EXAMENSARBETE INOM TEKNIK,
GRUNDNIVÅ, 15 HP
*STOCKHOLM, SVERIGE 2018*

# Predicting like-ratio on YouTube videos using sentiment analysis on comments

**MARTIN HYBERG**

**TEODOR ISAACS**

**KTH**
**SKOLAN FÖR ELEKTROTEKNIK OCH DATAVETENSKAP**

# Predicting like-ratio on YouTube videos using sentiment analysis on comments

MARTIN HYBERG & TEODOR ISAACS

Master in Computer Science
Date: June 6, 2018
Supervisor: Richard Glassey
Examiner: Jeanette Hellgren Kotaleski
School of Electrical Engineering and Computer Science
Swedish title: Förutspå gilla-förhållandet på YouTube-videor med hjälp av sentimentanalys av kommentarer

## Abstract

Social media is huge today. It allows anyone with an internet connection to voice their opinion all over the world with a single click. Many of the biggest social media platforms such as Facebook and Twitter do not allow users to leave negative feedback on posts. YouTube is different, it allows users to leave negative feedback in the form of a *dislike* with the click of a button. Each video also has a comment field with comments regarding the video. With the data that YouTube videos provide and sentiment analysis, the research question for this paper is: *Can the comments on a YouTube video be used to determine what ratio of the viewers liked or disliked the video using sentiment analysis?*. The results from this work showed that there is a weak correlation between the percentage of likes and the percentage of positive comments on a video. Because the fluctuation in the data and results, it is not accurate enough to be applied to any comment field on the internet, but it could be used as an indication. Many areas of the method could be improved for better results.

## Sammanfattning

Sociala medier är enorma idag. De gör det möjligt för vem som helst med en internetuppkoppling att uttrycka sin åsikt över hela världen med ett enda klick. Men många av de största sociala medieplattformarna tillåter inte användaren att lämna negativ feedback på inlägg. Men YouTube är annorlunda, de tillåter användarna att lämna negativ feedback i form av en *ogillning* med ett enkelt klick. Varje video har även ett kommentarsfält med kommentarer på videon. Med de data som YouTube videor tillhandahåller och sentimentsanalys lyder forskningsfrågan för denna rapport: *Kan kommentarerna på en YouTube-video användas för att avgöra vilket förhållande tittarna tyckte om eller ogillade videon med hjälp av sentimentsanalys?*. Resultaten från detta arbete visade att det finns en svag korrelation mellan andelen gillningar och procentandelen positiva kommentarer på en video. På grund av fluktuationen i data och resultat är det inte tillräckligt för att tillämpas på alla kommentarsfält på internet, men det kan användas som en indikation. Många delar av metoden kan förbättras för ett bättre resultat.

# Contents

**1 Introduction** **1**
1.1 Research Question . . . 2
1.2 Scope and Objective . . . 2

**2 Background** **3**
2.1 Sentiment analysis . . . 3
2.2 Classification . . . 4
2.3 Classifying using machine learning . . . 5
2.4 Bag-of-words model . . . 5
2.5 Machine learning algorithms . . . 6
2.5.1 Support Vector Machine . . . 6
2.5.2 Logistic regression . . . 7
2.5.3 Naïve Bayes . . . 8
2.6 Pearson correlation . . . 8
2.7 YouTube . . . 9
2.8 Related work . . . 10
2.8.1 Polarity detection . . . 10
2.8.2 Sentiment analysis on YouTube comments . . . 11

**3 Method** **13**
3.1 Selecting videos to analyze . . . 13
3.2 Gathering data . . . 14
3.3 Training data . . . 14
3.4 Classifiers . . . 15
3.5 Analyzing Data . . . 15
3.6 Validation & accuracy . . . 16
3.7 Limitations . . . 16

**4 Results** **18**
4.1 Support vector machine . . . . . 18
4.2 Logistic regression . . . . . 19
4.3 Naïve Bayes . . . . . 19
4.4 Combined data . . . . . 20
4.5 Pearson Correlation Coefficients . . . . . 21

**5 Discussion** **22**
5.1 Describing our results . . . . . 22
5.2 Problems with our method . . . . . 23
5.2.1 Channels . . . . . 23
5.2.2 Training data . . . . . 23
5.2.3 Lack of preprocessing . . . . . 24
5.2.4 Inherent problem with comments . . . . . 24
5.3 Further Research . . . . . 25

**6 Conclusion** **26**

**Bibliography** **27**

**A Graphs from individual channels** **30**
A.1 SVM . . . . . 30
A.2 Logistic regression . . . . . 31
A.3 Naïve Bayes . . . . . 32

**B Source code** **34**

# Chapter 1

# Introduction

Social media is getting more and more important when talking about public opinions and public discourse in general. Thanks to social media everyone with a network connection has the opportunity to voice their opinions at a global scale regarding current events worldwide. It also allows individuals to comment and discuss specific subjects such as the contents of a video, blog post or news article in more detail. The ability to gather data about the public opinion has never been greater, and that information perhaps never been so sought after. The problem lies in analyzing thousands of comments without actually having to read them manually. The field of sentiment analysis is the answer to this problem.

Sentiment analysis is the study of analyzing people's opinions, attitudes and emotions from a written text. Sentiment analysis is a large field and is used to solve a lot of problems. Generally, those problems are all related to classifying text into different categories. It can be used for troll and spam detection, determining levels of subjectivity and objectivity, and classifying emotion, also known as polarity detection [3]. In order to analyze public opinion on a subject, polarity detection can be used to classify peoples comments on that subject into either positive or negative, thumbs up or thumbs down.

