UPPSALA
UNIVERSITET

UPPTEC F 15045
Examensarbete 30 hp
Augusti 2015

Automatic irony- and sarcasm detection in Social media

Erik Forslid
Niklas Wikén

Abstract

Automatic irony- and sarcasm detection in Social media

Erik Forslid, Niklas Wikén

This thesis looks at different methods that have been used for irony and sarcasm detection and also includes the design and programming of a machine learning model that classifies text as sarcastic or non-sarcastic. This is done with supervised learning. Two different data set where used, one with Amazon reviews and one from Twitter. An accuracy of 87% was obtained on the Amazon data with the Support Vector Machine. For the Twitter data was an accuracy of 71% obtained with the Adaboost classifier was used. The thesis is done in collaboration with Gavagai AB, which is company working with Big-data text with expertise in semantic analysis and opinion mining.

Handledare: Magnus Sahlgren
Ämnesgranskare: Michael Ashcroft
Examinator: Tomas NYberg
ISSN: 1401-5757, UPTEC F15 045

Contents

1 Introduction

As we are entering the era of Big data there is an increasing amount of data being stored and processed every day, containing everything from customers shopping behavior to how different medicines act on patients. Handling this amount of data calls for an automated tool for data analysis, which is what Machine Learning provides.

Already at the introduction of the computer people wondered if we could make computers learn from experience. Machine learning is a field that grew out from Artificial Intelligence, as an academic interest in teaching machines to learn from data [Hosch, 2013]. As early as in 1959 Arthur Samuel, a pioneer in the field of artificial intelligence, defined machine learning as: *"The field of study that gives computers the ability to learn without being explicitly programmed"*.

In the past decade machine learning has given us various new things, like voice recognition, self-parking cars and improved web search engines. The idea with Machine Learning is that the system learn from previous information and make predictive analysis that finds the searched result.

1.1 Application domains

Machine learning can be applied to many different applications. Below we describe, briefly, three areas where machine learning is used successfully.

1.1.1 Spam filter

You might have not thought about it, but the spam-filter in your email client is most likely based on a Machine Learning algorithm. By scanning the context of the email it may predict if an incoming mail is spam or not with high accuracy. Table 1 is an example of the most different features in a spam classification example from the book *The Elements of statistical learning* [Hastie et al., 2010]. A simple rule for determining

Table 1: Biggest difference between features in a spam filter.

	george	you	your	hp	free	hpl	!	our	re	edu	remove
spam	0.00	2.26	1.38	0.02	0.52	0.01	0.51	0.51	0.13	0.01	0.28
email	1.27	1.27	0.44	0.90	0.07	0.43	0.11	0.18	0.42	0.29	0.01

whether an incoming mail is spam or not could be

```
if $free \geq 0.30$ and $your \geq 0.91$
    spam
else :
    not spam.
```

In general rules are much bigger than this and use more complex queries than just Boolean expressions. In the case of spam filters, it is essential not to classify normal emails as spam since it might get lost. The opposite, classifying spam as normal mail, is a minor mistake. Removing an email manually is easy.

1.1.2 Classifying flowers

Already in 1936, the statistician, Ronald A. Fisher tried to separate three different iris flowers by using discriminant analysis from iris data [Fisher, 1936] [1]. The famous data set is commonly known today as Fisher's iris data and is still one of the best data sets available for educational purposes in machine learning. The data consists of measurements of the petal and sepal length and width from 50 instances each from the irises Setosa, Versicolor and Virginica. From the scatter plots in Figure 1 we can see that Setosa is linearly separable just by examining the feature petal length. Separating the other two classes is more difficult and requires more features than just one to separate the classes properly. [2]

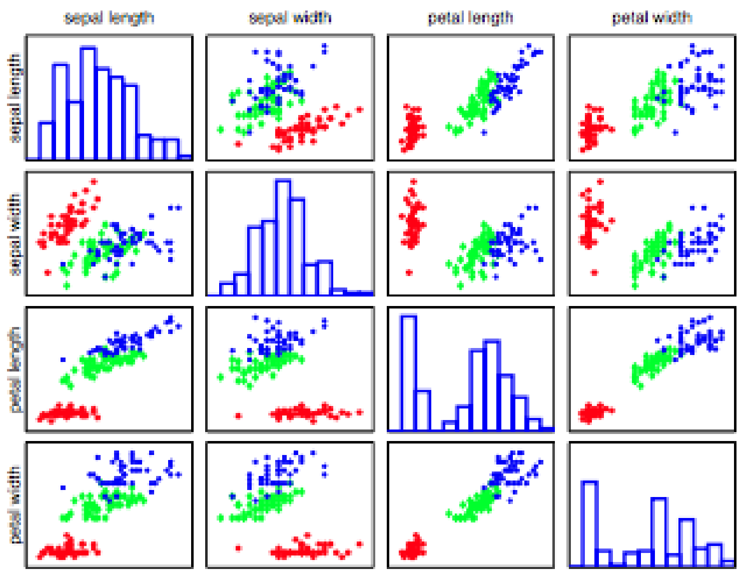


Figure 1: Red = Setosa, Green = Versicolor, Blue = Virginica.

1.1.3 Handwritten recognition

Finally a more complex problem than the former ones, especially since data has not been pre-processed by humans. Let us consider the special scenario where handwritten digits are isolated from each other. This is normally the situation when scanning e.g. ZIP codes. Data for this purpose can be found from Modified National Institute of Standards (MNIST), which database has 60'000 images of digits available for training. The images are of size 28×28 pixels and each pixel's intensity range between 0 and 255.[3]

Also, as for the spam-filter, in the case of a ZIP-code reader a classifier must be certain that a classification is correct, otherwise envelopes will be sent wrongly. A solution for avoiding that scenario is to create an additional class for instances where the classifier cannot decide a digit with a certainty above some threshold. Envelopes in that class can then be sorted manually.

[1] http://www.math.uah.edu/stat/data/Fisher.html
[2] https://github.com/probml/pmtk3/tree/master/demos
[3] http://yann.lecun.com/exdb/mnist/

1.2 Purpose

Gavagai AB is a company operating with Big-data text with expertise in semantic analysis and opinion mining. The large amount of data that they use enables them to provide an accurate understanding of words and topics in several languages. The model that they use is inspired by how the human brain works, which learns text effortlessly simply by reading it in a context. Gavagai's model learns the meaning of a word by observing the context and the usage of words. This is done continuously so that Gavagai's model will quickly learn if a word is changing in meaning or if a new word is created.

At the moment, Gavagai takes no account of irony. Thus this master thesis is a pre-study for building a model that automatically can detect irony and sarcasm.

1.3 Goal

The main goals with this project are:

- To identify different patterns in text that can give an indication of irony.
- Build a model that classifies new not previously seen documents with an accuracy statistically higher than the proposed baseline.
- An important subgoal we have is to get hold of good quality data that enables us to reach our main goal.

1.4 Delimitations

This study will only concern irony in the English language. What is perceived as irony and sarcasm can differ much depending on cultural background. Also, differences in in language may differ in how irony is expressed. Therefore, the model will only be trained and tested on English texts.

2 Irony Detection

One event that got much attention this year was when the comedian Ryan Hand posted the tweet: *"What a disgrace, there's a woman crying at the @Ryanair check in desk who's been made to pay more for emotional baggage."* To which Ryanair responded two minutes later: *"Hi Ryan, which airport is this happening at? IK"*. This conversation was retweeted by thousands of other users before the mistake was discovered and a spokesman from Ryanair said: *"We apologize for temporary technical difficulties with our sarcasm detector."*

Research has showed that kids in an early age are able to understand sarcasm. In an experiment conducted at the University of Calgary, showed that 5 year old children could easily pick up simple forms of sarcastic utterances [Pexman et al., 2013]. But even though most people are skillful in detecting irony in real world situations, irony in text is something different.

Unlike topic detection e.g. separating articles from different sports, irony detection is considered as a very difficult task. In the case of classifying sports, each sport (topic) normally has its own vocabulary that is unlikely to appear in any other document than that sport. This vocabulary is often enough for making an accurate classifier. However, irony detection is considered as a very difficult task due to that it appears in various forms and in all kinds of contexts. Irony often has ambiguous interpretations and it often expresses the contrary to what is literally being said [Butler, 1953]. Another reason why irony could be more difficult to detect in written text than in spoken conversation is e.g. due to that information contained in the tone of the voice or facial expressions that can be important for understanding irony are lost. Yet another theory is that it takes longer to write on a computer than talking face-to-face, and that people use that extra time to write more complex sarcasm than what they use normally [Chin, 2011].

2.1 Why is irony detection interesting?

Except for being an interesting field for human-computer interaction irony detection is also an important area in sentiment analysis and opinion mining. Understanding irony in text would enable a more accurate picture of what is requested [Carvalho et al., 2009].

In medical care sarcasm detection could work as detection for brain injuries in an early stage. In research done at the University College of London they were able to show that people suffering brain injuries had impaired sarcasm comprehension compared to the control group [Channon et al., 2004].

Another important area is for security reasons to evaluate whether a threat is real or not. The consequences could be devastating if some ironic posts were to be taken seriously. E.g. the humorous twitter account *dailykisev* tweeted *"And 12 bombers depart now ... to Lithuania!"* after it turned out that Lithuania gave Russia zero points for its performance in the Eurovision Song Contest 2015.

U.S. secret service also said that they are starting to work on a sarcasm detector for twitter. Even though experts in the area express skepticism towards the idea, it is considerable that at least two individuals have been falsely arrested as a consequence of sarcastic posts during the past five years [Kearns, 2013, Silver, 2014].

2.2 Definition of irony and sarcasm

Irony is a sophisticated form of language use that acknowledges a gap between the intended meaning and the literal meaning of the words. Even though it is a widely studied linguistic phenomenon no clear definition seems to exist [Filatova, 2012]. Irony can be divided into two broad categories: situational and verbal irony. The former one is e.g. a cigarette company having non-smoking signs in the lobby. The latter one, which is considered in this master thesis, is most commonly defined as ”saying the opposite of what you mean” [Butler, 1953] where the difference between the saying and meaning is supposed to be clear. Other definitions states that it is any form of negation without any markers [Giora, 1995], another one says that irony violates the maxim of not saying what you believe is false [Grice, 1975]. Yet another definition is that an utterance has to be echoic to be judged as ironic [Sperber and Wilson, 1995].

1. ”thank you Janet Jackson for yet another year of Super Bowl classic rock!”
2. ”He’s with his other woman: XBox 360. It’s 4:30 fool. Sure I can sleep through the gun- fire”
3. ”WOW I feel your interest...?”
4. ”[I] Love The Cover”

These examples demonstrate different situations of irony and sarcasm in different texts. Example (1), (2) and (3) are tweets from Twitter and example (4) is taken from an Amazon review. In example (1) it refers to a very poor performance at the super bowl and refers to the disgraceful performance of Janet Jackson the year before. Example (2) consists of three sentences that are sarcastic on their own, and when combining them the sarcasm becomes obvious. In example (3) an indication of irony is that ”WOW” is spelled with capital letters in combination with the ellipsis at the end of the sentence. The example from Amazon (4) could just as well be taken from a positive book review. However this is taken from a review entitled ”Do not judge the book on its cover”, and with this title it reveals a sarcastic tone.

To summarize, any linguistic, philosophical or psychological approach to capture irony theoretically has, rather than come with a final solution, given different perspectives to the phenomena [Sperber and Wilson, 1995]. It is obvious that there is no simple rule or algorithm that can capture irony. In this study we will look at several aspects (sentiments, vocabulary, linguistics, punctuations etc.) that can play a part in forming ironic and sarcastic content.

