", the bigrams are "I like", "like cake", and the trigram is "I like

cake". The idea behind using n-grams as features is that certain n-grams will be good identifiers for figurative language. A more general version of n-grams exist, called skip-grams. The difference is that in skip-grams the words don't necessarily need to be adjacent.

Part-of-speech: As described in Section 2.2, POS-tagging is the act of tagging each token in a text with its POS-tag. This information can then be used to create one or more features for the classifiers. For instance, Carvalho et al. [2009] suggested looking at the number of interjections. Several papers report good results from features derived from POS-tagging, e.g., [Barbieri et al., 2015] (UPF in Table 3.1) and [Karanasou et al., 2015] (DsUniPi in Table 3.1). Another benefit is that POS-tagging helps to distinguish between words that are different, but happen to have the same spelling.

Sentiment: A common strategy is to include the use of sentiment dictionaries, see for instance [Özdemir and Bergler, 2015] (CLaC in Table 3.1) and [Barbieri et al., 2015]. These dictionaries usually contain sentiment values for individual words, but can also contain sentiment values for other things, e.g., hashtags. The two dictionaries mentioned in Section 2.4 (AFINN and NRC) were highlighted by Özdemir and Bergler [2015] as being good dictionaries. The information provided by sentiment dictionaries can be used to create various features. Xu et al. [2015] (LLT in Table 3.1) use several dictionaries to look at the overall sentiment of tweets, as well as looking for sentiment polarity shifts, e.g., when a positive verb is used with reference to a negative clause. A similar approach was taken by Riloff et al. [2013]. Their system learns a set of positive verb phrases (e.g., "love", "excited", "can't wait"), and a set of negative situation phrases (e.g., "being ignored", "not getting", "doing homework"), and then look for these in tweets. The theory being that a polarity shift is an indicator of sarcasm. Joshi et al. [2015] also use this idea by looking for incongruity in tweets, i.e., when a positive word is followed by a negative word and vice versa.

User data: Some attempts have been made at using data about the author to help detect sarcasm. Given a tweet about a topic, Khattri et al. [2015] use the author's previous tweets about that topic to help detect

sarcasm. If the general sentiment of the previous tweets doesn't match the sentiment of the tweet in question, then this increases the chance of sarcasm. Note that people's opinions can change over time, so one should focus on the author's more recent tweets about the topic. This approach also assumes that the author's previous posts are factual. Other information about the author was looked at by Bamman and Smith [2015]. The most useful feature they found was a binary indicator of certain terms used by the author. In other words, an author would tend to use certain terms when being sarcastic, and not use those terms when not being sarcastic. Which terms to look for was decided on an individual basis by going through an author's tweets and selecting terms with high TF-IDF scores (a statistical score measuring how important a word is to a document). As pointed out by Khattri et al. [2015], obtaining the historical tweets of an author isn't always going to be possible, which is a weakness of using these features.

Other: Several other features have been mentioned, such as excessive use of upper case letters and use of ambiguous words [Barbieri et al., 2015], and temporal imbalance [Hee et al., 2015] (LT3 in Table 3.1). Karanasou et al. [2015], motivated by the work of Hao and Veale [2010], looked for patterns in the language such as *"as * as *"*. However, the contribution of these features seems only marginal. Ghosh et al. [2015b] looked at what they called the "literal/sarcastic sense disambiguation task" (LSSD). Given an utterance and a target word in that utterance, they aimed at identifying whether the target word was used in a literal or sarcastic sense. In doing so they (among other things) found a set of words that are often used sarcastically, see Section 5.1. Maynard and Greenwood [2014] developed a hashtag tokenizer, to correctly tokenize hashtags consisting of multiple words, e.g., #imsolucky. They then used the information contained in hashtags to look for sentiment and sarcasm in tweets.

3.2. Datasets

Obviously, data is required to train and test the system. One of the common ways of getting a dataset when working with Twitter is to use the Twitter API to download a large amount of tweets. When looking for

tweets containing a certain kind of content (e.g., sarcasm), it is possible to use hashtags to find tweets that might contain that kind of content. For the case of sarcastic messages, a common approach has been to download tweets with the hashtags #sarcasm and #sarcastic (see for instance [Liebrecht et al., 2013], [Forslid and Wikén, 2015], [Ghosh et al., 2015b]). The assumption here is that authors generally label their tweets correctly. An issue with this kind of dataset is that some of the tweets might be about sarcasm instead of containing sarcasm [Davidov et al., 2010]. To address this issue, González-Ibáñez et al. [2011] suggest only using tweets where the hashtag of interest appears at the very end. This doesn't eliminate the problem entirely though. The following tweet would still be in the dataset, even though it doesn't contain sarcasm: *"@username, he was using #sarcasm"*. Another issue raised by Davidov et al. [2010] is that the data might be biased towards the hardest forms of sarcasm, where the reader needs an explicit marker to understand that it's sarcasm. However, Bamman and Smith [2015] found that users are more likely to explicitly state sarcasm when they are not very familiar with their audience. The bias, if there is any, might therefore lie more towards sarcasm used between people that don't know each other very well, rather than towards difficult forms of sarcasm.

Using hashtags to select sarcastic and non-sarcastic tweets is known as automatic annotation. Another way of doing it is through manual annotation. Due to time constraints this will obviously lead to a smaller dataset, but one has complete control over what the dataset contains. This does not guarantee, however, that the dataset will be flawless. Humans can make mistakes about and disagree on the meaning of what someone has said or written, and as shown by González-Ibáñez et al. [2011], detecting sarcasm in tweets is a difficult task for humans. Also, McGillion et al. [2015] (CPH in Table 3.1) report that the dataset used in the SemEval workshop contains some repeated tweets that hadn't always received the same score by the annotators.

When collecting tweets it is possible, as just mentioned, to end up with duplicate tweets and retweets; these should be removed. Forslid and Wikén [2015] also suggest eliminating any tweets containing URLs or pictures, as their contents might be necessary to identify any figurative language present in the tweets.

3.3. Preprocessing

Before further work is done with the data, it can be a good idea to process it to achieve some form of standardization. Components that don't provide much information such as URLs and mentions can either be removed entirely or be replaced by general URL and mention tokens. Words that are spelled in unconventional ways, e.g., elongated words ("loooove" vs. "love"), can be replaced with the conventional spelling. However, if this information is going to be used as a feature later on, it must be saved. Emoticons can be replaced with a token signaling the emotion they express. Unusual punctuation sequences can be replaced, e.g., long sequences of exclamation and question marks can be replaced by a single "!?". Any undecipherable sequence of characters can be removed.

4. Data

Data forms a key part of the experiments underlying this thesis, and the datasets used for training and testing the system are presented here. Details on the creation of a manually annotated set that is used as one of the test sets are also given.

4.1. Datasets

As mentioned in Section 3.2, it is common to use data automatically annotated based on the presence of the hashtags #sarcasm and #sarcastic. Cliche [2014] collected such a dataset for his online sarcastic tweet detector, consisting of about 150,000 tweets with #sarcasm and 330,000 without. For this thesis, after the preprocessing (see Section 5.2.1), 100,000 of those tweets were chosen to be used as an automatically annotated dataset, out of which 20,000 have the sarcasm hashtag, and 80,000 don't. In the rest of this, thesis this dataset will be referred to as "AAD" (automatically annotated dataset). As a rough quality check of the dataset, 100 tweets with the sarcasm hashtag and 200 without were selected at random and manually investigated by the author, see Table 4.1. Assuming that tweets without a sarcasm hashtag are not sarcastic seems to be a good approximation, but the converse doesn't seem quite as good.

In addition, two manually annotated sets are also used. The first, which will be called "MAD1" (manually annotated dataset nr. 1) from now on, is a set made by Riloff et al. [2013]. The original set consists of 3200 tweets, half of which contain the sarcasm hashtag and half of which don't. The tweets were individually annotated by three annotators following certain guidelines. The result was that 742 of them were judged to be sarcastic. Unfortunately, some of the tweets have been deleted and are no longer available for download from Twitter. As such, MAD1 consists of 2116 tweets, out of which 1047 have the sarcasm hashtag, 1069 don't, and 459

	Tweets with #sarcasm	**Tweets without #sarcasm**
Number of tweets	100	200
Not Sarcastic	9	199
Sarcastic	64	1
Difficult to identify as sarcastic without the hashtag	27	N/A

Table 4.1.: Rough quality check of the AAD.