Many of the big platforms such as Facebook and Twitter does not allow dislikes on posts. This makes it difficult to evaluate the public opinion on a given post on those platforms at a glance. Translating comments into a simple like/dislike or positive/negative ratio on those platforms would go a long way as to be able to understand public opinion regarding a subject. There is however one major social media platform that does allow for both likes and dislikes, YouTube. YouTube is a platform that allows user to both comment and leave a like or dislike on almost all of its videos. This allows for both a way of easily identifying the overall sentiment of their videos, by comparing the amount of likes and dislikes, as well as a way to understand more nuanced opinions by reading the comments.

By comparing the emotional sentiment to the ratio between likes and dislikes on a YouTube video, one can analyze whether there is a correlation between those two on the YouTube platform. This could in turn mean that there could be such a correlation on other platforms as well, meaning that sentiment analysis could be used on platforms such as Facebook and Twitter to simulate a like to dislike ratio.

## 1.1 Research Question

Can the comments on a YouTube video be used to determine what ratio of the viewers liked or disliked the video using sentiment analysis?

## 1.2 Scope and Objective

The objective of this study is to investigate whether there is a correlation between the emotional sentiment of the comments on a YouTube video and the actual like to dislike ratio of the video. This will be done using already implemented machine learning approaches. It is not in the scope if this thesis to try to improve these tools. Only to investigate whether those which are already widely used today for other applications, can be applied to this specific problem.

# Chapter 2

# Background

Determining the emotional sentiment of YouTube comments is a sentiment analysis problem. Three different machine learning algorithms are used to classify each comment into two categories, positive or negative, in this report. In order to do this, this section of the report will begin by describing what sentiment analysis is. After that, classification will be described by giving examples of different approaches. The machine learning algorithms which are used as the classifiers will be explained. Followed by describing Pearson correlation and the social media platform YouTube.

Finally, related work will be examined in order to understand how this report relates to previous work.

## 2.1 Sentiment analysis

Sentiment analysis is the study of analyzing people's opinions, attitudes and emotions from a written text[10]. Generally it tries to classify a piece of text into different categories, objective or subjective, spam or not spam, or positive or negative. Determining whether a written text is positive or negative is called polarity detection. This is a very relevant field of study in the world of social media, since it can be used to gather the public opinion regarding different subjects.

There are a few different approaches to determine the sentiment of a text, from simple to very advanced. The most simple way of calculating the sentiment is by separating a piece of text into each of the words that it is made up of, and then give each of these words a sentiment value and calculate the average. This way of extracting the words from a text is called a "bag-of-words" approach, since the order of the words doesn't matter, only their frequency. This might seem like a trivial approach, though it can be improved greatly with different machine learning algorithms.

"This is amazingly bad" is a sentence which is easy for a person to tell is very negative. But analyzing the sentence using a simple bag-of-words approach might not give us the same sentiment. The words *this* and *is* doesn't carry any sentiment and won't affect the sentiment of the sentence. But the word *amazingly* is considered to be positive by itself and *bad* is considered negative. So using this bag-of-words and simple addition the sentence would be considered quite neutral, which we know intuitively isn't true.

A slightly more advanced approach is to take the order and word class of the words into account using a lexicon based approach. Using lexicon based approaches words can now be used as a multiplier instead of simply summarizing each words sentiment score[22]. This means that the word following *amazingly* will have its positive or negative sentiment multiplied with some constant associated with the word *amazingly*. In our case this results in the negative value of *bad* being multiplied with the constant of *amazingly* to give the whole sentence a more negative sentiment. This method can be advanced even further by taking all the word in a sentence and calculate how they affect each other. The further you go down this advancement ladder, the more computational time and power is needed.

Antoher approach is to turn it into a classifications problem. By applying machine learning to the bag-of-words approach you could find out the sentiment by giving it two possible classes, positive and negative.

## 2.2 Classification

Classification is the task of classifying an input into one of two or more discrete classes. An example of a classification problem with two classes is determining whether an image contains a cat or not. A classifier is a learning algorithm combined with a set of training data[1]. The classifier is trained using the training data which consists of already classified inputs. For example a set of pictures which are labeled with "contains cat" or "doesn't contain cat". The classifier can then be fed with new input and predict what the class of the new input should be[2].

## 2.3 Classifying using machine learning

A common way of creating a classifier for text is by using a machine learning algorithm. The algorithm will use the training data to find patterns between the given inputs and their sentiment. The problem with using text as an input for machine learning algorithms is that it doesn't work. Text has to be converted into numbers or vectors of some kind. This conversion is called feature extraction [7].

## 2.4 Bag-of-words model

One of the simplest and most popular ways of feature extraction is the bag-of-words (BoW) model mentioned earlier [7]. The BoW model is a representation of word occurrence in a given text. It contains two things, a vocabulary of all known words to the model, and a measure of which of those words occur in a given piece of text. It is called "bag-of-words" since it ignores the order of the words and only consider their occurrences. If we use the two sentences "Cats are very cute." and "I think about cute cats" as our entire corpus, it gives us the vocabulary: "cats", "are", "very", "cute", "i", "think", "about". The second sentence, "I think about cute cats", given this vocabulary would then be represented as table 2.1:

Table 2.1: Vector representation of a sentence

| cats | are | very | cute | I | think | about |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 1 | 1 | 1 |

Which is the vector (1, 0, 0, 1, 1, 1, 1). However, most practical applications of the BoW model will give us vectors with vastly larger dimensions since it needs to contain all words known to the model. Thus, a vector representation of our sentence, "I think about cute cats", given a corpus of all the words in the Oxford Dictionary would have 171,476 [8] dimensions where all except 5 values are "0".