2.3 Related Work

From the literature survey conducted for this project three articles that in some way have influenced this project are presented. The first article proposes that irony on twitter is often used to be funny and thus is close in relation to humor. In the second article the authors implement a bootstrap method for finding ironic tweets of the form where a positive verb is followed by a negative situation or event. The third article a semi-supervised approach for finding ironic amazon reviews is used.

2.3.1 A Multidimensional Approach for Detecting Irony in Twitter

In this article the authors build a classifiers for detecting ironic tweets by comparing them to tweets from three different topics. To perform the classification the authors have collected 40'000 tweets from 4 different hashtags(#humor, #irony, #education and #politics). These specific hashtags where used for 3 different reasons. (1) To avoid manual selection of tweets. (2) To allow irony beyond literary definitions because (3) irony hashtags may reflect a general opinion of what constitutes irony and humor. A multidimensional approach, based on 4 different properties, is suggested as a method to detect irony in the tweets.

- Signatures: concerning pointedness, counter-factuality, and temporal compression.
- Unexpectedness: concerning temporal imbalance and contextual imbalance.
- Style: as captured by character-grams, skip-grams, and polarity skip-grams.
- Emotional scenarios: concerning activation, imagery, and pleasantness.

Each property contains several attributes and each attribute represents some aspect of the phenomena to lead a micro-blogger to tag a tweet as ironic or sarcastic. We have been inspired by this project by also looking at a multidimensional approach to capture irony.

2.3.2 Sarcasm as Contrast between a Positive Sentiment and Negative Situation

In this article a bootstrap-algorithm is presented that automatically learns phrases with negative situations, or events, combined with positive sentiments, from twitter data with the hashtag *sarcasm*. This algorithm relies on the assumption that negative situations often appear after positive situations in sarcastic texts. In addition, the negative phrase has to be in the vicinity of the positive phrase.

[positive verb phrase] [negative situation phrase]

The bootstrapping algorithm starts with the seed 'love', which is, due to the authors, a common positive verb in sarcastic tweets. Given that a sarcastic tweet contains the word 'love' they gather *n*-grams succeeding the seed as looking for possible candidates for the negative situation. They select the best candidates by a scoring metric and then add them to their list of negative situations.

In the second step they use the structural assumption in the opposite direction, trying to find negative situations preceded by a positive verb. E.g. in the sarcastic example sentence *I love when it rains.* the algorithm would collect all the succeeding n-grams after 'love' and select the best candidates. Two possible candidates as negative situations could be "when it rains", "rains". Finally, it uses the learned phrases and iterates through a new set of tweets to automatically identify sarcastic tweets. Even though our approach differs a lot from this method, we understood that sentiment analysis is an important aspect when looking at irony. Regarding the conclusions from this article we will try to identify documents, just like in this article, that have high and mixed sentiment values i.e. have a frequency of positive / negative words.

2.3.3 Semi-Supervised Recognition of Sarcastic Sentences in Twitter and Amazon

Tsur et al 2010 focus on classifying sarcastic tweets and amazon product reviews by creating pattern-based features based on content words (CW) and high frequency words (HFW). Each pattern allows 2-6 slots for HFW and 1-6 slots for CWs allowing one sentence to generate multiple patterns. For the sentence ”Garmin apparently does not care much about product quality or customer support”, they generate patterns including: ”[Company] CW does not CW much”, ”does not CW much about CW CW or CW CW”, ”not CW much” and ”about CW CW or CW CW”. Both company names and product names are treated like HFWs. When performing features selection the most general patterns (patterns appearing in almost every review) and the most specific patterns (patterns appearing in just a few reviews) are removed. The remaining patterns are used as features with weights given as following:

1: Exact match - all the pattern components appear in the sentence in correct order without any additional words.

α: Sparse match - same as exact match but additional non-matching words can be inserted between pattern components

$\gamma \cdot n/N$: Incomplete match - only $n > 1$ of N pattern components appear in the sentence, while some non-matching words can be inserted in-between. At least one of the appearing components should be HFW.

0: No match - nothing or only a single pattern component appears in the sentence.

where $0 < \alpha < 1$ and $0 < \gamma < 1$.

3 Data

An important part when working with text classification is getting hold of good quality data, which is difficult in the case of ironic text. It is often necessary to annotate data manually, which is tedious work. In this master thesis we work with two different data sets, one from Amazon and a second one from Twitter. We have mainly focused on the Amazon data throughout this master thesis since we believe that Amazon reviews are more general. However, we will apply the same methods to the Twitter data.

In the rest of this chapter we describe the characteristics of the two data sets and how the data is obtained.

3.1 Amazon

Amazon is retail e-commerce company that sells new, second hand and refurbished items from a broad range of products.

The Amazon corpus consists of product reviews published on www.Amazon.com. Each review consists of a title, a rating (where 1 star is the lowest rating and 5 the highest), a publication date, a product name, an author and a review text. The reviews are limited to 5'000 words. Amazon recommend though that a review text should ideally range between 75 and 500 words and never be shorter than 20 words [Amazon, 2015]. Another recommendation for writing a review is to be sincere, an interesting recommendation, which we believe may be violated when writing an ironic review. Below is an example review about the movie "Mall Cop" from our Amazon corpus:

- *label: ironic*
- *rating: 5.0 stars*
- *review title: Best Movie of 2009*
- *product: Paul Blart: Mall Cop (DVD)*
- *review text: Paul Blart is a mall cop and although he's clumsy at times, he's definitely not a loser. He's an inspiration for security "officers" everywhere. As the drama unfolds, we see that Blart doesn't get the respect he deserves from anyone in the movie. Several times I had to wipe the tears from my eyes as I watched the film. Kevin James is a highly underrated actor in the mold of William Shatner as well as being a hilarious comedian. His role in "Mall Cop" is similar to Jackie Gleason's in "Smokey and the Bandit". If you don't like this movie, you probably hate weight challenged people, and if you hate them, you most likely hate America.*

3.1.1 Data collection

The data has been provided to us by Elena Filatova and are obtained by hiring Mechanical Turks (M-turks)[4]. The instruction given to the M-turks are:

[4]https://www.mturk.com/mturk/welcome

- First review should be ironic or sarcastic. Together with this review you should
    1. cut-and-paste the text snippet(s) from the review that makes this review ironic/sarcastic
    2. select the review type: ironic, sarcastic or both (ironic and sarcastic)
- The other review should be a regular review (neither ironic or sarcastic)

To make sure that the collected data has a high quality, five new M-turks are hired to evaluate the ironic the data set. Each review obtains a vote from three of the new M-turks whether they find the reviews ironic or not, where only reviews that obtain an ironic majority vote are kept. After the voting is done they end up with 437 ironic reviews and 817 regular reviews. Out of these 331 are paired reviews, i.e. reviews of the same product, and 486 (107 ironic and 486 regular) are unpaired reviews. Table 2 shows distributions of the ratings on the reviews showed. Most of the ironic reviews are rated with only 1 star and non-ironic reviews are in general positive.

Table 2: Distribution of the reviews from Amazon.

		Ratings				
		1⋆	2⋆	3⋆	4⋆	5⋆
ironic	437	262	27	20	14	114
regular	817	64	17	35	96	605

3.2 Twitter

Twitter is a micro-blog that allows broadcasting of posts consisting of up to 140 characters and a post is normally referred to as a *Tweet*. It is possible to write directly to other user by mentioning the other users username prefixed with a @ character. That tweet will then become visible also to the mentioned users followers. A tweet may also contain a hashtag prefixing a string of characters with the # character. Users may browse for hashtags, they are therefore normally used for putting a topic to the tweet or commenting on an event etc. Other type of data that are normal in tweets are URLs and pictures. An example of a tweet from our Twitter corpus:

- *label: non-ironic*
- *tweet: I'm at San Mateo County Fair - @<User>(San Mateo, CA) w 20 others <URL>*

An average of 500 million tweets per day. Twitter supports more than 35 different languages with English being the far most popular. 23% of the tweeters are located in the U.S. [Inc, 2015].

3.2.1 Data collection

The Twitter API enables developers to easily collect tweets. There are two different options. Either use a search query and collect all the tweets that match that query. The other option is to create a stream and collect tweets in real-time.

The corpus comprise of tweets collected from the Twitter API stream. The API only stores tweets one week and the maximum number of tweets that can be acquired in one query is 100 tweets. In average we get around 3000 ironic / sarcastic tweets every day. Since this is tedious work we extended our tweets with a collection of tweets collected during the period of June-July 2014 by Matheiu Cliche. All sarcastic tweets are collected by streaming tweets containing the hashtag sarcasm or sarcastic (#sarcasm, #sarcastic) and the non-sarcastic are collected in the same manner but without any restriction of hashtags. In total we have 120'000 tweets where 20'000 are labeled as sarcastic and 100'000 tweets are labeled as non-sarcastic. It is possible, and also very likely, that the non-sarcastic set also contain some sarcastic tweets even though no #sarcasm hashtag has been used. We assume though that the number of sarcastic tweets in the non-sarcastic set is relatively small and thus negligible.

4 Theory / Text Categorization

Text categorization is an area that overlaps both with Natural Language Processing (NLP) and Machine learning. The task is to automatically sort documents under some predefined categories. Text in itself is not amenable for classification, and the first step in text categorization is to process and extract features from the text that are.

This Chapter is devoted to explain methods that can be used in the process of text categorization and will follow the flowchart in Figure 2. In the first step we describe how the text is prepared for extracting features, which is unique for text categorization problems. The remainder steps are common for many types of Machine learning problems.

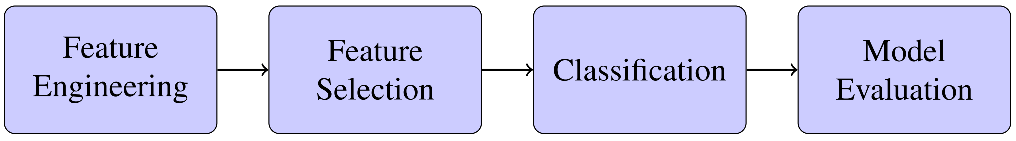


Figure 2: Text categorization process.

4.1 Feature Engineering

Creating features for the model is perhaps the heaviest and most important part and probably the reason why a model succeeds or fails. Unless the data has many independencies correlates well, engineering with the data is necessary. This part of the process is sometimes referred to as a ”black art”, since each classification problem requires a different set of features, and being creative when extracting features can be just as valuable as picking the best classifier [Domingos, 2012].

What type of features that can be extracted depend much on how the text has been processed. A simple method for classifying text is to use counts for each word appearing in the corpus. However, before doing so we may enhance the text using methods presented in this section.

4.1.1 Tokenization

A token is a sequence of characters that we want to treat as a group. Decomposing a text into tokens enables creating counts of tokens, which can be used as features. A token could be a paragraph, a sentence etc. but commonly words are chosen as tokens in text categorization. Example of different tokenizers from the NLTK library can be found on their demo page. [5]

4.1.2 Stemming and Lemmatization

Stemming, together with lemmatization, is a common method in NLP. The goal of the method is to reduce the amount of inflected words by stripping the suffix of the word to retrieve a ”base form” of the word. Stemming is built upon the idea that words with the

[5]http://text-processing.com/demo/tokenize/

same stem are close in meaning and that NLP will be improved if such different words can be grouped together to one single term [Porter, 1980], consequently in Machine learning the number of features can be reduced if stems are used instead of the original words. In the example of the words:

- Argue
- Argued
- Argues
- Arguing

A stemming algorithm could identify the suffixes ”e”, ”ed”, ”es”, ”ing” and strip the words to the stem ”argu”. As the example demonstrate, a stem does not necessarily has to be a word. The suffix stripping algorithm used for providing this example is the Porter Stemmer, from the python library NLTK, based on the algorithm developed by Martin Porter [Porter, 1980].