	AAD	**MAD1**	**MAD2**
Number of tweets	100,000	2116	2205
Tweets with #sarcasm	20%	49%	51%
Tweets without #sarcasm	80%	51%	49%
Annotated as sarcastic	N/A	22%	25%
Annotated as not sarcastic	N/A	78%	75%

Table 4.2.: Statistics for the different data sets.

are judged to be sarcastic.

The second manually annotated set, hereinafter referred to as "MAD2" (manually annotated dataset nr. 2), is made in a similar fashion to the first. 2205 tweets, 1115 with the sarcasm hashtag and 1090 without, were individually annotated by the author and two additional annotators. Out of all of the tweets, 554 were judged to be sarcastic. The statistics for the different data sets are summarized in Table 4.2. More details on how the MAD2 set was created are given in the next section.

If we assume that the data in Table 4.1 is valid for the whole automatically annotated dataset (a somewhat shaky assumption given the small sample size and single annotator), we see that about 87% of the tweets in the AAD would be annotated as not sarcastic, and about 13% as sarcastic. This is different from the two other sets, that have similar distributions to each other. If the AAD had contained 50,000 of each kind of tweet, the numbers (i.e., those marked by "N/A" in Table 4.2) would instead be 68% not sarcastic and 32% sarcastic.

4.2. Annotating Tweets for MAD2

As it's not certain that #sarcasm works well for identifying whether or not tweets are sarcastic, it is desirable to have manually annotated data to work with. As only smaller such sets were found, they could only be used for testing, rather than for both training and testing. In addition to acquiring MAD1, it was decided to make another similar set for two main reasons. Firstly, more data will hopefully give a better indication of system performance. Secondly, if there are big differences in system performance on the two sets, it will be interesting to study, as finding the root cause of those differences can yield some useful insight.

The tweets used for MAD2 were taken from a larger set of about 103 million tweets collected using the Twitter API during the period December 2015 - March 2016 by two fellow students working on Twitter sentiment analysis in a parallell project in the AI group at NTNU. More information about the data collection process can be found in Fredriksen and Jahren [2016]. In all, 2205 tweets, 1115 with the sarcasm hashtag and 1090 without were used. As with MAD1, the sarcasm hashtags were removed prior to annotation to prevent the annotators from being influenced by knowing whether or not a tweet is meant sarcastically. The tweets were given in random order, and tweets containing URLs were not included in the set. Three annotators, the author and two others, individually annotated the tweets. Like the author, the two other annotators are not native English speakers. However, they use English regularily in their line of work, and both of them have received formal education from, and lived several years in, the United States. Like many young Norwegians, the author has had significant exposure to the English language through education, TV, books, online media, and video games. The guidelines for annotation were the following:

1. Adhere to the definition of sarcasm used in this thesis.

2. For a tweet to be judged as sarcastic it must be sarcastic in and of itself. Trying to guess the context of a tweet should be avoided.

3. Tweets that can be interpreted as either sarcastic or not sarcastic should be judged as not sarcastic.

4. If an annotator doesn't understand a tweet or for some other reason is extremely unsure about how to judge it, the tweet is to be set aside and brought up for discussion with the other annotators to clarify its content.

The reasoning behind the second and third rules is that if a tweet needs the right context or interpretation to be sarcastic, the language (words, phrasing, emoticons, etc.) is not clear enough to be considered sarcastic.

	Number of Tweets
3-0 in favour of sarcastic	366
2-1 in favour of sarcastic	188
3-0 in favour of not sarcastic	1366
2-1 in favour of not sarcastic	285

Table 4.3.: Breakdown over inter-annotator agreement for the 2205 tweets in MAD2.

Unfortunately, due to a mistake by the author, the tweets were converted to lower case before given for annotation, so any upper case letters were seen as lower case by the annotators. The annotators were aware that roughly half the tweets used to have #sarcasm in them. In total, 35 tweets were brought up for discussion to clarify their content, and were then annotated. The final judgement for whether or not a tweet is sarcastic was done by a majority rules scheme, and a total of 554 tweets ended up being judged as sarcastic. The pairwise inter-annotator agreement scores calculated using Cohen's kappa are 0.59, 0.60, and 0.71. For comparison, they are 0.80, 0.81, and 0.82 for MAD1 (these values are for 200 of the original 3200 tweets, 100 with and 100 without the sarcasm hashtag). A breakdown over the number of total and partial agreements is given in Table 4.3.

5. Architecture

The design of the implemented system is described next. First, the features that have been chosen are detailed. Then the workflow of the system is described, from the initial preprocessing of data, to the final classification result.

5.1. Features

The following features were implemented in the system:

Unigrams: Certain words might be used more in either sarcastic or non-sarcastic tweets, and the goal of this feature is to capture these words. To that end, a dictionary containing all the unigrams in the dataset is created, and each unigram is considered a feature. However, this quickly leads to too many features for a personal computer to handle, so two measures are put in place to reduce the number of unigrams. First, a threshold is set for the minimum number of times a unigram has to occur, unigrams that appear fewer times than this threshold are discarded. In addition to help reduce the number of features, this also prevents overfitting the system to unusual vocabulary that might be specific to the data. The second measure is to only include unigrams whose distributions are skewed more than a certain ratio. A disadvantage with this is that clusters of words that together indicate either sarcasm or not sarcasm, but individually are insignificant, will be disregarded. The threshold for a unigram to be included is set to 200, and the distribution has to have a skewed ratio of at least 1.5. These numbers were chosen by trial and error using AAD, but the choice was somewhat constrained by the power of the computer used to run the system.

love	like	great	good	really
best	better	glad	yeah	nice
happy	cool	amazing	favorite	perfect
super	fantastic	joy	cute	beautiful
shocked	interested	brilliant	genius	mature
right	fun	attractive	lovely	proud
awesome	excited	always	sweet	hot
wonderful	wonder			

Table 5.1.: Words identified by Ghosh et al. [2015b] as good candidates for being used sarcastically.

Part-of-Speech: Sarcastic sentences might have characteristic structures such as excessive use of interjections. Such structures might be captured by POS-tagging the tweets. Each tag is used as a feature.

Punctuation: Similarily to the POS-tags, certain usages of punctuation such as heavy use of full stops might be characteristic of sarcasm. For each of a set of punctuation characters, a ratio of occurence of that character to the length of the tweet is considered a feature. The punctuation characters used were exclamation marks, question marks, colons, semicolons, commas, full stops, quotation marks, and ellipsis.

Sentiment: Various measures using sentiment values are used as features. Since sarcastic tweets are often negative, the sentiment of the whole tweet is considered as a feature. Furthermore, several sentiment differences are considered. One is the difference between the words of a tweet and the emoticons. In a sarcastic sentence, there can sometimes be one key word or phrase which is used sarcastically. Another feature consists of regarding every verb and noun as a possible such word, and comparing its sentiment value to the value of the rest of the tweet. The same is done for a short list of words that have been identified as good candidates for being used sarcastically by Ghosh et al. [2015b], see Table 5.1. Lastly, sarcasm can involve sudden shifts in sentiment polarity. As such, the sentiment differences between adjecent words of a tweet are considered a feature.

5.2. System Design

The system is written in the programming language Python. Instead of using the official release, a distribution of Python called Anaconda[1] is used. Anaconda comes with a slew of useful packages for Python already installed, that aren't included in the official release, e.g., `Scikit-learn` and `NLTK`, see Section 2.4.