## 2.5 Machine learning algorithms

### 2.5.1 Support Vector Machine

Support Vector machines(SVM) are a non probabilistic classification algorithm, meaning it is not based on probabilities. Instead it is based on separating two classes with a hyperplane with the largest possible margin between the classes[19]. In two dimensions, two classes can be separated with a line. An example of this are the $\times$ and $+$ classes, separated by the dotted line in the figure 2.1.

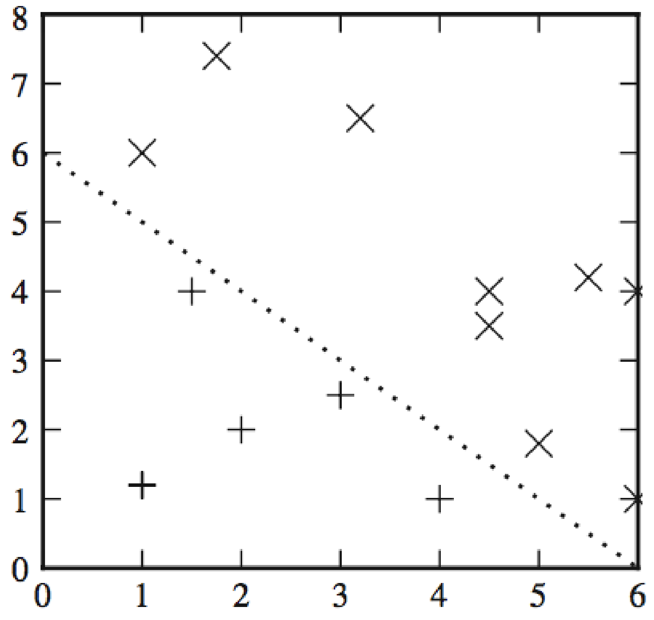


Figure 2.1: Source: [19]

In higher dimensions this is much harder to visualize, but the idea is the same: separate the two classes with the hyperplane with the largest possible margin.

Sometimes though, separating two classes with a linear hyperplane is not possible. An example of this in two dimensions is where one of the classes is located inside of a circle, and the other class outside of that circle. A solution to this is the so called kernel trick which is used to map the original inputs into a higher dimensional space where a linear hyperplane is able to separate them[19].

### 2.5.2 Logistic regression

Logistic regression is a probabilistic classification algorithm borrowed by machine learning from the field of statistics. It tries to find a relation between features and an classification[15]. This correlation is for one dimensional data a sigmoid function, which for some continuous variable, gives us the probability of having the given feature. As an example, given the hours studied, what is the probability of this person passing an exam. See figure 2.2.

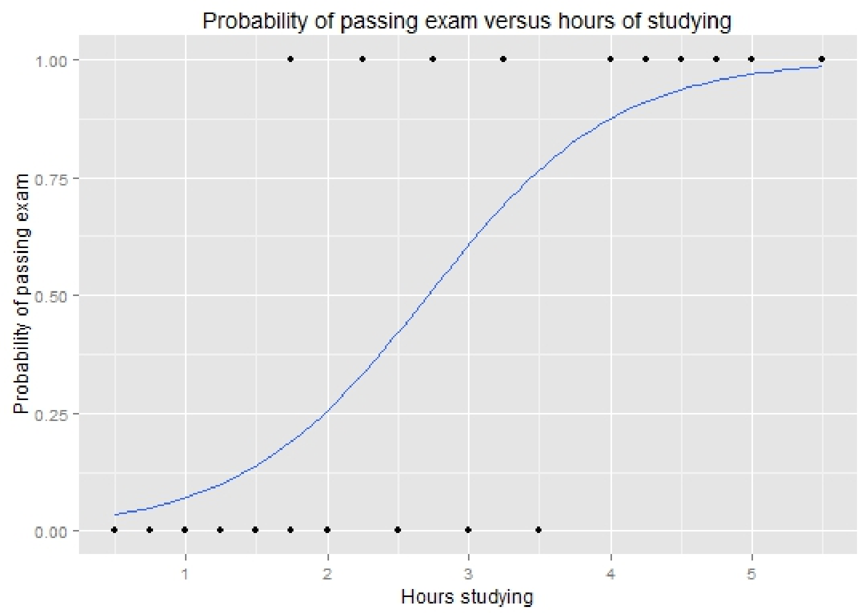


Figure 2.2: Source: [11]

The probability $p$ is then dependant on the two constants $b_0$ and $b_1$ which are the logistic regression coefficients. These coefficients $b_0$ and $b_1$ are what we are trying to determine when creating a logistic regression model for a given problem. If the probability output of this function for a given object is greater than 50% it will output 1, otherwise 0.

$$p = \frac{1}{1 + e^{-(b_0 + b_1 x)}}$$

Figure 2.3: Probability $p$ for logistic regression

This still works in higher dimensional space but instead of the s-shaped sigmoid function it is some equivalent for multi-dimensional space, and the coefficients increase by one for each dimension.

### 2.5.3 Naïve Bayes

Naïve Bayes is also a probabilistic classification algorithm. The algorithm creates an array of features and calculates a probability, the posterior probability, for each piece of text to be classified. One posterior probability is calculated for each class and the class with the highest probability gets assigned to the evaluated text.