In linguistics a lemma is the canonical form of a word. For example, the verb ”to work” can appear as ”working”, ”worked”, ”works”. Its canonical form ”work” is a lemma, and the method of converting a word to its lemma is referred to as lemmatization. A lemmatization would for example transform the word ”mice” in to the lemma ”mouse”. Lemmatization is consider as more complex than stemming since lemmatization algorithms has to parse a text first to identify the part of speech of words, due to this fact lemmatization will not be considered in this report.

4.1.3 Part-Of-Speech (POS)-tagging

Another tool that can be used for feature extraction is a POS-tagger. It is one of many interesting tools in stylometry that can be used in Natural language processing. POS-taggers assign each token its word class, i.e. tags the token with its part of speech. A benefit of using a POS-tagger is to distinguish homonyms, which is shown in the example below.

1. Put/VB it/PRP back/RB.
2. I/PRP hurt/VBP my/PRP$ back/NN.

Here *back* is tagged as an adverb (RB) in the first sentence and as a noun (NN) in the second. The POS-tagger used for this example is based on a simple rule-based tagger developed by Eric Brill in 1992. Brill’s tagger algorithm has two steps. In the first step, each word is assigned a tag estimated by a large manually tagged corpus. In addition, if new words appear that do not exists in the tagged corpus, it is assigned according to the assumptions: If the first letter is capital then it is assigned a noun tag. Otherwise it is assigned a tag of the class with words that most commonly end with the same three letters. In the second step it examines the tags using a set of rules that examine the order of the tags [Brill, 1992].

4.1.4 n-grams

Patterns can be created by concatenating adjacent tokens into n-grams where $n = \{1, 2, 3...\}$. For the simple case where n is equal to one is also called unigram. An example of how the n-grams are constructed is explained with the example sentence: ”Niklas likes his new backpack”, in Table 3. By using n-grams it may be possible to capture how a word tend to appear in text with respect to other words. Usually n-grams are never longer than $n = 3$. Greater values are likely to create ”too” complex patterns that rarely match.

Table 3: n-grams sequence

n-**gram**	**Sequence**	**Length**
1-gram:	”Niklas”, ”likes”, ”his”, ”new”, ”backpack”	5
2-gram:	”Niklas likes”, ”likes his”, ”his new”, ”new backpack”	4
3-gram:	”Niklas likes his”, ”likes his new”, ”his new backpack”	3

4.2 Feature selection

The goal with feature selection is to select a subset to use for classification. In text data there will always be features that are irrelevant and redundant. The redundant features are those that do not contribute anything or very little in distinguishing the classes from each other. By doing this the dimensionality will be reduced so the amount of data to process is less and this will save time when performing the classification. Another benefit, for some classification algorithms, is that we lower the risk of overfitting the data.

4.2.1 Tf-Idf

Tf-Idf stands for Term frequency - inverse document frequency and is popular in information retrieval (IR). Tf-Idf is a statistical measure that evaluates how important a word is to a document in a corpus and is based upon the idea that words that appear many times in a document provide much information about that document, at the same time, words that appear in all documents provide little, or no, information about any document. Tf-Idf increase the importance of words with many occurrences in a document and decrease the importance of words that appear in many different document.

$$\text{tf-idf} = \mathrm{tf}(t, f) \times \mathrm{idf}(d, D) \tag{1}$$

The simples choice of choosing the term frequency $\mathrm{tf}(t, f)$ is simply to use frequency $\mathrm{f}(t, d)$, i.e. the number of occurrences t in a document d. If the length of the documents differ a lot it could be a good idea to normalize the term frequency since longer documents have more occurrences of words. Equation (2) is an example of what $\mathrm{tf}(t, f)$ might look like.

$$\mathrm{tf}(t, d) = 0.5 + \frac{0.5 \times \mathrm{f}(t, d)}{\max\{\mathrm{f}(w, d) : w \in d\}} \tag{2}$$

Idf is a measure of how important a word is. The $\mathrm{tf}(t, f)$ treats every word equally important. It is obvious that common words such as ”and” and ”to” etc. have high frequencies but give little information about what class the document belongs to. Thus the $\mathrm{idf}(t, d)$, Equation (3), diminishes the importance of such words that occur in many different documents in the corpus by dividing the logarithm of the number total number of documents N by the number of documents d in D which contains the word t.

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{3}$$

4.2.2 Chi-square

The Chi-squared statistics is a measure for investigating the independence between stochastic variables. If the variables are similar chi-square yields a low result and conversely a high result if the variables differ much.

Definition: If ν independent variables x_i are each normally distributed with mean μ_i and variance σ_i^2, then chi-squared is defined as:

$$X^2 = \frac{(x_1 i \mu_1)^2}{\sigma_1^2} + \frac{(x_2 - \mu_2)^2}{\sigma_2^2} + \cdots + \frac{(x_\nu - \mu_\nu)^2}{\sigma_\nu^2} = \sum_{i=1}^{\nu} \frac{(x_i - \mu_i)^2}{\sigma_i^2} \tag{4}$$

Normally chi-squared statistic are used for estimating how good distributions fit together. Here the chi-squared statistic is used as a measure for finding the opposite by filtering out words in a text that have similar distributions for both classes, hence irrelevant for classification [Pedregosa et al., 2011].

4.2.3 Information gain

For understanding information gain we need to explain what entropy is. Entropy is a statistical measure that can be interpreted as the (im)purity of collection or corpus [Mitchell, 1977] and is defined as

$$H(p) = -\sum_{x \in X} p(x) \log p(x). \tag{5}$$

Figure 3 shows the value of the entropy for different probabilities. Information gain can be used to select a sequence of attributes for narrowing down the class of an instance in the shortest way. In terms of entropy information gain is

$$I(X, Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{6}$$

By combining Equation (5) and (6) we get the Equation for information gain.

$$IG(X; Y) = \sum_{x \in X} \sum_{y \in Y} p(x_i, y_i) \log \frac{p(x_i, y_i)}{p(x_i)p(y_i)}. \tag{7}$$

Instead of just comparing two arbitrary attributes, as in Equation (7), information gain can be generalized to measure the decrease in entropy by comparing an attribute to a set of attributes. This is preferable when dealing with text categorization [Nicolosi, 2008]. Information gain is in that case defined as

$$IG(t) = \sum_{c \in \{c_k, !c_k\}} \sum_{t \in \{t_i, !t_i\}} P(t, c) \log \frac{p(t, c)}{p(t)p(c)}. \tag{8}$$

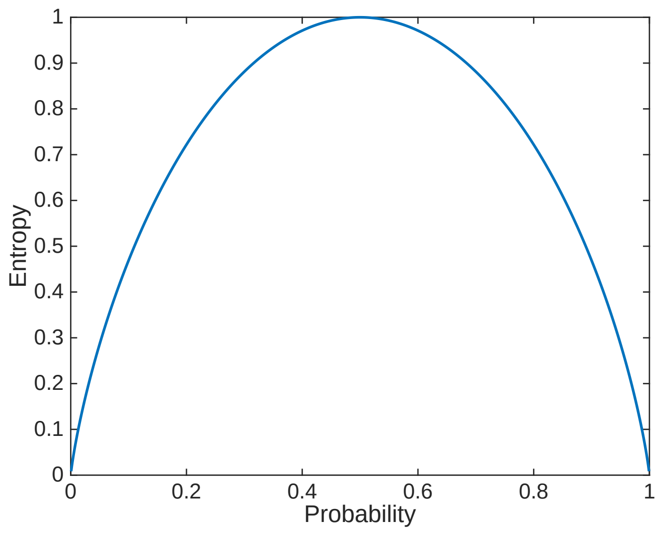


Figure 3: Entropy plot.

4.3 Classification

When building a model for text classification it is important to ask how much data that is available. Much? Little? None? One of the most challenging problems in classification is to get hold of enough training data. If a huge amount of data is available then it is not so obvious what classifier to use and an idea might be to choose a classifier with good scalability [Banko and Brill, 2001]. However, depending on how the data is available to us there are three different learning methods.

Supervised learning

Supervised learning is based on learning a model given a set of correctly classified data. The aim of supervised learning is to train a model to recognize discriminant attributes (in statistical literature supervised learning is sometimes known as discriminant learning) in the data [Michie et al., 1994]. Let us say that we want to build a model that can separate news articles about soccer from religion. With a given set of labeled data the model can be trained to e.g. learn that some words (attributes) are used solely, or more frequently, in one of the classes. The model might have learned that articles containing words such as ”referee”, ”player” or ”goaltender” is more likely to be an article about sport. Conversely, words such as ”god”, ”church” and ”Islam” are more likely to occur in an article about religion. New unseen news articles can then be predicted to be an article about soccer or religion depending on the frequency of its words.

Unsupervised learning

Labelled data is not always available and in those cases supervised learning is not possible. Unsupervised learning is a method for finding patterns in data without any information about the outcome measure of the data. It is difficult to validate results made from a unsupervised classification since the outcome measure is unknown. The strength of unsupervised learning lies within finding associations in data and show how it is arranged [Mitchell, 1977, Michie et al., 1994].

Semi-supervised learning

Semi-supervised learning is based on the fact that labelled data can be hard to obtain and unlabelled data is cheap. The idea is to combine a small set of labelled data and expand it using unlabelled data with the help of unsupervised learning [Zhu, 2007]. The result is then a big set of labelled data, perhaps containing some noise, that can be used for supervised classification.

For getting a high accuracy a good idea is to try some different classifiers on your data. In the remainder of this section we present four classifiers that have been used during this project, every single one is used for supervised learning.

4.3.1 Decision Tree

Tree-based learning can be applied to both classification and regression problems. The main procedure is to recursively split nodes of data $\{x_i\}$ into two new nodes $(N_{x_i \geq t}, N_{x_j \leq t})$ by one dimension of feature X_j and a threshold t. The algorithm for building a decision tree is basically two steps

1. Divide the feature space $X_1, X_2, ..., X_p$ into J distinct and non-overlapping regions, $R_1, R_2, ..., R_J$.
2. Every observation that falls into the region R_J, are predicted the same, which is simply the majority (mean for regression) of the target values from the training instances in R_J.

Say that we after step 1 obtain two regions R_1 and R_2 and that the majority of the training instances in R_1 are ironic. Then all new examples $x \in R_1$ will predicted as ironic. An

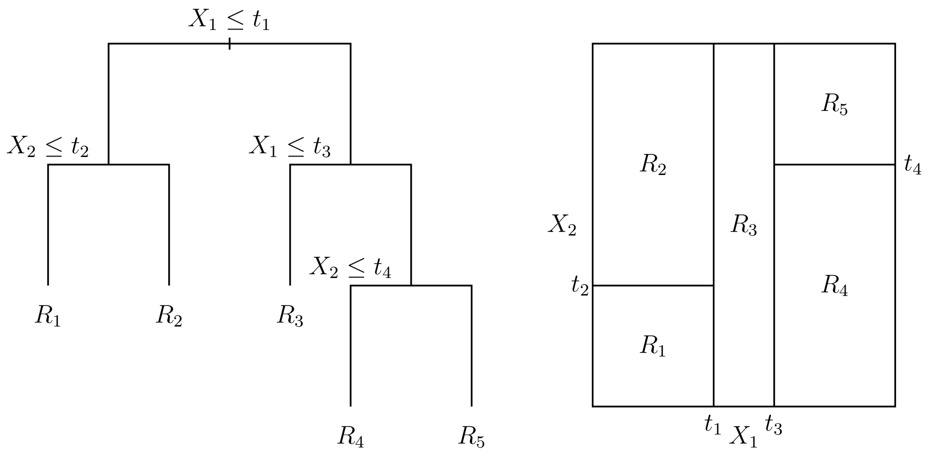


Figure 4: The features space divided into regions R_i by splits t_j. To the left in the figure is the image of the decision tree. All new instances $x \in R_i$ will be predicted the same.

example of how the tree grows with each split and how the feature space gets divided into regions, is shown in Figure 4. We apply recursive splitting using two features X_1 and X_2. First we split $X_1 = t_1$. Then we split the region $X_1 \leq t_1$ at $X_2 = t_2$. After that we split the region $X_1 > t_1$ at $X_1 = t_3$. Finally we split the region $X_1 > t_3$ at $X_2 = t_4$. From this we obtain five regions.