5.2.1. Preprocessing

For the AAD set, tweets containing URLs were removed. Tweets where the sarcasm hashtag is placed somewhere other than at the end were also removed. If a tweet ends with a sequence of hashtags, the entire sequence is considered the end of the tweet. For instance, the following tweet would *not* be discarded: *"i love waking up at 5 am #sarcasm #notreally #fml"*. Out of the remaining tweets, 100,000 were selected for the AAD. The tweets were then converted to lower case, and a selection of common emoticons were converted from Unicode to ASCII, see Appendix A.

For the MAD1 set, the tweets were converted to lower case and the same list of emoticons were converted from Unicode to ASCII, but otherwise they were left untouched. The MAD2 set was treated the same way as the AAD set, with the removal of tweets containing URLs and having the sarcasm hashtag somewhere other than at the end, as well as conversion to lower case and conversion of emoticons.

5.2.2. Workflow

For an overview of the system workflow, see Figure 5.1. After the tweets are preprocessed, they are fed to the system. The initial step is tokenization and POS-tagging. The tokenization is done by a special tokenizer from `NLTK` called `TweetTokenizer`, which is calibrated for more casual language, like the kind of language often found in tweets. The POS-tagging is done by the standard `NLTK` POS-tagger, which uses the "University of Pennsylvania Treebank Tag-set",[2] with a few additions for special characters and symbols. The following set of tags were not used as features due

[1] `https://www.continuum.io/downloads`

[2] `www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`

to either beeing deemed unimportant, or being part of the punctuation features: {“$”, “”, “(”, “)”, “,”, “–”, “.”, “:”, “CD”, “FW”, “LS”, “POS”, ““”, “#”}. The system then makes a pass through all the tokens to make the dictionary of unigrams. Since the hashtag #sarcasm wasn't removed from the tweets in the preprocessing stage, it ends up as one of the unigrams. After the dictionary is finished, #sarcasm is explicitly removed from it.

With all the groundwork done, the system then generates the feature matrix based on the chosen combination of features. The sentiment features make use of the AFINN lexicon [Årup Nielsen, 2011] and the NRC lexicon [Mohammad et al., 2013] to calculate needed sentiment values. As each tweet only has a minority of the total number of features (mostly due to containing only a few of all the unigrams in the dictionary), it is important to store the feature matrix in a sparse format to save memory. The feature matrix is then given as input to a classifier. The system uses one of two classifiers, either the `LogisticRegression` or the `SVC` classifer. Both are from the `Scikit-learn` package, the latter being an implementation of the Support Vector Machine algorithm. See section 2.3 for a description of the algorithms used by these classifiers. Features are then calculated for the test data, and the predicton function yielded by the classifer is used to classify the data. Finally, the result is compared to the gold standard (automatic/manual annotation) for the test data to evaluate classifier performance.

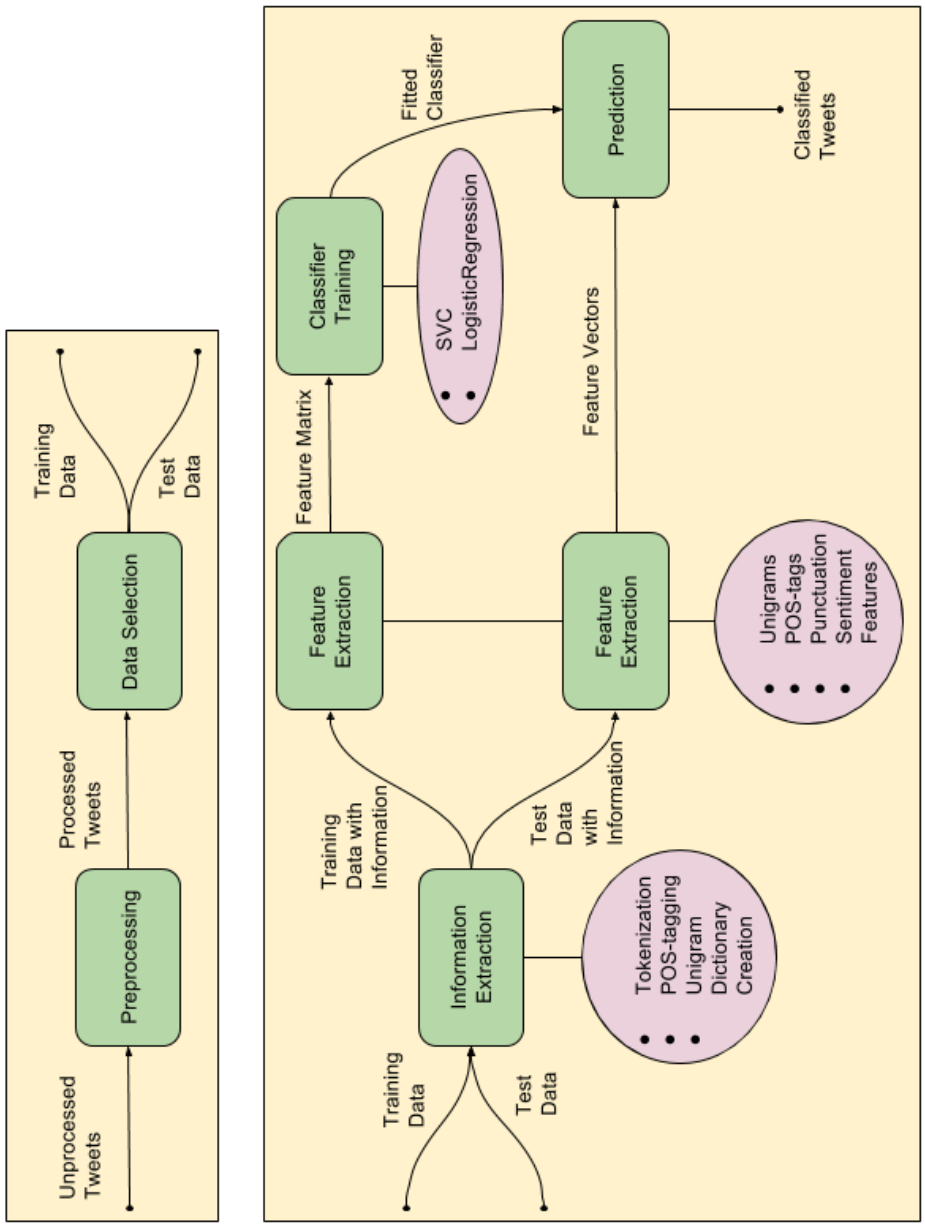


Figure 5.1.: An overview of the system architecture.

6. Experimental Setup

After the system is implemented, a grid search is performed to optimize its performance, the details of which are given in this chapter. The experiments that have been chosen for testing the system, along with the baselines used to judge system performance, are also presented here.

6.1. Algorithms and Grid Search

Some preliminary testing was done with four different classification algorithms commonly used in Twitter NLP applications: Decision Trees, LReg, Naïve Bayes, and SVM. The best performances were given by LReg and SVM, hence these were chosen to conduct further testing.

To improve the perfomance of the classifiers, a grid search was performed using the AAD with 10-fold cross validation. To do this, the AAD was divided into ten folds, each with 10,000 tweets. The tweets for each fold were selected at random, but under the constraint that all folds ended up being stratified, i.e., they all have the same distribution of sarcastic (20%) and non-sarcastic (80%) tweets as the AAD. One by one, each fold was used as the test data, while the other nine were used as training data, leading to 10 different data configurations. For each of these cofigurations, the system was run multiple times with different parameter settings. The parameter settings that led to the overall highest scores were chosen as the best. The grid search process was performed twice, first it was done on a coarse level, and then on a finer level, centered around the best result from the coarse search. The parameters used in the grid search for LReg are "C", which controls the regularization strength, and "penalty", which dictates the penalization norm. For SVM the parameters are "C", which controls the regularization strength, "kernel", which dictates the kernel, and "γ", which is a parameter used by certain kernels, see Section 2.3.1. The results of the coarse searches are given in Table 6.1, and the results

LReg	
Parameter	**Values**
C	$\{0.01, 0.1, 1, 10, 100\}$
Penalty	{'l1', 'l2'}
Best result: {1, 'l2'}	
SVM	
Parameter	**Values**
C	$\{0.1, 1, 10\}$
Kernel	{'linear', 'rbf'}
γ (only used with 'rbf' kernel)	$\{0.0001, 0.001, 0.01\}$
Best result: {1, 'rbf', 0.01}	

Table 6.1.: Parameter values for the coarse grid search.