To get an idea of how the posterior probability is calculated, observe the simplified formula[2] in figure 2.4:

$$Posterior = \frac{Prior \times Likelihood}{Evidence}$$

Figure 2.4: Equation for calculating the *Posterior* probability

Prior is the probability of a class. The total number of occurrences of a specific class divided by the total number of training data entries there is. It is called Prior due to the fact that it can be calculated before the rest of the data is analyzed and/or trained.

Likelihood is the probability of a word occurring in a training data document with the same class as in the prior-calculation.

Evidence is the marginal probability that a piece of text is seen, whether what class it gets labeled with. This has a normalization effect in order for all posteriors for a piece of text summing up to 1.

The algorithm must have a *Likelihood* probability for each feature for each class. The other parts of the calculations is done once for each class.

## 2.6 Pearson correlation

The correlation between two variables is called Pearson correlation or bivariate Correlation. It is a measurement of the linear correlation of two variables. The Pearson Correlation Coefficient is usually denoted as $p$ or $r$ for a sample. It is calculated for the variables $X$ and $Y$ with the following formula, figure 2.5:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Figure 2.5: The equation for Pearson correlation

In this formula $x$ and $y$ are the sample means and $x_i$ and $y_i$ are the values of the $i$:th instance of the respective variables. It is always a value between -1 and 1 where 1 is total positive correlation, -1 total negative correlation and 0 no correlation at all. A value between 0 and 1 indicates that as $X$ increases $Y$ tends to increase, and between 0 and -1 that as $X$ increases $Y$ instead tends to decrease[4].

In a scatter diagram of two variables, if there is a correlation between the variables the pattern of dots will resemble a line. The slope of this line is dependant on the correlation coefficient. A positive correlation coefficient gives positive slope and vice versa. If the pattern of plots in the scatter diagram strongly resemble a line, figure 2.6, it indicates that there is a strong correlation. If they are more spread out but still resembles a line, figure 2.7, there can be some correlation. If the dots are spread out equally over the graph, figure 2.8, shows that there is no correlation. [4].

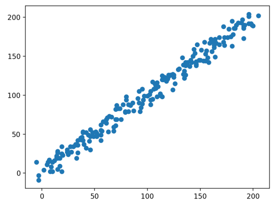


Figure 2.6: Strong correlation

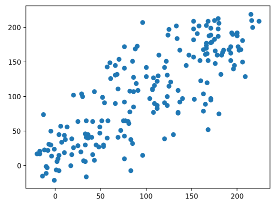


Figure 2.7: Some correlation

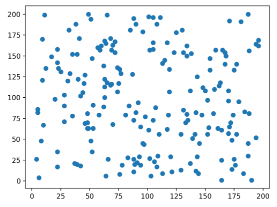


Figure 2.8: No correlation

## 2.7 YouTube

YouTube is an online video sharing service[**YouTube**]. They allows anyone to upload a video on their platform for everyone in the world to watch. On each video there is the option for the viewer to either *like* or *dislike* the video and the ability to leave a comment. The likes and dislikes are displayed as a ratio underneath the video, and below that a list of comments. There is also the ability to like or dislike a specific comment, or answer it with a comment of your own. This starts a thread of comments following the main comment. These comments on comments are not displayed until the user clicks on the main comment. No comment can go below 0 likes, a dislike will only affect a comments likes if they already have one or more likes. Disliking a comment with one or more likes subtracts a like from that comment.

## 2.8 Related work

Polarity detection using sentiment analysis is something that has been researched thoroughly in previous reports. Those reports overall show that is is indeed possible to create a accurate classifier. They also highlight what the many challenges of sentiment analysis are. In this section some notable previous reports on the topic of polarity detection in general will be discussed. As well as reports which take polarity detection into social media.

### 2.8.1 Polarity detection

One of the earliest works which researches polarity detection with modern machine learning classifiers is the paper *Thumbs up?: sentiment classification using machine learning techniques* by B.Pang[17].

Notable previous work according to B. Pang is one paper by Turney and Littman[23] which used statistical methods to determine individual words' emotional sentiment. The paper by B. Pang et al. compares emotional sentiment detection to topic detection. Topic detection had been done successfully earlier by Finn et al.[5].

B. Pang et al. used three machine learning classifiers to evaluate the emotional sentiment of movie reviews. They managed to create classifiers with up to 83% accuracy, the best one being a support vector machine classifier. They conclude that is it harder to evaluate sentiment than topic and had a few reasons why this was the case. Primarily that there are cases where a bag-of-words classifier would be fooled by *"many words indicative of the opposite sentiment to that of the entire review"* i.e. a lot of positive words but an overall negative sentiment or vice versa. This paper laid a lot of the groundwork in the field of sentiment analysis using machine learning algorithms. Other reports would later expand on this by improving the accuracy of classifiers by applying some sort of preprocessing, bringing sentiment analysis into the domains in which they widely used today.

One way to improve the accuracy of a classifier according to B. Pang and L. Lee[16] is to use so called "minimum cuts". The proposed technique would be implemented to reduce a piece of text into only the parts of it which are subjective. This is done by first training a classifier which determines if a sentence is subjective or not. Then, after using that to remove all objective sentences in a text, suitable algorithms can be used to determine the emotional sentiment. They conclude that not only can movie reviews be compressed into smaller pieces of text using this method, the accuracy of their classifiers increased using that method as well.