Some different methods for measuring the impurity in the node, and evaluating where to perform the next split, are the Gini impurity and entropy and the misclassification error

$$\text{Gini: } \sum_{k=1}^{K} p_{mk}(1 - p_{mk}). \tag{9}$$

$$\text{Entropy: } -\sum_{k=1}^{K} p_{mk} \log p_m k \tag{10}$$

$$\text{Misclassification: } 1 - \max_k p_{mk} \tag{11}$$

For a binary classification $(K = 2)$ equations (9), (10) and (11) collapse to

$$\text{Gini: } 2p(1 - p) \tag{12}$$

$$\text{Entropy: } -p\log(p) - (1 - p)\log(1 - p) \tag{13}$$

$$\text{Misclassification: } 1 - \max(p, 1 - p) \tag{14}$$

A common approach is to keep splitting the tree until it reaches some defined stopping criteria e.g. when all leaf nodes are pure. This yield a perfect classification on the training data but is very likely to overfit and perform badly on new unseen data. Another approach is to stop building the tree when a split does not improve the classification by some threshold value. This approach will generate a smaller tree but is short-sighted since a bad split might be followed by a good split that separates the classes effectively. A better approach is to generate a large tree T_0 and then prune it, i.e. remove leaf nodes, to a smaller subtree T. The idea is to keep the structure of the basic tree but remove unnecessary splits that may cause overfit. The candidate subtrees can be selected using the weakest link criterion defined as

$$C_\alpha = \sum_{m=1}^{|T|} Q_m(T) + \alpha|T| \tag{15}$$

where α is a tuning parameter, each corresponding to a subtree, that represents a trade-off between the complexity of the tree and how good the tree fits the data. A value of $\alpha = 0$ corresponds to the original tree T_0. $Q_m(T)$ is any of the impurity measures in Equations (9), (10) and (11). The goal is to, for each α, find a subtree that minimizes Equation (15). Figure 5 is an example how to grow a tree and selecting a suitable subtree.

4.3.2 Support vector machines (SVM)s

SVM are a set of un-/supervised learning methods for machine learning that can be used for classification, anomaly detection and regression. One of SVMs strong suits is that they are very efficient when your data is high dimensional and also effective in cases

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply weakest link pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, .., K$:
   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.
   (b) Evaluate the prediction error on the data in the left-out kth fold, as a function of α. Average the results for each value of α, and pick α to minimize the average error.
4. Return the subtree from Step 2 that corresponds to the chose value of α.

Figure 5: Algorithm for building a decision tree.

where number of dimensions are greater than the number of samples. SVMs are especially good to solve text and hypertext categorization since the application decreases the use of label training occasions [Witten and Frank, 2005]. Disadvantages with SVM are that they do not provide an estimate of the probability. To get these a calculation of an expensive cross-validation may be performed. In cases when the number of samples are significantly less than the number of features is likely that the method gives a poor result.

The SVM classifies the categories by dividing them with a hyperplane. A hyperplane is a subspace of one dimension less than the surrounding space. As in Figure **??**, which is two dimensional, the hyperplane is one dimensional (a straight line). The idea is to find a hyperplane that has the biggest distance to each class because it strives to separate the categories as much as possible. The hyperplanes equation that divides the categories is the SVM for the specific case.

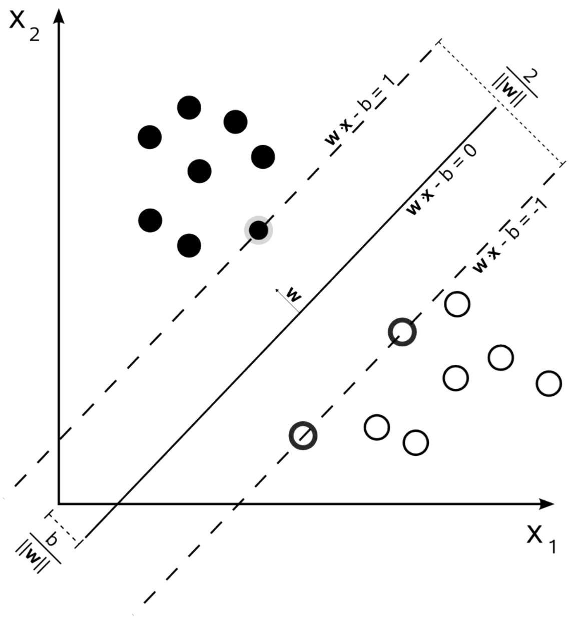


Figure 6: Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vector.

For cases when training data is linearly inseparable, it is common to project the data into a higher dimension so an easier separation of non-linear classes could be done, see Figure 7. A classic statement from T.M. Cover in 1965 called Cover's theorem states that:

A complex pattern-classification problem, cast in high-dimensional space nonlinearly, is more likely to be linearly separable than in a low-dimensional space, provided that the space is not densely populated.

This is true because a higher dimension of the hyperplane is now used to make the separation, hence the new hyperplane will fit the data better. To avoid projection into higher space something called the "kernel-trick" can be used. This "trick" is often computationally cheaper compared to project into a higher dimension. What the kernel-trick does is that it uses the result of a kernel function of two data points, which equal to projecting them into a higher dimensional space and use their inner product in this new space. This is possible since the data points entering the SVM equation only get there as inner products.

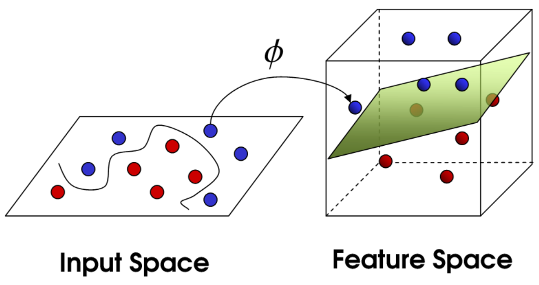


Figure 7: Linearly inseparable data projected into a higher space.

4.3.3 Naive Bayes

The Naive Bayes classifier is a popular machine learning classifier, because of its simplicity and is often used as baseline for text categorization. The classifier makes the "naive" assumption that independence occurs between all the features. The classifier is applied from Bayes' theorem, Equation 16. Naive Bayes is often used in spam filtering and document classification. One downside with Naive Bayes that it is a bad estimator so probability outputs are not very trustworthy. For the cases where the features are discrete as in text documentation multinomial and Bernoulli are popular distributions.

$$p(C_k|x) = \frac{p(C_k) \cdot p(x|C_k)}{p(x)} \tag{16}$$

4.3.4 Boosting

Boosting is common in machine learning and used to decrease bias and variance during supervised learning. The boosting algorithms turns weak learners into strong ones by putting more weights on those that are seen as weak learners. The concept of boosting was first introduced at the end of the late 80-s [Kearns, 1988]. At that point the boosting algorithms were not adaptive, and could not specifically focus on the weak learners. But Schapire and Freund later developed an algorithm that was adaptive, called AdaBoost. Examples of other boosting algorithms are LPBoost and LogitBoost. The main difference between these are how they divide their weights on the points in the training data. In this thesis have Adaboost been used.

AdaBoost is designed to turn multiple weak learners into a strong one. Adaboost starts by making a poor classification, slightly better than guessing randomly. And from that are the data that is not correctly classified given an extra ”weight”. The ”weight” increases the probability for that data point to be correctly classified in the following iterations. An input value of how many iterations the algorithm should run before it stops is chosen by the user. However if all data is correctly classified the algorithm will stop in advance.

Below are the steps how the adaboost algorithm is calculated described [Hastie et al., 2010]. In the steps $G(x)$ is the final classifier and $G_m(x)$ the classifier in each iteration. α_m is the weight factor calculated in each iteration.

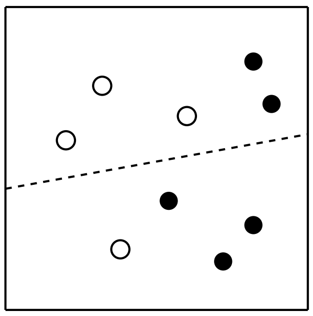
1. Initialize the observation weights w_i=1/N, i=1,2,...,N.
2. From m=1 to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute err$_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$.
   (c) Compute α_m =log((1-err$_m$)/err$_m$).
   (d) Set $w_i \leftarrow w_i \cdot exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]\ i = 1, 2, ..., N$.
3. Output G(x) = sign $[\sum_{m=1}^{M} \alpha_m G_m(x)]$.

An illustration of how Adaboost makes the weak learners into a strong one is shown in Figure 8.

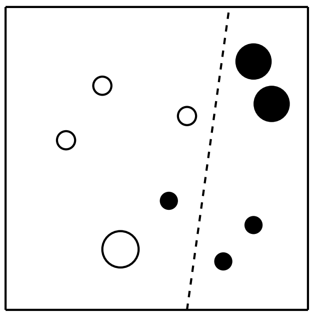
4.4 Model Evaluation

It is of great importance to estimate how well the model works. For error estimating it is common to divide the entire data set in to three sets, namely: train set, validation set and test set [Hastie et al., 2010]. The train set is used to train the model, the training is done by using the labeled data so the model learns how the input and the expected output is connected. The validation set is used for finding the best model fitted to the train set. The parameters for the model is also tuned using the validation set. The model that obtain the best result on the validation data is then used on the test set to see how it performs on unbiased data. If much data is available, it is preferable to split the data into three equally big sets, otherwise a normal split is to assign 50 % for training 25 % each for validation and testing [Hastie et al., 2010].

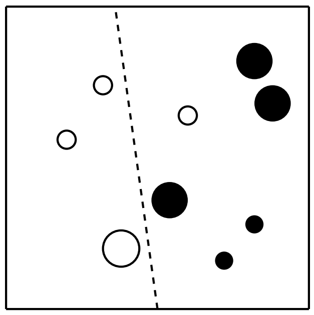
However, by partitioning the entire set into three separate sets we drastically decrease the number examples for training. If the number of training instances are low it might



(a) Initial uniform weight on training examples and the dashed line symbolize the weak classifier 1.



(b) Incorrect classifications re-weighted more heavily and a ”new” weak classifier is marked.



(c) Final classifier is weighted combination of weak classifiers.

Figure 8: Example of the Adaboost algorithm.

impact the result of the whole model in a negative way. Cross-Validation (CV) is a solution to that problem. A set should still be set aside for the test but CV remove the need of a validation set. A common approach is K-fold CV, where the data is split to K number of folds. The model is then trained on $(K-1)$ folds and tested on the remainder [Hastie et al., 2010]. This is done for $k = 1, 2, ..., K$. Finally we combine these results and use it as the validation estimate.

Table 4: 5-fold Cross-Validation. $(k-1)$ folds are used for training and the remainder is used for testing.