LReg	
Parameter	**Values**
C	$\{0.4, 0.6, 0.8, 1, 1.2, 1.4, 1.6\}$
Penalty	{'l2'}
Best result: {1, 'l2'}	
SVM	
Parameter	**Values**
C	$\{0.6, 0.8, 1, 1.2, 1.4\}$
Kernel	{'rbf'}
γ	$\{0.006, 0.008, 0.01, 0.012, 0.014\}$
Best result: {1.4, 'rbf', 0.014}	

Table 6.2.: Parameter values for the fine grid search.

of the fine searches are given in Table 6.2. Since the best result for the SVM is one of the corner points of the grid (C = 1.4 and $\gamma = 0.014$), it is possible that even better parameters could have been found through more searching. This was not done due to time constraints.

LReg Parameters: C = 1.0, penalty = 'l2'		
SVM Parameters: C = 1.4, kernel = 'rbf', γ = 0.014		
Both classifiers are used for all experiments		
Training Data	**Test Data**	**Features**
AAD 10-fold cross validation		All
AAD 10-fold cross validation		Sentiment
AAD 10-fold cross validation		Lexical
AAD	MAD1	All
AAD	MAD2	All
AAD	MAD1-auto	All
AAD	MAD2-auto	All

Table 6.3.: List of experiments.

6.2. Experiments

To gauge the effect of the different features, three 10-fold cross validations are run using the best parameters found in the grid search. The first uses all the features, the second uses only the sentiment features, and the third uses only the lexical features (unigrams, POS-tags, punctuation). The classifier is then trained on the whole AAD and tested using all the features on MAD1 and MAD2. Lastly, instead of using the manual annotation, the MAD1 and MAD2 sets are automatically annotated based on #sarcasm (these sets will be referred to as "MAD1-auto" and "MAD2-auto"), and the system is tested with all features on these as well. Table 6.3 lists the experiments.

6.3. Baselines

To assess the skill of the system, it will be compared to two theoretical baseline systems. The first system always classifies a tweet as the most frequent class in the training data. The second system classifies at random, with an even chance for both classes. The performance of these systems on the different data sets is given in Table 6.4. For a classification system with only two classes, if the precision, recall, and F-score are calculated

Most Frequent Classifier				
	Precision	**Recall**	**F_1-score**	**Accuracy**
AAD	0.40	0.50	0.44	0.80
MAD1	0.39	0.50	0.44	0.78
MAD2	0.37	0.50	0.43	0.75
MAD1-auto	0.25	0.50	0.34	0.51
MAD2-auto	0.25	0.50	0.34	0.51

Random Classifier				
	Precision	**Recall**	**F_1-score**	**Accuracy**
AAD	0.50	0.50	0.45	0.50
MAD1	0.50	0.50	0.46	0.50
MAD2	0.50	0.50	0.47	0.50
MAD1-auto	0.50	0.50	0.50	0.50
MAD2-auto	0.50	0.50	0.50	0.50

Table 6.4.: Performance of theoretical baselines for the different data sets.

using the micro-average (see Section 2.5.2), they will all be equal to the accuracy. The scores given for the baselines are computed using the macro-average. Since the accuracy is the same regardless which average is used, all the micro-averaged scores can still be seen by looking at the accuracy.

In addition to the theoretical baselines, some comparisons will also be done with results from work done by others using similar data, more on this in Section 8.1.

7. Results

The results of the experiments described in Table 6.3 are presented in this chapter. For each experiment, the precision, recall, F-score, and accuracy are listed. In addition, the raw classification numbers are also given, under the labels “Correct S”, “Correct NS”, “Incorrect S”, and “Incorrect NS”. These have the following meanings:

Correct S: A sarcastic tweet correctly classified as sarcastic.
Correct NS: A non-sarcastic tweet correctly classified as not sarcastic.
Incorrect S: A non-sarcastic tweet incorrectly classified as sarcastic.
Incorrect NS: A sarcastic tweet incorrectly classified as not sarcastic.

The confusion matrix for each experiment is also given. The confusion matrices are normalized to the range $[0, 1]$. The upper-left and lower-right squares indicate correctly classified tweets. A good result will have high values in these, represented by dark green colours, and low values in the lower-left and upper-right squares, represented by dark brown colours. Observations made for each result are included, and discussed further in Chapter 8.

As mentioned in Section 6.3, using the micro-average causes the precision, recall, and F-score to be the same as the accuracy. The scores in this chapter are computed using the macro-average, as described in Section 2.5.2. The micro-averaged scores can still be seen by looking at the accuracy.

Result from the AAD 10-fold cross validation using all features and the `SVC` classifier

Correct S	1041
Correct NS	7814
Incorrect S	186
Incorrect NS	959
Precision	0.87
Recall	0.75
F_1-score	0.79
Accuracy	0.89

Table 7.1.: Evaluation metrics and raw classification numbers for the AAD 10-fold cross validation using all features and the `SVC` classifier.

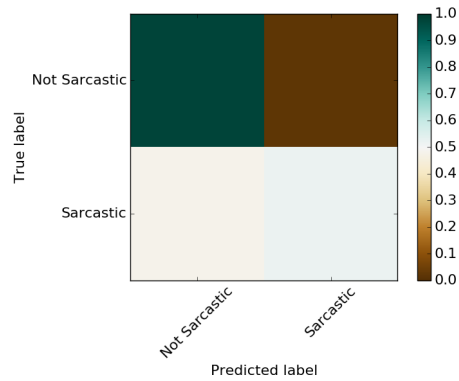


Figure 7.1.: Normalized confusion matrix for the AAD 10-fold cross validation using all features and the `SVC` classifier.

Almost none of the non-sarcastic tweets are misclassified, but almost half of the sarcastic tweets are. The system is conservative, i.e., the classification leans towards the biggest class. This result is the best for the AAD cross validation.

Result from the AAD 10-fold cross validation using lexical features and the `SVC` classifier

Correct S	808
Correct NS	7777
Incorrect S	223
Incorrect NS	1192
Precision	0.83
Recall	0.67
F_1-score	0.72
Accuracy	0.86

Table 7.2.: Evaluation metrics and raw classification numbers for the AAD 10-fold cross validation using lexical features and the `SVC` classifier.

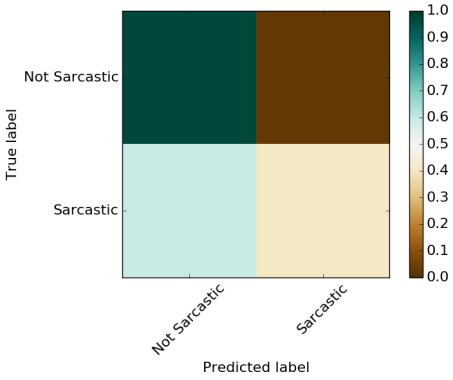


Figure 7.2.: Normalized confusion matrix for the AAD 10-fold cross validation using lexical features and the `SVC` classifier.

When using only the lexical features, the results worsen slightly, but the general classification distribution is the same as with all features.

Result from the AAD 10-fold cross validation using sentiment features and the `SVC` classifier

Correct S	98
Correct NS	7957
Incorrect S	43
Incorrect NS	1902
Precision	0.75
Recall	0.52
F_1-score	0.49
Accuracy	0.81

Table 7.3.: Evaluation metrics and raw classification numbers for the AAD 10-fold cross validation using sentiment features and the `SVC` classifier.