Another similar way of improving results is proposed by T. Wilson et al.[24]. Their approach is based on first determining whether the sentence is neutral or polar and then, if polar, what polarity that sentence has. With this approach they are able to automatically determine the sentences contextual polarity i.e. is the sentence positive, negative, both or neutral in the context of the analyzed document. Their findings are that this approach performs significantly better than baseline.

### 2.8.2 Sentiment analysis on YouTube comments

No earlier work on predicting a like/dislike ratio on YouTube videos based on comments was found while researching this topic. Though, research on YouTube comments has been published. Atte Oksanen et al. published a research paper on pro-anorexia and anti-anorexia videos on YouTube in 2015[14]. The aim of their study was study emotional reactions to these anorexia-topic videos using sentiment analysis. They analyzed the sentiments on comments in both pro- and anti-anorexia videos using ordinary least squares regression models. The results from the sentiment analysis show that anti-anorexia videos had both more positive comments and more likes than pro-anorexia videos. Similar to our work they count the comments with positive sentiment and base their conclusion on that. But unlike our comparison they are not trying to find a correlation between two sets of data based on their found emotional sentiment.

A paper by Peter Schultes et al. called *Leave a Comment! An In-Depth Analysis of User Comments on YouTube*[20] discusses the (bad) quality of YouTube comments. They write about the poor public image that YouTube comments has in social media, and that the users attach little or no value to the comments of a video. But the aim of the study was to use a comment classification approach that captures the salient aspects of YouTube comments. P. Schultes et al. found that their classifier is able to perform very fast lightweight semantic video analysis. And in addition to that, they find that a videos likes and dislikes are influenced by the distribution of valuable and invaluable comments.

The report *How useful are your comments?: Analyzing and Predicting YouTube Comments and Comment Ratings*[21] studies the correlation between comment sentiment and comment rating eg. like or dislike on the comment itself. Their study concluded that it is indeed possible to create a classifier that accurately predicts which comments are useful and which ones are not. Comments which are not deemed useful are ones that contain discriminatory language.

# Chapter 3

# Method

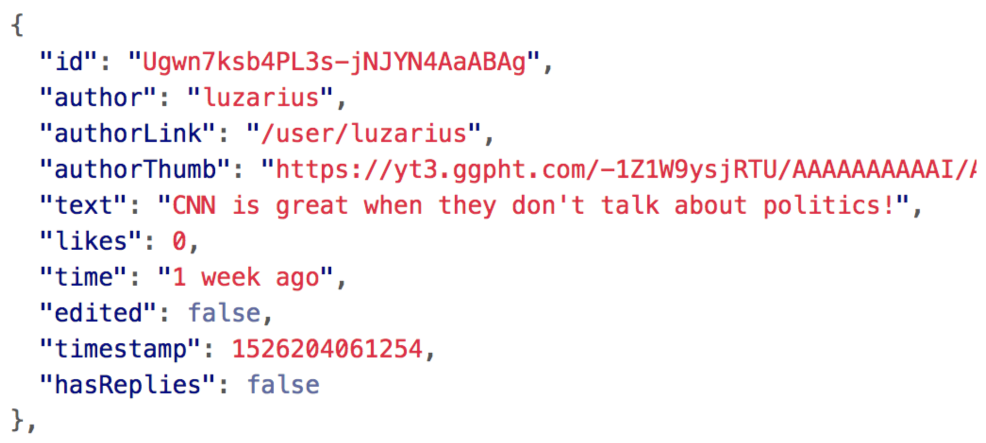
In this section the following steps will be explained: Firstly, how the videos to analyze were selected and how the comments from those videos were gathered. Secondly, what data was used to train the classifiers. Thirdly, why the selected machine learning algorithms were chosen and how those were used in the classifiers. Fourthly, how the classifiers were used to get the sentiment from our selected comments. And finally how we validated our work and results, and what limitations the work has.

## 3.1 Selecting videos to analyze

Since it was important to have videos with varying amount of likes to be able to perform our study, a few channels which are widely popular and fit this description were selected. These channels were consistently creating videos on controversial topics which made them prone to polarized comments. These channels were Vox, Vice, CNN, Fox News and TED Talks. Videos were split into two categories, videos with less than 10% dislikes and more than 10% dislikes, this was to get videos with varying amount of likes. And for each of those categories 10 videos were selected based on a few criteria, giving us 20 videos for each channel. The channels videos were sorted by most popular and had the requirement of having between 1000 and 10000 comments. The reason for the limitations is that the scraping of the comments would take too much time. Also that we wanted a range in the videos we analyzed, so that not all would have the same ratio between likes and dislikes.

## 3.2 Gathering data

To gather the comments we will use a NPM-package called YouTube-comment-scraper-cli[25]. It is a command line interface to scrape YouTube comments. It is used in order to scrape a videos comments to a JSON file with in the format below, figure 3.1. Only original comments will be scraped, no answers to comments or threads will be scraped.



```
{
  "id": "Ugwn7ksb4PL3s-jNJYN4AaABAg",
  "author": "luzarius",
  "authorLink": "/user/luzarius",
  "authorThumb": "https://yt3.ggpht.com/-1Z1W9ysjRTU/AAAAAAAAAAI/
  "text": "CNN is great when they don't talk about politics!",
  "likes": 0,
  "time": "1 week ago",
  "edited": false,
  "timestamp": 1526204061254,
  "hasReplies": false
},
```

Figure 3.1: Snippet of a comment in JSON

## 3.3 Training data

In order to classify YouTube comments into positive or negative emotional sentiment, training data is needed to train a classifier. It is important to have a comprehensive training data set to create as accurate predictions as possible. The training data used in this report consist of 1 600 000 classified tweets. Not all tweets were used to train our models, we set our limitations on when computational time became very long and when the accuracy stopped improving. The classification was binary with 1 being connected to positive tweets and 0 connected to negative tweets. It was fetched from sentiment140, a project which openly shares it's training data for students to use in research purposes[6]. Important to note is that this data has not been classified manually but automatically. The effect this has on our results is discussed in 3.7 and in 5.3.