1	2	3	4	5
Train	Validation	Train	Train	Train

A common problem in machine learning is overfitting. In this thesis overfitting gets when the model over adapts to the perculiarities of the training data. The problem with this is that the noise that occur in the training data will also be adapted to the model, which will make the model perform poorly on new unseen data [Domingos, 2012]. The model should only be fitted to the patterns that are discriminating for the whole population not just the part that the model is being trained on.

4.4.1 Error Estimation

Some interesting estimations in classification problems are accuracy, precision and recall. In statistical terms recall the ratio between the number of true positives and the sum of true positives and negative positives

$$\text{recall} = \frac{t_p}{t_p + f_n} \tag{17}$$

where t_p is the number of true positives and f_n the number of false negatives. Precision is the ratio between the true positive and the sum of true positives and false positives

$$\text{precision} = \frac{t_p}{t_p + f_p} \tag{18}$$

where t_p is the number of true positives and t_p the number of false positives [Pedregosa et al., 2011]. Finally the F1-score is an accuracy measure that can be interpreted as weighted mean of the precision and recall:

$$\text{F1-score} = \frac{2t_p}{2t_p + f_p + f_n}. \tag{19}$$

In the case of the spam filter example in Chapter 1, the recall for the spam is the ratio of how many of the true spam emails were classified as spam. The precision is the ratio of how many of the classified spam emails were actually spam.

5 Method

In this chapter we will described how the data is processed from raw-text. We will also present the features used for this model and briefly motivate why they are selected. In addition we will outline the used tools for this study and the purpose of using them.

5.1 Technologies - Python

This project needs a programming language that is convenient to use for particular machine learning methods. Python is a versatile programming language that supports both object oriented programming and functional programming. It provides a wide spectra of libraries that supports a lot of different programming tasks and is also open source. The libraries are easy to install and well documented. One more advantage with Python is that it is easy to read because the use of indents and new lines instead of more brackets. The libraries described below are some of the most common that have been used in the project. More information about Python and the libraries can be found at Pythons own webpage.[6]

- **Numpy** gives support for high-level mathematical functions, large arrays and matrices. When using linear algebra and random number capabilities is Numpy very useful. Numpy also has utilities for integrating Fortran and C/C++ code.
- **Scikit-learn** is a Machine Learning library that Google started working on in 2007. The first public release was in 2010 and has been updated continuously. It is built on the math packages numpy and scipy. Among many other features it includes a variety of regression, classification and clustering algorithms such as naive Bayes, random forests, support vector machines, boosting algorithms etc. [Pedregosa et al., 2011]
- **NLTK** is an abbreviation for Natural Language ToolKit, it is a Python library that is useful when you work with human language data. The library consists of more than 50 different lexical assets and corpora including such as WordNet, parsing and tagging.
- **SciPy** is a library containing many routines for numerical integration and optimization.
- **Textblob** is a library that is used when working with textual data. It provides an API for text processing, text mining and text analysis. Some specific tools in these areas that we have used are sentiment analysis, classification and translation.

5.2 Data preprocessing

Text data is seldom on a form amenable for classification. Before extracting any features we process the text so that it better fit our purpose, but also for reducing the amount of noise in our text such as illegal characters and misspellings.

[6]https://www.python.org/doc/

5.2.1 Amazon

We initially obtain the Amazon reviews as an XML-file with the entities: rating, date, title, author, and review-text. If we were to work only with amazon reviews then we could use all entities when extracting features, specially rating. But since we only consider text categorization, we only use the review-text. After extracting the text we eliminate all characters that are not unicode. To improve the quality of the text we created a dictionary of the most common misspellings [7] in the English language, which we used to perform a spell check on the corpus. Finally we created a dictionary for expanding most of the contractions [8] and auxiliaries (we do not expand all contractions since some may be ambiguous e.g. ”he’s” may refer to ”he is” or ”he has” depending on the context). Table 5 is an example how the text is processed. The contractions are expanded, all capital letters are turned into lowercase letters and the text is stemmed using the Lancaster Stemmer from the NLTK library.

Table 5: Text-processing, before and after.

The reason I’m giving such a great rating to this netbook is that having said all of the things that the NB305 - and for that matter netbooks in general - can’t do the simple fact is that I don’t want to do and wouldn’t expect to do those things on a 10.1 in screen to begin with.
the reason i am giv such a gre rat to thi netbook is that hav said al of the thing that the nb305 - and for that mat netbook in gen - can not do the simpl fact is that i do not want to do and would not expect to do thos thing on a 10.1 in screen to begin with.

5.2.2 Twitter

Tweets are in general very noisy i.e. people tweeting do not necessarily follow all lexical rules when posting a tweet [Tsur et al., 2010]. In twitter it is common that tweets refer to a picture or a link, and tweets with #sarcasm are no exception. Irony may be impossible to identify if taken out of context. We have therefor filtered out all tweets containing any hyperlink or picture attached to the tweet. To avoid that the sarcasm in a tweet actually occurred in a previous tweet we remove all tweets that contain a reference to another user (i.e. containing @<Username>). We also remove tweets containing the keyword ”RT” indicating that the tweet is re-tweet (post /share someone else’s entry). Finally we removed duplicates. From the tweets that are kept we remove the hashtags from the tweets, including the sarcasm and sarcastic hashtags. After filtering the corpus we have in total a set of 143’120 tweets containing 25’278 sarcastic tweets and 117’842 non-sarcastic tweets. As a final step before the features are extracted from the text, we correct the misspellings and expand the contractions, using the same methods as for the amazon corpus.

[7] http://www.oxforddictionaries.com/words/common-misspellings

[8] http://grammar.about.com/od/words/a/EnglishContractions.htm

5.3 Chosen Features

The features considered for detecting irony and sarcasm comes from the literature survey that was conducted in the beginning of this study and are based on different aspects that we believe may display irony. All the different features are grouped together and always tested together since we believe that they represent similar aspects. In total we have four groups of features:

- Sentiments
- Punctuations
- Vocabulary
- Structure

In total we have 26 features and an additional number of unigrams and bigrams depending of the feature selection conducted after extracting all the features.

5.3.1 Sentiments

The sentiments features are constructed to find the polarity in the reviews. By examining our corpus with ironic snippets it shows that it is rarely the case that the whole document is ironic, rather there are one or a few sentences that are truly ironic in an ironic document. The idea is to find irony following the definition of ”saying the opposite of what you believe is true”. The classifier scores each sentence with a value in the range of [-1,1], where negative one is a really negative sentence and positive one is a very positive sentence (0 is a neutral sentence). All the sentiment scores are obtained by the sentiment classifier in the TextBlob library.

5.3.2 Punctuation

We create features consisting of ratios of commas, full stops, colons, semi-colons, ellipsis, quotations, hyphens, question marks and exclamation marks against the total number of punctuations characters that a review may contain. We have also added a count for number of words capital letters. With these features we aim to capture any marker that can indicate irony. E.g. as in the example (3) from chapter 1, the tweet was understood as ironic due to the combination of the words capital letters combined with the ellipsis. Other aspects that we aim to capture are exaggerations by heavily usage of e.g. exclamation marks.

5.3.3 Vocabulary

In this group we have features for capturing the style in the ironic text. The aim is to find words that often express irony e.g. ”yeah right” [Tepperman et al., 2006]. To enhance the text, we stem all words using the Porter stemmer algorithm and create unigrams and bigrams(1-gram and 2-gram) from the stems. For avoiding an unnecessary large dimension space only n-grams that occur more than 10 times in the training data are considered as features.

5.3.4 Structure

For understanding the structure of the document we have created features from the ratios of: adverbs, adjectives, nouns, determiners, prepositions, interjections, modal verbs and verbs. Interjections are of special interest here and have been shown to be a discriminant feature for irony [Carvalho et al., 2009]. For the other tags we hope to detect some differences in the usage. All the part of speech are obtained by using the POS-tagger in the NLTK library.

5.4 Feature selection

The feature scheme from the vocabulary features results in a high dimensional feature space with over 10’000 features. Both for avoiding overfitting but also for reducing the dimensionality of the problem, we perform two steps for selecting a subset containing the most discriminant features for the task. First we use the chi-square measure to select the 2’000 words with the highest variance. Secondly we use information gain on the entire set to select a final subset for classification. Finally we scale the dimensions using tf-idf-transformation explained in Chapter 4.

6 Results

The result section includes an evaluation of the different data sets where information gain have been used for all special features. Results of how the different classifiers perform for different features are showed. Tables of the classification report(precision, recall and f1-score), confusion matrix and a plot where a comparison of the validation - and training error is done. A final model for both the data sets is presented after that.

Before classifying the data we balance the data so that we have the same amount of instances from both classes. Similar to [Reyes et al., 2012] we compare our result with a baseline based on a trivial classifier that constantly classify all instances as non-sarcastic. Hence, the baseline is an accuracy of 50 %.

Both datasets are split into one train set and one test set where the test set accounts for 20% of the entire dataset. We perform a 5-fold cross-validation on the train set for validating our models and selecting the best parameter settings.

In this section we mention ”all features” and ”our own features”. Our own features refers to punctuation and sentiments and all features refers to our own features plus the words.

The results will be presented in the following order:

1. Information gain on special features for both data sets.
2. Classification report, confusion matrix and plot of training and test error for each classifier and case.
3. Final model for the data sets.

6.1 Data evaluation

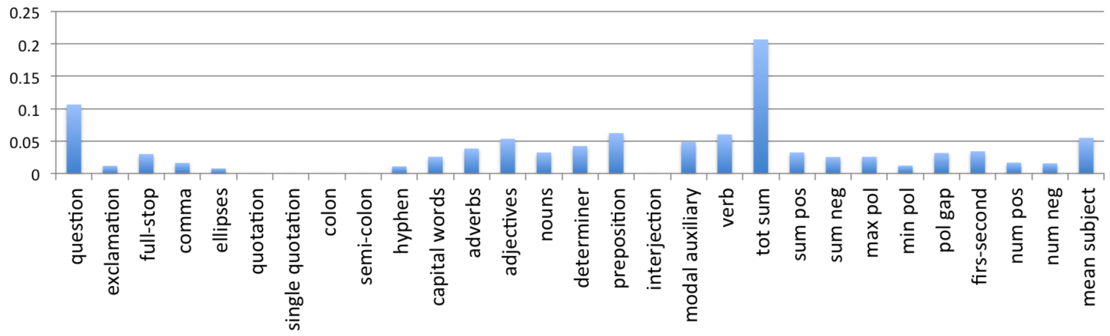


Figure 9: Information gain for all of our special features between the ironic and the regular corpora. A higher information gain indicates a more discriminating feature for the Amazon reviews.

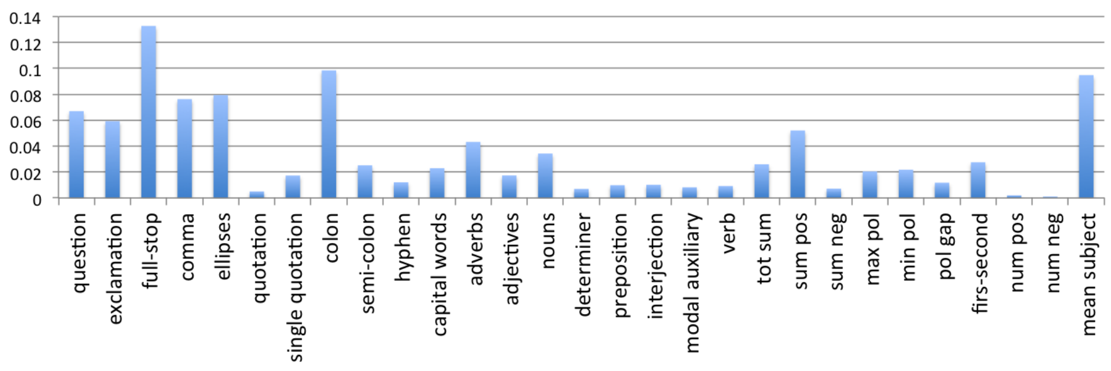


Figure 10: Information gain for all of our special features between the ironic and the non-ironic corpora. A higher information gain indicates a more discriminating feature for the twitter data.