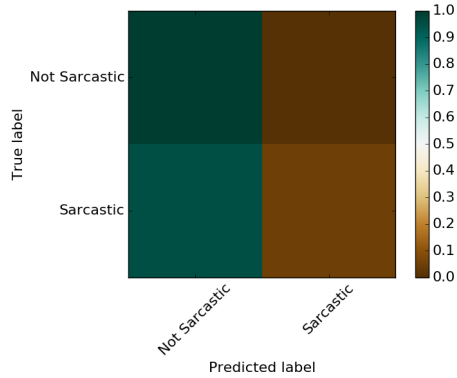


Figure 7.3.: Normalized confusion matrix for the AAD 10-fold cross validation using sentiment features and the `SVC` classifier.

With only sentiment features, very few tweets as classified as sarcastic, and the performance becomes very close to that of the most-frequent classifier.

Result from the AAD 10-fold cross validation using all features and the `LogisticRegression` classifier

Correct S	801
Correct NS	7648
Incorrect S	352
Incorrect NS	1198
Precision	0.78
Recall	0.68
F_1-score	0.71
Accuracy	0.84

Table 7.4.: Evaluation metrics and raw classification numbers for the AAD 10-fold cross validation using all features and the `LogisticRegression` classifier.

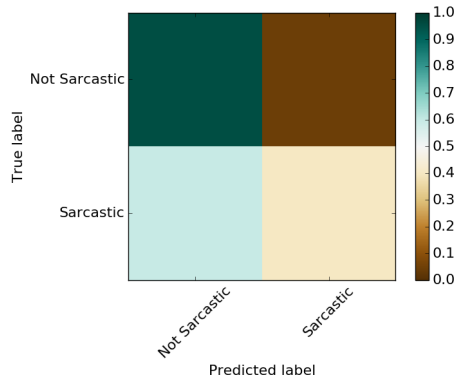


Figure 7.4.: Normalized confusion matrix for the AAD 10-fold cross validation using all features and the `LogisticRegression` classifier.

When the `LogisticRegression` classifier is used with all features, the result is still conservative, like with `SVC`. However, the result is a bit worse.

Result from the AAD 10-fold cross validation using lexical features and the `LogisticRegression` classifier

Correct S	753
Correct NS	7652
Incorrect S	348
Incorrect NS	1247
Precision	0.77
Recall	0.67
F_1-score	0.70
Accuracy	0.84

Table 7.5.: Evaluation metrics and raw classification numbers for the AAD 10-fold cross validation using lexical features and the `LogisticRegression` classifier.

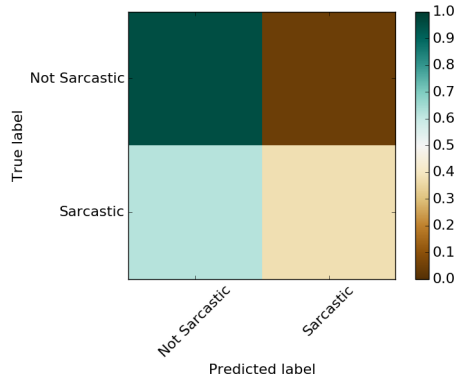


Figure 7.5.: Normalized confusion matrix for the AAD 10-fold cross validation using lexical features and the `LogisticRegression` classifier.

With lexical features only, the result changes very little, but becomes slightly worse.

Result from the AAD 10-fold cross validation using sentiment features and the `LogisticRegression` classifier

Correct S	123
Correct NS	7853
Incorrect S	147
Incorrect NS	1877
Precision	0.63
Recall	0.52
F_1-score	0.50
Accuracy	0.80

Table 7.6.: Evaluation metrics and raw classification numbers for the AAD 10-fold cross validation using sentiment features and the `LogisticRegression` classifier.

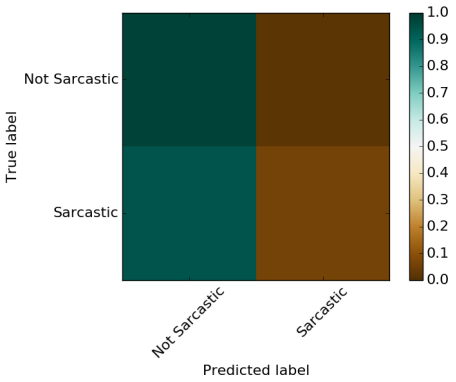


Figure 7.6.: Normalized confusion matrix for the AAD 10-fold cross validation using sentiment features and the `LogisticRegression` classifier.

Like with the `SVC` classifier, very few tweets are classified as sarcastic when only using sentiment features. Again, the behaviour is very close to the most-frequent classifier.

Result from testing with MAD1 using all features and the `SVC` classifier

Correct S	145
Correct NS	1458
Incorrect S	199
Incorrect NS	314
Precision	0.62
Recall	0.60
F_1-score	0.61
Accuracy	0.76

Table 7.7.: Evaluation metrics and raw classification numbers from testing with MAD1 using all features and the `SVC` classifier.

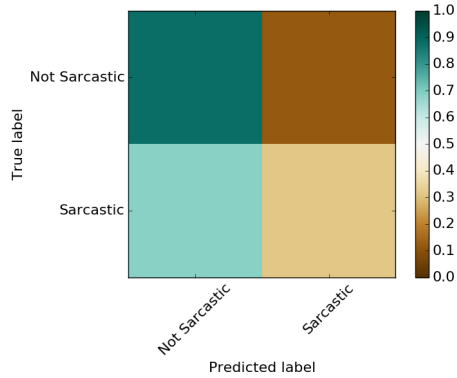


Figure 7.7.: Normalized confusion matrix for the testing with MAD1 using all features and the `SVC` classifier.

The classification distribution is different from the AAD cross validation. A greater portion of non-sarcastic tweets have been classified as sarcastic. The classifier is still very conservative.

Result from testing with MAD1 using all features and the `LogisticRegression` classifier

Correct S	249
Correct NS	1277
Incorrect S	380
Incorrect NS	210
Precision	0.63
Recall	0.66
F_1-score	0.64
Accuracy	0.72

Table 7.8.: Evaluation metrics and raw classification numbers from testing with MAD1 using all features and the `LogisticRegression` classifier.

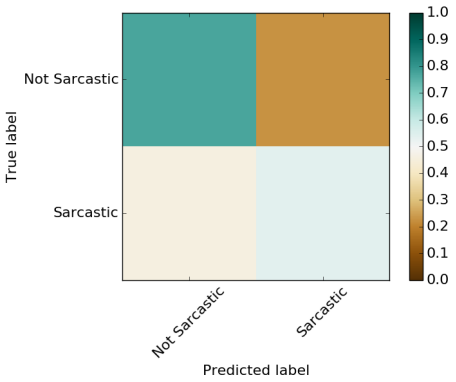


Figure 7.8.: Normalized confusion matrix for the testing with MAD1 using all features and the `LogisticRegression` classifier.

Like for the `SVC` classifier, the classification distribution changes a lot when testing with MAD1. More non-sarcastic tweets have been incorrectly classified as sarcastic, but unlike the `SVC` classifier, the proportion of correctly classified sarcastic tweets has increased.

Result from testing with MAD1-auto using all features and the `SVC` classifier

Correct S	274
Correct NS	999
Incorrect S	70
Incorrect NS	773
Precision	0.68
Recall	0.60
F_1-score	0.55
Accuracy	0.60

Table 7.9.: Evaluation metrics and raw classification numbers from testing with MAD1-auto using all features and the `SVC` classifier.

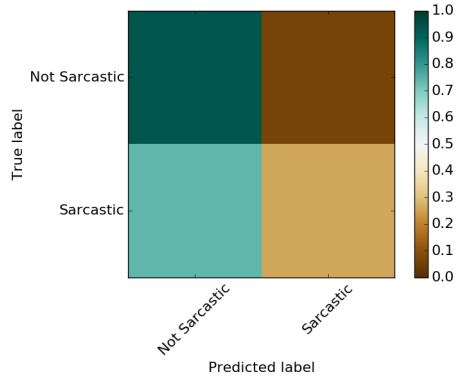


Figure 7.9.: Normalized confusion matrix for the testing with MAD1-auto using all features and the `SVC` classifier.

A greater portion of the sarcastic tweets have been classified as non-sarcastic compared to the AAD cross validation results. Other than that the distribution is similar.