## 3.4 Classifiers

The choice of the three algorithms used in this paper was based on that probabilistic algorithms and SVM are the most popular when in comes to short text classification[13].

As mentioned in 2.6, Naïve Bayes is a probabilistic algorithm and suitable for binomial classification. YouTube comments are short, often just a single or maybe a few sentences long. BoW models are inherently highly dimensional (big dictionary and a lot of features) and instances are very sparsely populated vectors of that space. Naïve Bayes is widely used due to its efficiency and ability to combine evidence from a large number of features[12].

SVM are also very commonly used when building and training classifications models. It is not only mentioned in [13] but also described in *learning to classify text using support vector machines*[9] as to have a state-of-the-art performance with classification problems.

In our research, logistic regression is not commonly used as a classifier on text input. Though it should be suitable since as we mention in 2.5.2, it is a bivariate classifier and the logic behind the formula is still valid in highly dimensional space. How this might affect results is discussed in 3.7.

## 3.5 Analyzing Data

To analyze our data we used the three text classification algorithms mentioned in 2.6. Each algorithm was trained with the same data set and was fed the same YouTube comments. All the comments from all the videos were classified as positive or negative.

For each video a positive comment percentage was created and compared to the total like percentage of the same video. Videos from the same channel where plotted in a graph to illustrate how the positive comment percentage and the total like percentage compared and how they correlated.

At the end of each script, when all the videos was classified, a graph for all the videos combined was generated. This gives a better opportunity to evaluate the algorithms side by side.

The Python library used to build and train our models came from Scikit-learn[18].

## 3.6 Validation & accuracy

To validate our results and to measure the accuracy of our classification the library `Model` selection from SciKit-Learn was used[18]. The method `train_test_split()` splits the arrays of training data into one train- and one test subset. 80% of the initial data and was be used to train the model. The other 20% was used as test after the training to get an accuracy score of the model.

The purpose of the the test data is to get a percentage of how accurate the trained model is. The model is fed with the test data and then classifies it. The classification of the test data is then compared to the actual class given to the test data to obtain percentage of how successful the model is.

## 3.7 Limitations

There are several limitations inherent to the method used in this report. The first and most significant is perhaps the training data used to train our classifiers. Since it was automatically generated it is difficult to determine its quality and correctness. This could potentially affect our results negatively since the quality of a machine learning based classifier is hugely dependant on the quality of the training data. Also, the use of logistic regression as a classifier is not common. This might result in the logistic regression classifier performing poorly relative to the other more widely used classifiers.

Another limitation is that the training data is gathered from Twitter, a completely different platform than YouTube. Although the size of a tweet is roughly similar to an average comment on YouTube, the different platforms could have a significant difference in how their users express themselves and their opinions. This difference in expression could also affect the results since the classifiers might only accurately recognize the difference between positive and negative tweets, and not between positive and negative YouTube comments.

The fact that there are only two classes is also a limitation. Many of the comments that is classified might not have a true positive or true negative sentiment. Many comments tend to be neutral or nonsensical. As was shown by B. Pang and L. Lee[16] and T. Wilson et al.[24] improvements to accuracy can be found by applying some kind of preprocessing to the data. Since no preprocessing is applied to our comments there will be neutral comments which are incorrectly classified as either positive or negative. This could skew our results significantly.

# Chapter 4

# Results

Here follows the results for our three different machine learning algorithms. Each YouTube channel is run through each of the algorithms.

## 4.1 Support vector machine

The support vector machine algorithm was trained on 150 000 tweets, had a accuracy of 75,14% and a Pearson correlation coefficient of 0.30. See figure 4.1.

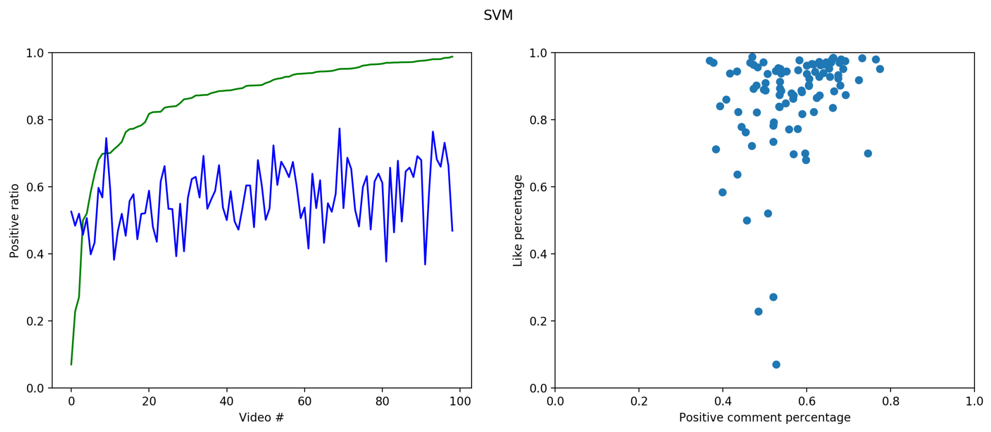


Figure 4.1: Graph and scatter plot for SVM

## 4.2 Logistic regression

The logistic regression algorithm was trained on 500 000 tweets, had an accuracy of 77,79% and a Pearson correlation coefficient of 0.34. See figure 4.2.