6.2 Amazon Classification

The final classification of the data sets will be given for each of the four classifiers presented in chapter 4.

6.2.1 Decision tree classifier

Table 6: Decision tree classification of the Amazon reviews with all features. Parameter values for the best classifier: the function to measure the quality of the split is Gini and number of leaf nodes = 13. Model accuracy and confusion matrix.

	precision	recall	f1-score
Non-sarcastic	0.82	0.70	0.75
Sarcastic	0.69	0.81	0.75
avg / total	0.76	0.75	0.75

		Predicted class		
		Non-sarcastic	Sarcastic	
Actual class	Non-sarcastic	61	14	75
	Sarcastic	27	62	89
		88	76	

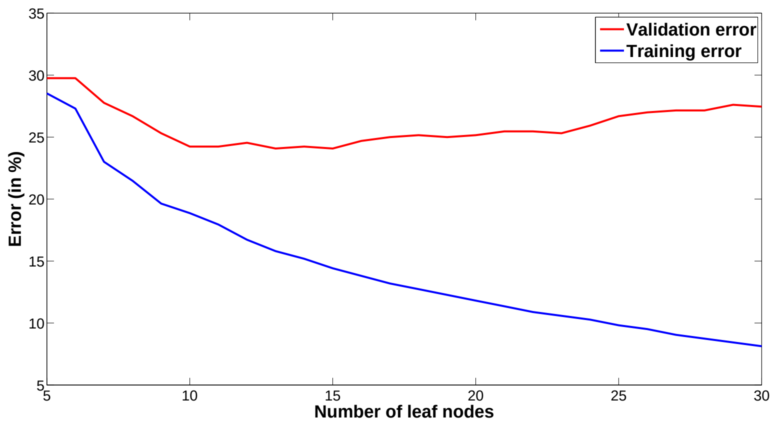


Figure 11: Validation - and training error vs. number of leaf nodes.

An accuracy of 75 % is obtained with the decision tree classifier when all features are used. In the figure above is the training error decreasing when the number of leaf nodes is increasing. The validation error is increasing when the number leaf nodes is greater than 14.

Table 7: Decision tree classification of the Amazon reviews with our own features. Parameter values for the best classifier: the function to measure the quality of the split is gini and number of leaf nodes = 14. Model accuracy and confusion matrix.

	precision	recall	f1-score
Non-sarcastic	0.77	0.69	0.73
Sarcastic	0.67	0.76	0.71
avg / total	0.73	0.72	0.72

		Predicted class		
		Non-sarcastic	Sarcastic	
Actual class	Non-sarcastic	57	18	75
	Sarcastic	28	61	89
		85	79	

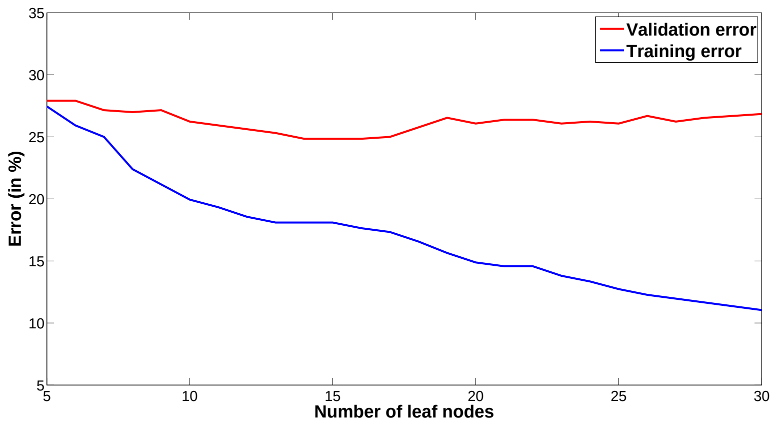


Figure 12: Validation - and training error vs. number of leaf nodes.

An accuracy of 72 % is given with the decision tree classifier when our own features were used. In the figure above is the training error decreasing when the number of leaf nodes is increasing. The validation error is increasing when the number leaf nodes is greater than 14.

6.2.2 Naive Bayes

Table 8: Naive Bayes classification of the Amazon reviews with Bernoulli distribution on our own features. Model accuracy and confusion matrix.

	precision	recall	f1-score
Non-sarcastic	0.94	0.74	0.83
Sarcastic	0.76	0.95	0.84
avg / total	0.86	0.84	0.83

		Predicted class		
		Non-sarcastic	Sarcastic	
Actual class	Non-sarcastic	71	4	75
	Sarcastic	23	66	89
		94	70	

An accuracy of 83 % is obtained with the naive Bayes classifier when our own features are used.

Table 9: Naive Bayes classification of the Amazon reviews with Bernoulli distribution for all features. Model accuracy and confusion matrix.

	precision	recall	f1-score
Non-sarcastic	0.73	0.78	0.75
Sarcastic	0.71	0.67	0.69
avg / total	0.73	0.73	0.72

		Predicted class		
		Non-sarcastic	Sarcastic	
Actual class	Non-sarcastic	50	25	75
	Sarcastic	20	69	89
		70	94	

An accuracy of 72 % is obtained with the naive Bayes classifier when all features are used.

6.2.3 Adaboost

Table 10: Adaboost classification of Amazon reviews with our own features. Parameter values for the best classifier when the number of iterations = 5. Model accuracy and confusion matrix.

	precision	recall	f1-score
Non-sarcastic	0.83	0.61	0.70
Sarcastic	0.65	0.85	0.74
avg / total	0.75	0.73	0.73

		Predicted class		
		Non-sarcastic	Sarcastic	
Actual class	Non-sarcastic	64	11	75
	Sarcastic	35	54	89
		99	65	

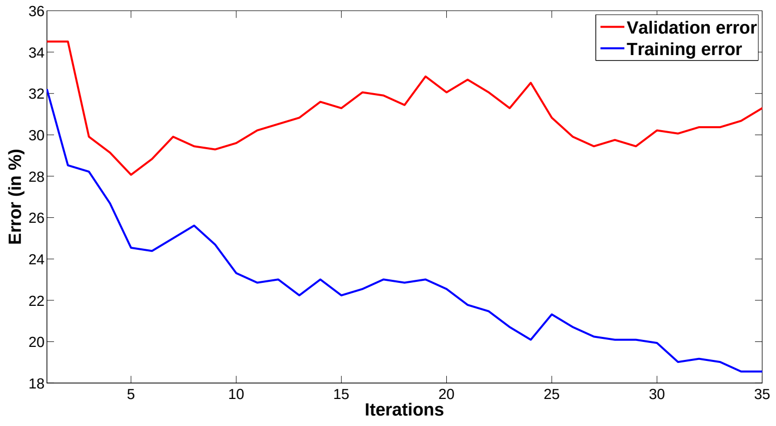


Figure 13: Validation - and training error vs. number iterations for the amazon reviews with our own features.

An accuracy of 83% is obtained with the adaboost classifier when our own features are used.

Table 11: Adaboost classification of the Amazon reviews with all features. Parameter values for the best classifier when the number of iterations = 49. Model accuracy and confusion matrix.

	precision	recall	f1-score
Non-sarcastic	0.85	0.67	0.75
Sarcastic	0.69	0.85	0.76
avg / total	0.77	0.76	0.76

		Predicted class		
		Non-sarcastic	Sarcastic	
Actual class	Non-sarcastic	64	11	75
	Sarcastic	29	60	89
		93	71	

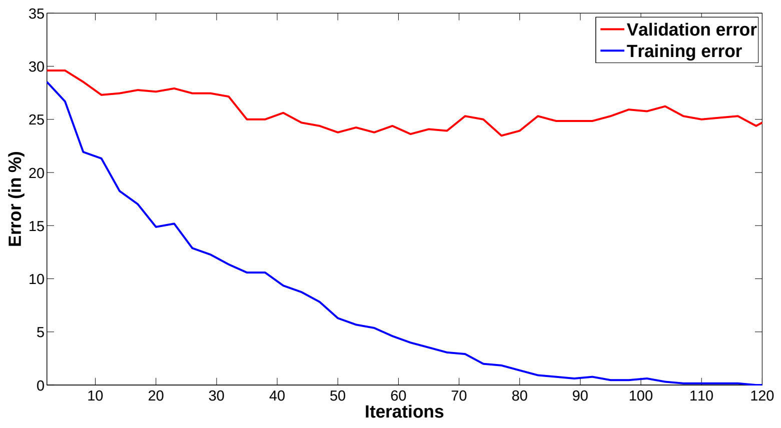


Figure 14: Validation - and training error vs. the number of iterations for the Amazon reviews with all features.

An accuracy of 76% is obtained with the adaboost classifier when our own features are used. The training error decrease as the number of iterations increase and the validation error is lowest after 49 iterations and increases after that.

6.2.4 SVM

Table 12: SVM classification of the Amazon reviews with all features. Parameter values for the best classifier is obtained with penalty parameter = 0.5 and a linear kernel function. Model accuracy and confusion matrix.

	precision	recall	f1-score
Non-sarcastic	0.89	0.85	0.87
Sarcastic	0.84	0.88	0.86
avg / total	0.87	0.87	0.87

		Predicted class		
		Non-sarcastic	Sarcastic	
Actual class	Non-sarcastic	69	9	78
	Sarcastic	13	73	86
		82	82	

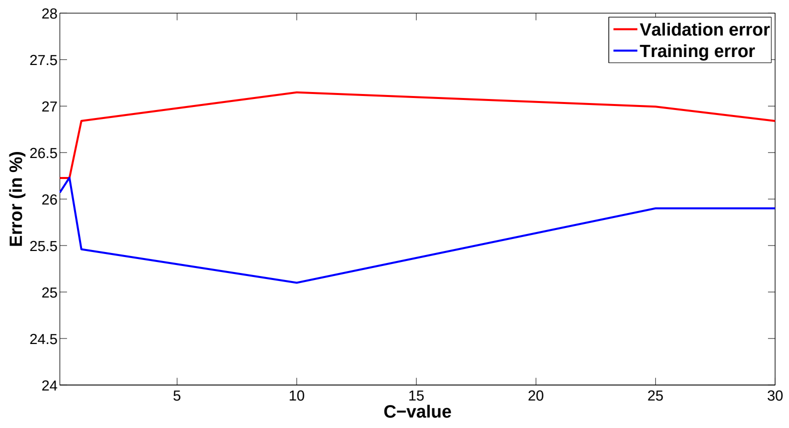


Figure 15: Validation - and training error vs. penalty parameter C.

An accuracy of 87% is obtained with the SVM classifier when all features are used. The training error starts to decrease but increases after a while. The validation error is lowest from the start.