Result from testing with MAD1-auto using all features and the `LogisticRegression` classifier

Correct S	475
Correct NS	917
Incorrect S	152
Incorrect NS	572
Precision	0.69
Recall	0.66
F_1-score	0.64
Accuracy	0.66

Table 7.10.: Evaluation metrics and raw classification numbers from testing with MAD1-auto using all features and the `LogisticRegression` classifier.

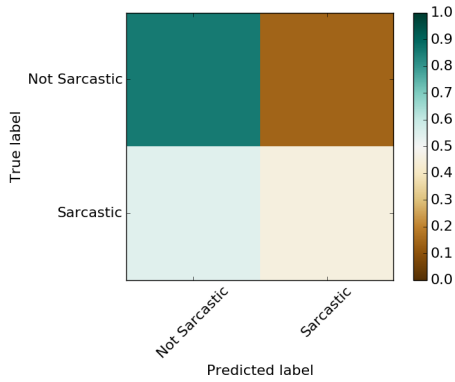


Figure 7.10.: Normalized confusion matrix from testing with MAD1-auto using all features and the `LogisticRegression` classifier.

The result is closer to the AAD cross validation than the MAD1 result is. It is also less conservative than the AAD result.

Result from testing with MAD2 using all features and the `SVC` classifier

Correct S	169
Correct NS	1494
Incorrect S	157
Incorrect NS	385
Precision	0.66
Recall	0.60
F_1-score	0.62
Accuracy	0.75

Table 7.11.: Evaluation metrics and raw classification numbers from testing with MAD2 using all features and the `SVC` classifier.

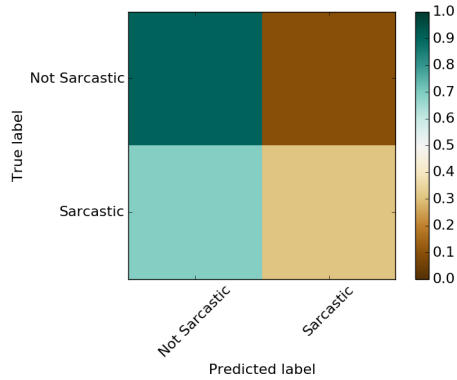


Figure 7.11.: Normalized confusion matrix for the testing with MAD2 using all features and the `SVC` classifier.

The result is very similar to the `SVC` MAD1 result.

Result from testing with MAD2 using all features and the `LogisticRegression` classifier

Correct S	307
Correct NS	1310
Incorrect S	341
Incorrect NS	247
Precision	0.66
Recall	0.67
F_1-score	0.66
Accuracy	0.73

Table 7.12.: Evaluation metrics and raw classification numbers from testing with MAD2 using all features and the `LogisticRegression` classifier.

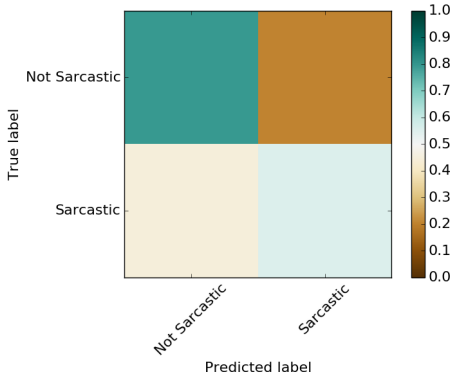


Figure 7.12.: Normalized confusion matrix from testing with MAD2 using all features and the `LogisticRegression` classifier.

The result is very similar to the `LogisticRegression` MAD1 result.

Result from testing with MAD2-auto using all features and the `SVC` classifier

Correct S	282
Correct NS	1046
Incorrect S	44
Incorrect NS	833
Precision	0.71
Recall	0.61
F_1-score	0.55
Accuracy	0.60

Table 7.13.: Evaluation metrics and raw classification numbers from testing with MAD2-auto using all features and the `SVC` classifier.

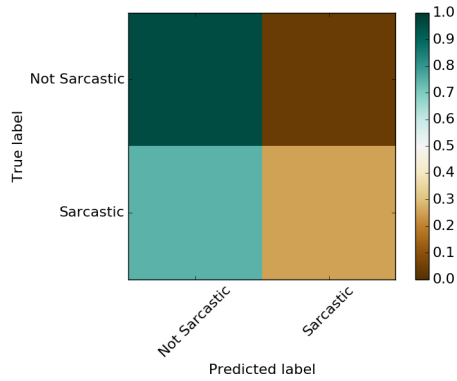


Figure 7.13.: Normalized confusion matrix for the testing with MAD2-auto using all features and the `SVC` classifier.

The result is very similar to the `SVC` MAD1-auto result.

Result from testing with MAD2-auto using all features and the `LogisticRegression` classifier

Correct S	509
Correct NS	950
Incorrect S	140
Incorrect NS	606
Precision	0.70
Recall	0.66
F_1-score	0.65
Accuracy	0.66

Table 7.14.: Evaluation metrics and raw classification numbers from testing with MAD2-auto using all features and the `LogisticRegression` classifier.

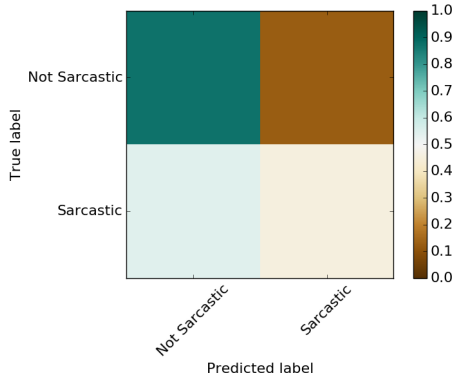


Figure 7.14.: Normalized confusion matrix from testing with MAD2-auto using all features and the `LogisticRegression` classifier.

The result is very similar to the `LogisticRegression` MAD1-auto result.

Tables 7.15 and 7.16 summarize the results for the experiments using all features. The numbers show the classification distribution in percentages.

	AAD	**MAD1**	**MAD2**	**MAD1-auto**	**MAD2-auto**
Correct S	52%	32%	31%	26%	25%
Correct NS	98%	88%	90%	93%	96%
Incorrect S	2%	12%	10%	7%	4%
Incorrect NS	48%	68%	69%	74%	75%

Table 7.15.: The classification of each class of tweet in percentages, using all features and the `SVC` classifier.

	AAD	**MAD1**	**MAD2**	**MAD1-auto**	**MAD2-auto**
Correct S	40%	54%	55%	45%	46%
Correct NS	96%	77%	79%	86%	87%
Incorrect S	4%	23%	21%	14%	13%
Incorrect NS	60%	46%	45%	55%	54%

Table 7.16.: The classification of each class of tweet in percentages, using all features and the `LogisticRegression` classifier.

A comparison between the baselines and the results for `SVC` and `LogisticRegression` with all features is given in Table 7.17.

	Precision	Recall	F_1-score	Accuracy
AAD				
Random	0.50	0.50	0.45	0.50
Most-Frequent	0.40	0.50	0.44	0.80
SVC	**0.87**	**0.75**	**0.79**	**0.89**
LogisticRegression	0.78	0.68	0.71	0.84
MAD1				
Random	0.50	0.50	0.46	0.50
Most-Frequent	0.39	0.50	0.44	**0.78**
SVC	0.62	0.60	0.61	0.76
LogisticRegression	**0.63**	**0.66**	**0.64**	0.72
MAD1-auto				
Random	0.50	0.50	0.50	0.50
Most-Frequent	0.25	0.50	0.34	0.51
SVC	0.68	0.60	0.55	0.60
LogisticRegression	**0.69**	**0.66**	**0.64**	**0.66**
MAD2				
Random	0.50	0.50	0.47	0.50
Most-Frequent	0.37	0.50	0.43	**0.75**
SVC	**0.66**	0.60	0.62	**0.75**
LogisticRegression	**0.66**	**0.67**	**0.66**	0.73
MAD2-auto				
Random	0.50	0.50	0.50	0.50
Most-Frequent	0.25	0.50	0.34	0.51
SVC	**0.71**	0.61	0.55	0.60
LogisticRegression	0.70	**0.66**	**0.65**	**0.66**

Table 7.17.: System performance compared to theoretical baselines for the different data sets.