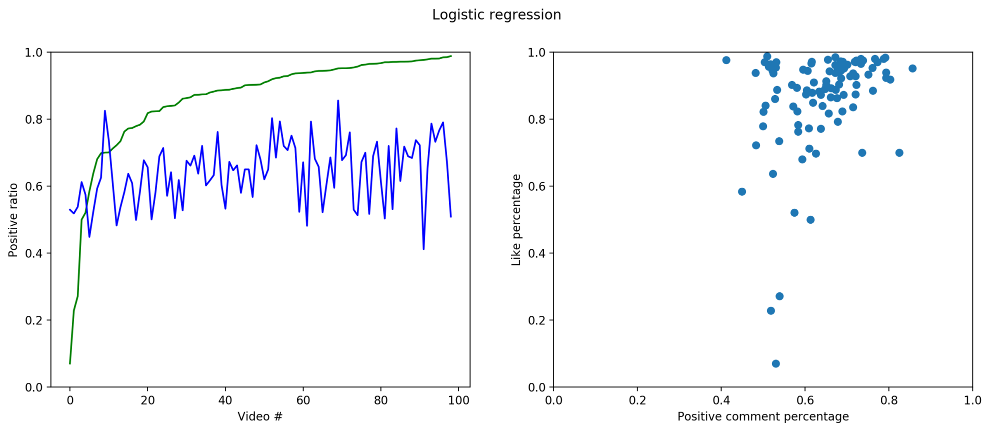


Figure 4.2: Graph and scatter plot for logistic regression

## 4.3 Naïve Bayes

The Naïve Bayes algorithm was trained on 1 000 000 tweets, had a accuracy of 77,47% and a Pearson correlation coefficient of 0.27. See figure 4.3.

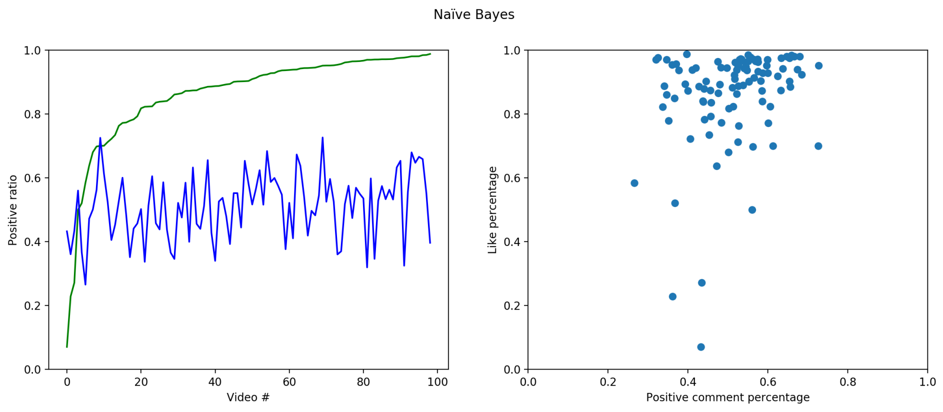


Figure 4.3: Graph and scatter plot for Naïve Bayes

## 4.4 Combined data

This graph, figure 4.4, contains the likes and emotional sentiment of all videos using every classifier. The lime green line is the like ratio. The other lines shows the overall emotional sentiment of video,each classifier is represented with one line. The videos are sorted by like percentage.

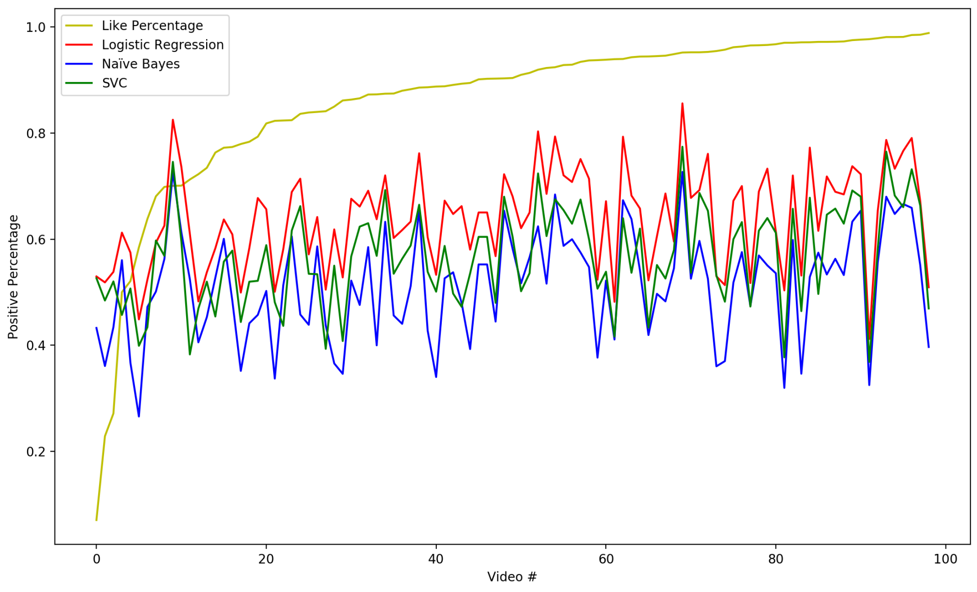


Figure 4.4: All algorithms combined to show the correlation

## 4.5 Pearson Correlation Coefficients

Pearson Correlation coefficients between comment sentiment and like/dislike for each algorithm and channel.