Table 13: SVM classification of the Amazon reviews with our own features. Parameter values for the best classifier is obtained with penalty parameter = 0.1 and a linear kernel function. Model accuracy and confusion matrix.

	precision	recall	f1-score
Non-sarcastic	0.72	0.72	0.72
Sarcastic	0.69	0.69	0.69
avg / total	0.71	0.71	0.71

		Predicted class		
		Non-sarcastic	Sarcastic	
Actual class	Non-sarcastic	54	24	78
	Sarcastic	24	62	86
		78	86	

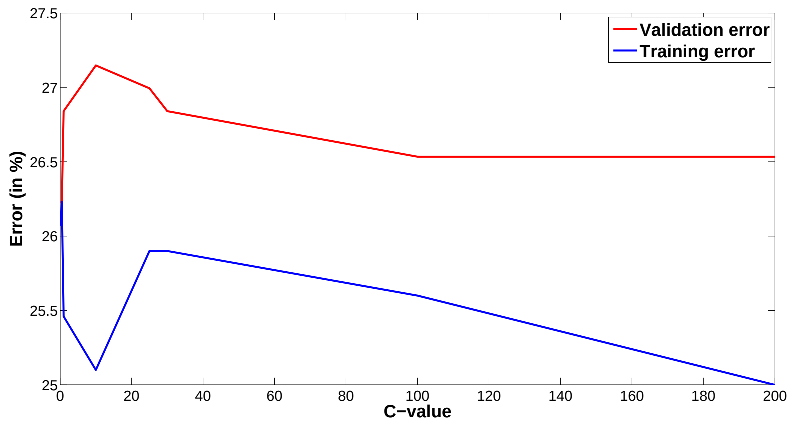


Figure 16: Validation - and training error vs. penalty parameter C.

An accuracy of 71% is obtained with the SVM classifier when all features are used. The training error is increasing when the penalty parameter increases. The validation error is lowest when the penalty parameter is small.

6.3 Twitter Classification

6.3.1 Decision Tree

Table 14: Decision tree classification of the Twitter data with all features. Parameter values for the best classifier: number of leaf nodes = 350. Model accuracy and confusion matrix.

	precision	recall	f1-score
Non-sarcastic	0.68	0.75	0.71
Sarcastic	0.73	0.66	0.69
avg / total	0.70	0.70	0.70

		Predicted class		
		Non-sarcastic	Sarcastic	
Actual class	Non-sarcastic	1353	692	2045
	Sarcastic	498	1457	1955
		1851	2149	

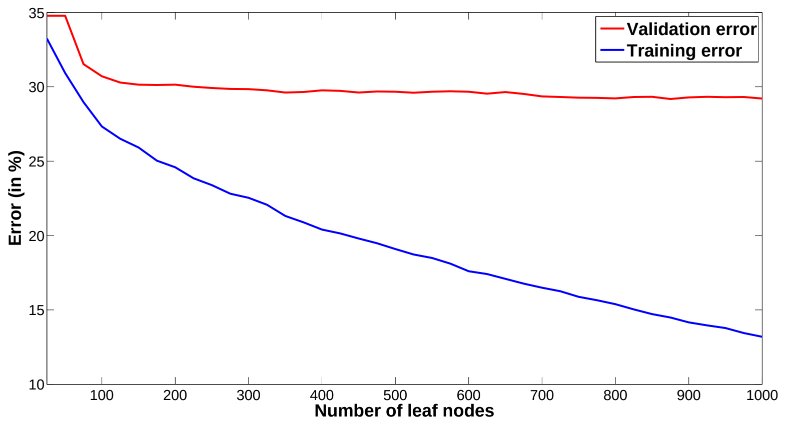


Figure 17: Validation - and training error vs. number of leaf nodes.

An accuracy of 70% is obtained with the decision tree classifier when all features are used. The training error is decreasing with the increasing number of leaf nodes.

Table 15: Decision tree classification of the twitter data with our own features. Parameter values for the best classifier: number of leaf nodes=14. Model accuracy and confusion matrix.

	precision	recall	f1-score
Non-sarcastic	0.72	0.67	0.68
Sarcastic	0.68	0.73	0.72
avg / total	0.70	0.70	0.70

		Predicted class		
		Non-sarcastic	Sarcastic	
Actual class	Non-sarcastic	1585	460	2045
	Sarcastic	611	1344	1955
		2196	1804	

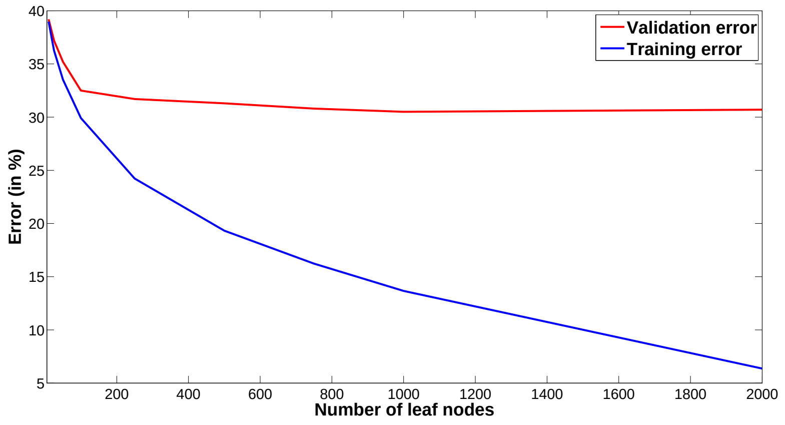


Figure 18: Validation - and training error vs. number of leaf nodes.

An accuracy of 70% is obtained with the decision tree classifier when our own features are used. The training error is decreasing with the increasing number of leaf nodes.

6.3.2 Naive Bayes

Table 16: Naive Bayes classification of the Twitter data with Bernoulli distribution for all features. Model accuracy and confusion matrix.

	precision	recall	f1-score
Non-sarcastic	0.65	0.64	0.64
Sarcastic	0.65	0.66	0.66
avg / total	0.65	0.65	0.65

		Predicted class		
		Non-sarcastic	Sarcastic	
Actual class	Non-sarcastic	3339	1717	5056
	Sarcastic	1797	3147	4944
		4693	5419	

An accuracy of 65% is obtained with the naive Bayes classifier when all features are used.

Table 17: Naive Bayes classification of the Twitter data with Bernoulli distribution for our own features. Model accuracy and confusion matrix.

	precision	recall	f1-score
Non-sarcastic	0.57	0.66	0.62
Sarcastic	0.61	0.52	0.56
avg / total	0.59	0.59	0.59

		Predicted class		
		Non-sarcastic	Sarcastic	
Actual class	Non-sarcastic	2629	2470	5099
	Sarcastic	1685	3328	5013
		4314	3927	

An accuracy of 59% is given with the naive Bayes classifier when our own features were used.

6.3.3 Adaboost

Table 18: Adaboost classification of twitter data with all features. Parameter values for the best classifier: number of iterations = 250. Model accuracy and confusion matrix.

	precision	recall	f1-score
Non-sarcastic	0.69	0.75	0.72
Sarcastic	0.74	0.67	0.70
avg / total	0.71	0.71	0.71

		Predicted class		
		Non-sarcastic	Sarcastic	
Actual class	Non-sarcastic	1346	699	2045
	Sarcastic	504	1451	1955
		1850	2150	

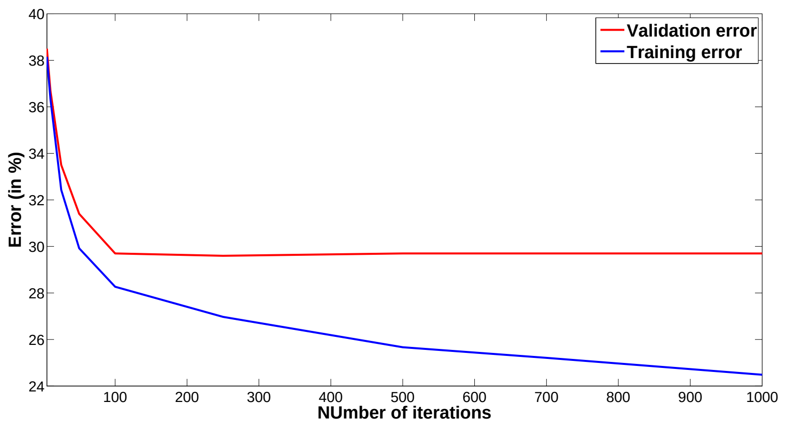


Figure 19: Validation - and training error vs. number of iterations.

An accuracy of 70% is given with the adaboost classifier when all features were used. The training error is decreasing when the number of iterations increases. The lowest validation error is given when the number of iterations are 250.

Table 19: Adaboost classification of twitter data with our own features. Parameter values for the best classifier: number of iterations = 250. Model accuracy and confusion matrix.

	precision	recall	f1-score
Non-sarcastic	0.60	0.65	0.63
Sarcastic	0.64	0.59	0.61
avg / total	0.62	0.62	0.62

		Predicted class		
		Non-sarcastic	Sarcastic	
Actual class	Non-sarcastic	1201	844	2045
	Sarcastic	675	1280	1955
		1876	2124	

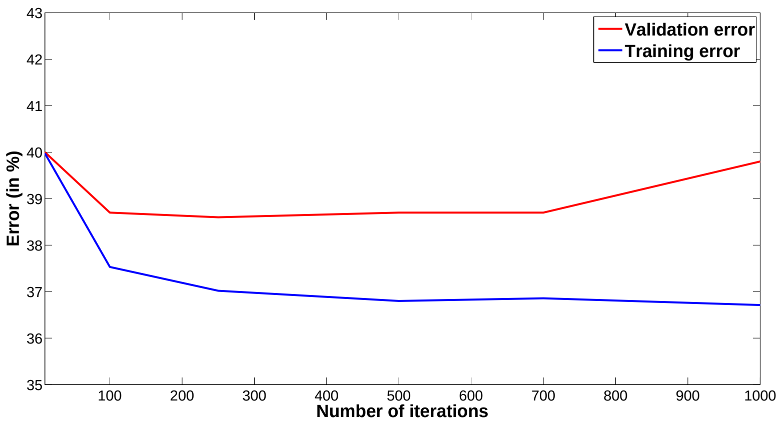


Figure 20: Validation - and training error vs. number of iterations.

An accuracy of 62% is given with the adaboost classifier when our own features are used. The training error decreases at the beginning when the number of iterations increases but then stabilize. The lowest validation error is given when the number of iterations are 250.

6.3.4 SVM

Table 20: SVM classification of the twitter data with all features. Parameter values for the best classifier is obtained with penalty parameter = 1 and a linear kernel function. Model accuracy and confusion matrix.

	precision	recall	f1-score
Non-sarcastic	0.64	0.66	0.70
Sarcastic	0.73	0.58	0.65
avg / total	0.68	0.67	0.67

		Predicted class		
		Non-sarcastic	Sarcastic	
Actual class	Non-sarcastic	1192	853	2045
	Sarcastic	448	1507	1955
		1640	2360	

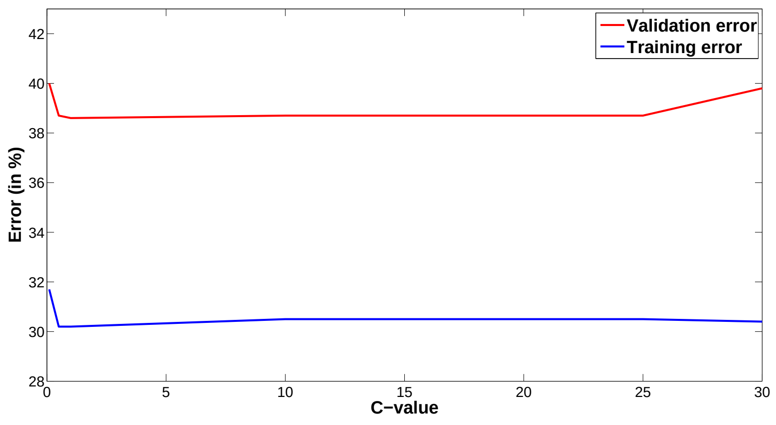


Figure 21: Validation - and training error vs. penalty parameter C.