8. Discussion and Conclusion

This chapter discusses the results obtained from the experiments and looks at some interesting observations made. Some topics for future work are suggested at the end.

8.1. Discussion

The average inter-annotator agreement score for MAD2, as given by Cohen's kappa, was 0.63. This value falls between two values reported by others; Riloff et al. [2013] report 0.81 and Ptáček et al. [2014] report 0.54. The number of tweets annotated, tweet language, and the instructions given to the annotators vary for these three results, but sarcasm generally appears difficult to agree upon. As long as humans don't agree, how well a system performs is somewhat subjective. It is interesting to note that MAD1 and MAD2 have a very similar sarcastic vs. non-sarcastic distribution. In both of them, slightly more than half the tweets containing #sarcasm are either not sarcastic or need context to be identified by the annotators, see Table 4.2. Creating datasets by automatically annotating tweets based on the presence of #sarcasm seems inaccurate.

Upon a rough inspection of the AAD, it was discovered that it contained a few duplicates, and a few foreign language tweets. These should ideally have been removed. There were also numerous elongated words and spelling errors. Shortening the elongated words and correcting the spelling would have led to each word being more accurately represented during the unigram dictionary creation. It would also have helped POS-tagging. This could be taken even further by doing stemming and lemmatization. Mentions (see Section 2.1) should also have been removed from the tweets, as they are disruptive for the POS-tag features.

For the results using just the sentiment features (and AAD cross validation), only a small amount of tweets are classified as sarcastic; the behaviour is thus very close to the most-frequent classifier. This shows that the sentiment features are quite bad at separating sarcastic and non-sarcastic tweets from each other. The sentiment features used in this system have very naïve implementations. More robust implementations are required to take advantage of sentiment information. The results that use just lexical features perform much better, being only slightly worse than using all features together. Both the `SVC` and the `LogisticRegression` classifiers perform best when using all the features, with the `SVC` classifier being better, see Tables 7.1 and 7.4. Both classifiers are conservative, classifying a large portion of the smaller class as the larger class. This implies that the features are mediocre at separating the sarcastic and non-sarcastic tweets.

When tested with MAD1 and MAD1-auto, the performance for the `SVC` classifier drops compared to the AAD results. For MAD1-auto, the change is mainly that a larger portion of the sarcastic tweets are classified as non-sarcastic, as can be seen in Table 7.15. If a lot of the chosen unigrams derived from the training data don't exist in MAD1-auto, it could explain this behaviour. For MAD1, the biggest difference is an increase in non-sarcastic tweets being classified as sarcastic. These tweets might have originally contained #sarcasm, but then been manually annotated as not sarcastic. It is not surprising that the system, which is only trained on an automatically annotated set, would pick up some of these as sarcastic. When using the `LogisticRegression` classifier, the drop in performance is not as drastic as with `SVC`, see Table 7.16. Logistic regression assigns weights to the features, and it seems like it has emphasized features which are not so specific to AAD. For MAD1, the biggest difference is still that non-sarcastic tweets are classified as sarcastic at a much higher rate than for AAD. Both classifiers indentify a higher percentage of sarcastic tweets for MAD1 than for MAD1-auto. This might be because the MAD1 contains more stereotypical sarcasm than MAD1-auto, which makes them easier to group together. The discussion for MAD2 and MAD2-auto is the exact same as for MAD1 and MAD1-auto because the results are almost identical, which can be seen in Tables 7.15 and 7.16. This is an interesting result in and of itself, as it implies that both MAD1 and MAD2 are well annotated and reasonably representative.

Comparing the results for MAD1 and MAD2 with the baselines, as done in Table 7.17, we see that both `SVC` and `LogisticRegression` score better in all metrics than the random classifier. Compared to the most-frequent classifier they score better in precision, recall, and F-score, but tie or get beaten in accuracy. Cliche [2014] and Forslid and Wikén [2015] use similar data for training, and report F-scores of 60% and 71% respectively, but the test data is different so it's difficult to compare directly. Riloff et al. [2013], Joshi et al. [2015], and Khattri et al. [2015] used MAD1 for testing (but with more of the tweets available for download, see Section 4.1), and report F-scores of 51%, 61%, and 88% respectively. The best F-score for MAD1 from the results is 64%, given by the `LogisticRegression` classifier.

While the accuracy for MAD1 and MAD2 fail to beat the most-frequent baseline, this is not the case for MAD1-auto and MAD2-auto. The results for these beat the baseline accuracy by 9% for `SVC` and 15% for `LogisticRegression`. The baselines are also beaten in the other metrics. As mentioned above, since the system is trained on an automatically annotated set, it isn't surprising that it is more accurate for automatically annotated datasets.

8.2. Conclusion

In this thesis, a dataset (MAD2) of tweets manually annotated with respect to the presence of sarcasm was built. The result was very similar to that of a previously made set (MAD1), and both of them showed considerable deviation from the automatic annotation. This implies that using automatically annotated data for the task of sarcasm detection in tweets is a mediocre approximation.

Experiments with both of the manually annotated datasets yield very similar results, even though the sets are completely independent. This shows that the sets are well annotated and reasonably representative for sarcasm detection in tweets.

When experimenting with different algorithms and features, it became evident that a naïve implementation of sentiment features is not sufficient, allthough it did slightly improve overall performance.

The experiments also showed that with the chosen features, both the

SVM and Logistic Regression algorithms beat theoretical baseline F-scores by 13%-23% when tested with previously unseen manually annotated data. They also beat F-scores from some of other work using the same data by 3%-13%. However, the best score from this thesis (for MAD1) is outperformed by 24% when compared with the work of [Khattri et al., 2015], who make use of user data.

In Section 1.1, two goals were stated for this thesis. While both goals were accomplished in principle, more work could certainly have been done. For the first goal, a state-of-the-art study has been performed, and a system has been implemented, although the approach was not very novel. For the second goal, several features and algorithms were tested on both automatically and manually annotated data. However, the differences in system performance, especially between MAD1/2 and MAD1/2-auto, could have been investigated more.

8.3. Future Work

After the annotation of MAD2, the annotators mentioned that context would have helped a lot with some of the tweets. This is not surprising, as sarcasm is rarely used in a vacuum, but rather in response to some occurrence or utterance. Taking into account the context of a tweet, like other tweets about the same topic, or other tweets from the same feed, is a natural next step in the quest for an automatic sarcasm detection system. One could also account for context by using information about the author. Another thing brought up by the annotators was the issue of domain knowledge. Some tweets would refer to people or incidents that they had no knowledge about, which made it difficult to judge them. Classifying tweets by topic and then using domain knowledge is another area to explore. This could be of particular interest when dealing with tweets about current topics and news stories. Using domain knowledge is also a way of looking at the context.

An issue that has been brought up by some, e.g., Liebrecht et al. [2013], is that the amount of sarcasm on Twitter is far less than the amount typically used in training data. The sarcastic tweets used for MAD2 were taken from a set of about 103 million tweets, out of which roughly 2000 had the sarcasm hashtag. Not all sarcastic tweets have this hashtag, but

it gives some impression of the difference between real data and training data. Even a rather conservative system like the one presented in this thesis would almost certainly have a poor precision score when dealing with realistically distributed data. Finding a way to deal with this issue is a topic to consider for future work.

In the introduction it was mentioned that sarcasm plays a role in sentiment analysis of tweets. Exactly what role it plays and how to incorporate it is yet another path to explore.

Finally, two straightforward improvements of the system presented here would be the use of manually annotated data for testing, and better preprocessing. In regards to the latter, the treatment of slight variations of the same words (e.g., elongated or misspelled) would probably greatly benefit the unigram selection process.

Bibliography

David Bamman and Noah A. Smith. Contextualized Sarcasm Detection on Twitter. In *Proceedings of the 9th International Conference on Web and Social Media*, pages 574–577, Oxford, United Kingdom, 2015.

Francesco Barbieri, Francesco Ronzano, and Horacio Saggion. UPF-taln: SemEval 2015 Task 10 and 11 Sentiment Analysis of Literal and Figurative Language in Twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 704–708, Denver, Colorado, 2015.

Steven Bird, Edward Loper, and Ewan Klein. *Natural Language Processing with Python*. O'Reilly Media Inc., 2009.

Paula Carvalho, Luís Sarmento, Mário J. Silva, and Eugénio de Oliveira. Clues for Detecting Irony in User-Generated Contents: Oh...!! it's "so easy" ;-). In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 53–56, Hong Kong, China, 2009.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011. (note: first version published in 2001).

Mathieu Cliche. The Sarcasm Detector. `http://www.thesarcasmdetector.com/about/`, 2014. [Online; accessed 29-May-2016].

Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, 20:273–297, 1995.

Daily Mail. Frustrated air passenger arrested under Terrorism Act after Twitter joke about bombing airpot. `http://www.dailymail.co.uk/news/article-1244091/Man-arreste`

d-Twitter-joke-bombing-airport-Terrorism-Act.html, 2010. [Online; accessed 29-May-2016].

Dmitry Davidov, Oren Tsur, and Ari Rappoport. Semi-Supervised Recognition of Sarcastic Sentences in Twitter and Amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 107–116, Uppsala, Sweden, 2010.

DHS. Computer Based Annual Social Media Analytics Subscription. https://www.fbo.gov/?s=opportunity&mode=form&id=8aaf9a50dd4558899b0df22abc31d30e&tab=core&_cview=0, 2014. [Online; accessed 29-May-2016].

Facebook. Company Information. http://newsroom.fb.com/company-info/, 2016. [Online; accessed 29-May-2016].

Erik Forslid and Niklas Wikén. Automatic irony- and sarcasm detection in Social media. Master's thesis, Uppsala Universitet, 2015.

Valerij Fredriksen and Brage Ekroll Jahren. Twitter Sentiment Analysis: Exploring Automatic Creation of Sentiment Lexica. Master's thesis, Norwegian University of Science and Technology, June 2016.

Aniruddha Ghosh, Guofu Li, Tony Veale, Paolo Rosso, Ekaterina Shutova, Antonio Reyes, and John Barnden. SemEval-2015 Task 11: Sentiment Analysis of Figurative Language in Twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 470–478, Denver, Colorado, 2015a.

Debanjan Ghosh, Weiwei Guo, and Smaranda Muresan. Sarcastic or Not: Word Embeddings to Predict the Literal or Sarcastic Meaning of Words. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1003–1012, Lisbon, Portugal, 2015b.

Roberto González-Ibáñez, Smaranda Muresan, and Nina Wacholder. Identifying Sarcasm in Twitter: A Closer Look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics:shortpapers*, pages 581–586, Portland, Oregon, 2011.

Yanfen Hao and Tony Veale. An Ironic Fist in a Velvet Glove: Creative Mis-Representation in the Construction of Ironic Similes. *Minds and Machines*, 20(4):635–650, 2010.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. LT3: Sentiment Analysis of Figurative Tweets: piece of cake #NotReally. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 684–688, Denver, Colorado, 2015.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. Harnessing Context Incongruity for Sarcasm Detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 757–762, Beijing, China, 2015.

Maria Karanasou, Christos Doulkeridis, and Maria Halkidi. DsUniPi: An SVM-based for Sentiment Analysis of Figurative Language on Twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 709–713, Denver, Colorado, 2015.

Anupam Khattri, Aditya Joshi, Pushpak Bhattacharyya, and Mark James Carman. *Your Sentiment Precedes You*: Using an author's historical tweets to predict sarcsm. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA 2015)*, pages 25–30, Lisbon, Portugal, 2015.

Christine Liebrecht, Florian Kunneman, and Antal van den Bosch. The perfect solution for detecting sarcasm in tweets #not. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA 2013)*, pages 29–37, Atlanta, Georgia (USA), 2013.

Christopher D. Manning. Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics? In *Computational Linguistics and Intelligent Text Processing*, pages 171–189, Tokyo, Japan, 2011.

Diana Maynard and Mark A. Greenwood. Who cares about sarcastic tweets? Investigating the impact of sarcasm on sentiment analysis. In *Proceedings of the 9th edition of the Language Resources and Evaluation Conference*, pages 4238–4243, Reykjavik, Iceland, 2014.

Sarah McGillion, Héctor Martínez Alonso, and Barbara Plank. CPH: Sentiment analysis of Figurative Language on Twitter #easypeasy #not. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 699–703, Denver, Colorado, 2015.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval-2013)*, pages 321–327, Atlanta, Georgia (USA), 2013.

Kjersti Nipen, June Westerveld, and Fredrik Hager-Thoresen. Mor blir forstått og unnskyldt i retten. *Aftenposten*, 1:4–6, February 1st 2016.

Canberk Özdemir and Sabine Bergler. ClaC-SentiPipe: SemEval2015 Subtasks 10 B,E, and Task 11. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 479–485, Denver, Colorado, 2015.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Tomáš Ptáček, Ivan Habernal, and Jun Hong. Sarcasm Detection on Czech and English Twitter. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 213–223, Dublin, Ireland, 2014.

Finn Årup Nielsen. A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. In *Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages*, pages 93–98, Heraklion, Crete, 2011.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. Sarcasm as Contrast between a Positive Sentiment and Negative Situation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, Seattle, Washington, 2013.

Drahomíra Spoustová, Jan Hajič, Jan Raab, and Miroslav Spousta. Semi-supervised training for the averaged perceptron POS tagger. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 763–771, Athens, Greece, March 2009. ACL.

Twitter. Company Information. `https://about.twitter.com/company`, 2016. [Online; accessed 29-May-2016].

C. J. van Rijsbergen. *Information Retrieval*. Butterworths, 1975.

Hao Wang and Jorge A. Castanon. Sentiment Expression via Emoticons on Social Media. In *IEEE International Conference on Big Data (Big Data 2015)*, pages 2404–2408, Santa Clara, California, 2015.

Hongzhi Xu, Enrico Santus, Anna Laszlo, and Chu-Ren Huang. LLT-PolyU: Identifying Sentiment Intensity in Ironic Tweets. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 673–678, Denver, Colorado, 2015.

A. List of Emoticons Converted to ASCII

Unicode	ASCII
U+263A	:)
U+1F494	</3
U+1F495	<3
U+1F496	<3
U+1F497	<3
U+1F498	<3
U+1F499	<3
U+1F49A	<3
U+1F49B	<3
U+1F49C	<3
U+1F49D	<3
U+1F49E	<3
U+1F49F	<3
U+1F600	:D
U+1F601	:D
U+1F602	:D
U+1F603	:)
U+1F604	:D
U+1F605	:D
U+1F606	:D
U+1F607	:)
U+1F608	:=)
U+1F609	;)
U+1F60A	:)

Unicode	ASCII
U+1F60B	;p
U+1F60C	:)
U+1F60D	:*
U+1F60E	:)
U+1F60F	;)
U+1F610	:|
U+1F611	−_−
U+1F612	:(
U+1F613	:(
U+1F614	:(
U+1F615	:/
U+1F616	:$
U+1F617	:*
U+1F618	:*
U+1F619	:*
U+1F61A	:*
U+1F61B	:P
U+1F61C	;P
U+1F61D	:P
U+1F61E	:(
U+1F61F	:(
U+1F620	>:(
U+1F621	:@
U+1F622	:'(

Unicode	ASCII
U+1F623	:(
U+1F624	:(
U+1F625	:(
U+1F626	D:
U+1F627	D:
U+1F628	D:
U+1F629	D:
U+1F62A	:/
U+1F62B	:S
U+1F62C	:|
U+1F62D	:(
U+1F62E	:O
U+1F62F	:O
U+1F630	:(
U+1F631	>:O
U+1F632	:X
U+1F633	:S
U+1F634	−_−
U+1F635	:X
U+1F636	:X
U+1F637	:X