An accuracy of 67% was obtained whit the SVM classifier when all features are used. The training is constant and the validation error is lowest when C = 1.

Table 21: SVM classification of the twitter data with our own features. Parameter values for the best classifier is obtained with penalty parameter = 30 and a linear kernel function. Model accuracy and confusion matrix.

	precision	recall	f1-score
Non-sarcastic	0.56	0.69	0.61
Sarcastic	0.61	0.48	0.54
avg / total	0.59	0.58	0.58

		Predicted class		
		Non-sarcastic	Sarcastic	
Actual class	Non-sarcastic	978	1067	2045
	Sarcastic	615	1340	1955
		1593	2407	

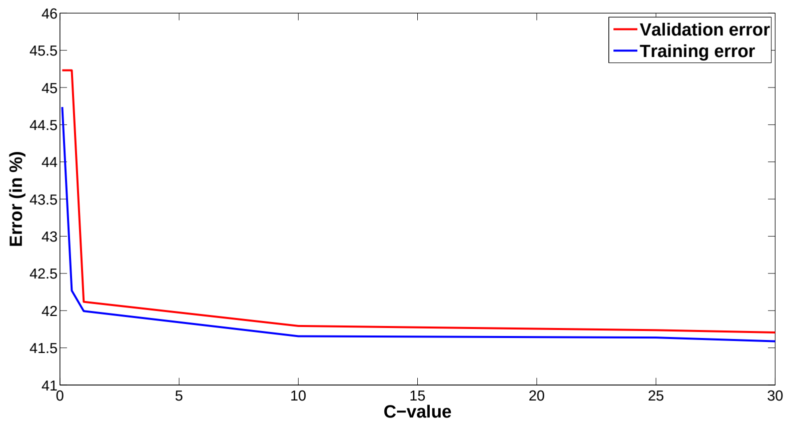


Figure 22: Validation - and training error vs. penalty parameter C.

An accuracy of 58% was obtained with the SVM classifier when our own features are used. The training - and validation error follows each other and the lowest validation error is obtained for C = 1.

6.4 Final model

The best classification on the Amazon reviews was performed by the SVM, with linear kernel and a penalty parameter $C = 5$ on all features. The accuracy was 87%. Table 22 presents the results for different kinds of combinations of the features.

Table 22: Final model for the Amazon dataset.

Features	F1-score	Precision	Recall
vocabulary	0.84	0.84	0.84
punctuation	0.72	0.79	0.74
sentiments	0.67	0.67	0.67
structure	0.51	0.55	0.53
vocabulary + punctuation	0.87	0.87	0.87
vocabulary + sentiments	0.87	0.88	0.87
vocabulary + structure	0.85	0.85	0.86
all features	0.87	0.87	0.87

The best classification on the tweets was performed by the Adaboost classifier, using a number of 250 estimators. The accuracy obtained was 71%.

Table 23: Final model for the Twitter dataset.

Features	F1-score	Precision	Recall
vocabulary	0.69	0.69	0.70
punctuation	0.61	0.61	0.61
sentiments	0.61	0.61	0.61
structure	0.57	0.57	0.57
vocabulary + punctuation	0.71	0.72	0.71
vocabulary + sentiments	0.70	0.70	0.70
vocabulary + structure	0.70	0.70	0.70
all features	0.71	0.71	0.71

7 Discussion and conclusions

7.1 Discussion

Table 22 shows that the punctuation features alone is the most important group for the Amazon dataset, apart from the vocabulary, obtaining an accuracy of 72%. For the tweets, the punctuation features and the sentiment features alone performed best apart from the vocabulary, obtaining an accuracy of 61%, Table 23. The structure features performed worst for both datasets. On the Amazon dataset they only manage to obtain an accuracy of 51%, which is similar to guessing randomly. The sentiment and punctuation features performed equally well combined with the vocabulary features obtaining an accuracy of 87%, improving the accuracy with 3%. For the Twitter data the punctuation features performed best combined with the vocabulary features improving the accuracy by 3%.

As expected, since our focus have been on the Amazon data, the results shows that the model is much more effective on the Amazon data compared to the tweets. We have disregarded many features that could have been useful for Twitter classification. Another cause for the poor result for the tweet could be the lack of manually annotated Twitter data. One might argue that it is up to the author to decide what is ironic or not, but the fact remains that it would be impossible to understand the sarcasm in many cases if the sarcasm hashtag were to be removed from the tweet.

We had expectations that interjections could be an interesting feature for irony detection based on the article by [Reyes et al., 2012]. Unfortunately the POS-tagger used in this master thesis failed to tag almost all of the interjections. A likely cause is that the Wall Street Journal, which the POS-tagger is trained on, contains few interjections. For that reason we cannot say anything about that specific feature.

7.2 Conclusions

In this master thesis we have built two models for classifying irony and sarcasm. The first is trained on Amazon product reviews and the second is trained on tweets. The models are able to process raw text as input and outputs whether the text is ironic or not. The final models obtain an accuracy of 87% and 71% for the Amazon products and the tweets reviews respectively tested on an unbiased dataset. The results were provided using the adaboost for the Twitter data and SVM with linear kernel and penalty parameter $C = 0.5$ for the Amazon data. All of our classification results had an accuracy over our proposed baseline on 50%. From this can we draw the conclusion thar the model worked for these specific data sets and could identify the majority of the ironic texts.

7.3 Future work

This model has so far only been tested on similar data. A first step for further work with this model is to test it on text collected from the Gavagai API. Problems with this is to get hold of annotated data that is sarcastic. We recommend that M-turks should be hired to do this time consuming work. The benefit by using these persons is that it is done in different steps so more than one person's opinion of what is sarcastic gets chosen in the end.

Other things that could be interesting to do is to look at the sentences separately and classify how sarcastic they are, e.g this sentence is 45% sarcastic.

However to find a general solution to this problem is very hard. But to build models that can find irony in a specific type of text is something that could be easier. E.g. one classifier that only classifies tweets from Twitter and one that only classifies news articles. This since text differ a lot depending on the context and language used. Also from the results is it showed that the classifiers perform differently on the data sets.

When looking at our results is it clear that the vocabulary is the most discriminant feature for finding irony in the data sets that has been used. So this is important for future work to test other features than we did together with the vocabulary to see if it can give the model an higher accuracy.

Finally we recommend that more research in the area should be done in this area. There is still a long way until a proper irony detector could be used in generally situations.

References

[Amazon, 2015] Amazon (2015). General review creation guidlines. `http://www.amazon.com/gp/community-help/customer-reviews-guidelines`.

[Banko and Brill, 2001] Banko, M. and Brill, E. (2001). Scaling to very very large corpora for natural language disambiguation.

[Brill, 1992] Brill, E. (1992). A simple rule-based part of speech tagger.

[Brown, 1980] Brown, R. (1980). The pragmatics of verbal irony. language use and the use of language.

[Butler, 1953] Butler, Q. . (1953). The instituto oratoria of quintilian. with an english translation.

[Carvalho et al., 2009] Carvalho, P., Silva, M. J., Sarmento, L., and de Oliveira, E. (2009). Clues for detecting irony in user-generated contents: Oh...!! it's "so easy" :-).

[Channon et al., 2004] Channon, S., Pellijeff, A., and Rule, A. (2004). Social cognition after head injury: sarcasm and theory of mind. *Brain and Language*.

[Chin, 2011] Chin, R. (2011). The science of sarcasm? yeah, right. `www.smithsonianmag.com/science-nature/the-science-of-sarcasm-yeah-right-25038/?no-ist`.

[Church, 1988] Church, K. W. (1988). A stochastic parts program and noun phrase parser for unrestricted text.

[Cover, 1965] Cover, T. (1965). *Geometrical and Statistical properties of systems of linear inequalities with applications in pattern recognition.*

[DeRose, 1988] DeRose, S. J. (1988). Grammatical category disambiguation by statistical optimization. pages 31–39.

[Domingos, 2012] Domingos, P. (2012). A few useful things to know about machine learning.

[Filatova, 2012] Filatova, E. (2012). Irony and sarcasm corpus generation and analysis using crowdsourcing.

[Fisher, 1936] Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*.

[Giora, 1995] Giora, R. (1995). On irony and negation. discourse processes.

[Grice, 1975] Grice, H. P. (1975). Logic ad conversation.

[Hastie et al., 2010] Hastie, T., Tibshirani, R., and Friedman, J. (2010). *The Element of Statistical Learning*. Springer.

[Hosch, 2013] Hosch, W. L. (2013). Machine learning. `http://global.britannica.com/technology/machine-learning`.

[Hosch, 2014] Hosch, W. L. (2014). Machine learning. `http://global.britannica.com/EBchecked/topic/1116194/machine-learning`.

[Inc, 2015] Inc, T. (2015). Info about twitter. `https://about.twitter.com/sv/company`.

[Kearns, 1988] Kearns, M. (1988). Thoughts on hypothesis boosting.

[Kearns, 2013] Kearns, M. (2013). Us secrete services seeks twitter sarcasm detector. `http://www.bbc.com/news/technology-27711109`.

[Michie et al., 1994] Michie, D., Spiegelhalter, D., and Taylor, C. (1994). *Machine Learning, Neural and Statistical Classification*. Broke Books.

[Mitchell, 1977] Mitchell, T. M. (1977). *Machine Learning*. McGraw-Hill.

[Nicolosi, 2008] Nicolosi, N. (2008). Feature selection methods for text classification.

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

[Pexman et al., 2013] Pexman, P. M., Nicholson, A., and Whalen, J. M. (2013). Children's processing of emotion in ironic language.

[Porter, 1980] Porter, M. F. (1980). An algorithm fo suffix stripping. *American Society for Information Science*.

[Reyes et al., 2012] Reyes, A., Rosso, P., and Veale, T. (2012). A multidimensional approach for detecting irony in twitter.

[Sayad, 2015] Sayad, D. S. (2015). Support vector machine - classification(svm). `http://www.saedsayad.com/support_vector_machine.htm`.

[Siegelhalter, 1994] Siegelhalter, M. . (1994). Machine learning, neural and statistical classification.

[Silver, 2014] Silver, J. (2014). Us secret service wants software to ”detect sarcasm” on social media. `http://arstechnica.com/tech-policy/2014/06/us-secret-service-wants-software-to-detect-sarcasm-on-social-media/`.

[Silver, 2012] Silver, N. (2012). *The signal and the noise: why most predictions fail - but some don't*. Penguin Press.

[Simon,] Simon, P. *Too Big to Ignore: The Business Case for Big Data*.

[Sivic, 2009] Sivic, J. (2009). Efficient visual search of videos cast as text retrieval. pages 591–605.

[Sperber and Wilson, 1995] Sperber, D. and Wilson, D. (1995). Postface to the second edition of relevance: Communication and cognition.

[Tepperman et al., 2006] Tepperman, J., Traum, D., and Narayanan, S. (2006). ”yeah right”: Sarcasm recognition for spoken dialogue systems. *In Proceedings of InterSpeech*.

[Tsur et al., 2010] Tsur, O., Davidov, D., and Rappoport, A. (2010). Semi-supervised recognition of sarcastic sentences in twitter and amazon. *CoNLL ’10 Proceedings of the Fourteenth Conference on Computational Natural Language Learning*.

[Witten and Frank, 2005] Witten, I. H. and Frank, E. (2005). Data mining: Practical machine learning tools and techniques, second edition.

[Zhu, 2007] Zhu, X. (2007). Semi-supervised learning tutorial. `http://pages.cs.wisc.edu/~jerryzhu/pub/sslicml07.pdf`.