Table 4.1: Pearson correlation coefficients

Table 4.2: LR

| | |
|---|---|
| Vox | 0.39 |
| CNN | 0.25 |
| Vice | 0.59 |
| Ted Talks | -0.10 |
| Fox news | 0.28 |
| Alla | 0.34 |

Table 4.3: Naïve Bayes

| | |
|---|---|
| Vox | 0.25 |
| CNN | 0.17 |
| Vice | 0.62 |
| Ted Talks | -0.04 |
| Fox news | 0.16 |
| Alla | 0.27 |

Table 4.4: SVM

| | |
|---|---|
| Vox | 0.22 |
| CNN | 0.21 |
| Vice | 0.55 |
| Ted Talks | 0.12 |
| Fox news | 0.24 |
| Alla | 0.30 |

# Chapter 5

# Discussion

## 5.1 Describing our results

All our classifiers have a similar accuracy of about 77%, this is somewhat surprising since one could expect that the logistic regression classifier would perform poorly since it is not commonly used in sentiment analysis. Also surprising is that the SVM classifier performed worst out of all our classifiers (75,14%) since it has been widely used and proven to have state-of-the-art performance with classification problems [9] [13] [17] [20]. Though it should be noted that the SVM classifier was the classifier with the least training which might be the reason for it performing poorly, 150 000 as opposed to 500 000 and 1 000 000.

77% is a lot better than a guess (50%), but worse than even the early classifiers created by B Pang et al. [17]. Regarding correlation between likes and sentiment our results show an overall weak positive correlation with Pearson coefficients of 0.27, 0.30 and 0.34. This indicates that as the like ratio increases, the positive comment sentiment percentage tends to increase as well. But since the comment sentiment varies by up to 20% from video to video with approximately the same like percentage, it can not be deemed very reliable. Though this is not necessarily true for the channels individually. Ted Talks stands out with a correlation close to 0, which indicates close to no correlation at all, and Vice stands out with a high correlation at around 60 percent while the others hover at around 20-30 percent.

## 5.2 Problems with our method

### 5.2.1 Channels

This disparity between the channels is hard to explain, especially since the number of videos from each channel was quite low at 20 each. There might be an innate difference between the channels but that is hard to prove. This is because the comments on the videos we selected for a channel might not have been a perfect indication of that channel's comments as a whole. There is also the problem of channel selection. Almost certainly these channels are not a perfect representation of YouTube as a whole, which might give us skewed results. This point can be further stressed since all the channels have one thing in common, which is mentioned in our method, they all contain controversial and viral videos. Since we limited ourselves to only select the most popular videos on controversial channels this gives us a pretty small subset of YouTube videos. A larger selection of more diverse channels and videos would go a long way as to solving this problem.

### 5.2.2 Training data

The algorithms, which produced very similar results, have a hard time figuring out what differentiates a negative comment from a positive comment. This could be because of the training data. We are classifying YouTube comments with classifiers trained with tweets. Just like what was expressed in 3.7 there might be a big difference in language between YouTube comments and tweets which is making it hard for our classifiers to classify the comments correctly.

As we mentioned in the method, the training data consist of 1 600 000 automatically scored pieces of text. The quality of such a huge amount of automatically scored training data is not easily determined. Hence it is difficult to determine whether the training data itself is at fault for the poor results.

It is mentioned in 3.3 that we don't use all the data to train our models. The reason why is simply that we reached a point where the accuracy wasn't improving fast enough and computational time was getting too high. If we had more powerful computers we might have been able to utilize more of the training data to get more accurate classification. The different algorithms were trained with different amount of training data. The algorithms differ in the time it takes to train with a fixed number of data. For example, training a binomial Naïve Bayes model with 1 000 000 test documents takes the same amount of time as training a logistic regression model with 500 000. This is why the amount of training data for each classifier is different, this could have been a problem, but as we can clearly see by the results this does not seem to have mattered significantly.

### 5.2.3 Lack of preprocessing

The classified comments were not preprocessed in any way.One way to improve the accuracy of our classifiers would be to preprocess comments as mentioned in 2.8.1 [16][24]. Many comments could be classified as neutral, but the fact that the classifiers in this research only had two classes might have been the reason for a big grey-zone of comments. So many of the comments that actually did not have anything to do with the sentiment of the video might have caused the big fluctuation in the results. Filtering out junk comments would also improve the results. Junk comments that isn't related to the subject of the video will also skew the output from the classifiers.

### 5.2.4 Inherent problem with comments

One interesting observation that can be made on the most liked video on the CNN channel is that it had a surprising amount of negative comments. The video is about a young child getting a successful heart transplant. Most of the comments are viewers expressing how sad they are and how much they are crying. The models are reading the words *sad* and *crying* and classifying them as negative. The viewers have clearly liked the content of the video but are not expressing that in a way the model can understand. The model can't distinguish between good sadness and bad sadness. If a video is supposed to be sad, sadness expressed in the comments indicates that the video is good. On the other hand, if a video is supposed to be fun and lighthearted, the complete opposite of this is true. Since it is impossible for our model to understand the purpose of a video, the distinction between good sadness and bad sadness is impossible to make. This is a inherent problem that the models face and will make the results harder to interpret as the results are not able to show the true sentiment of the comments.

## 5.3 Further Research

If we were to continue this research and improve upon it, we could take the following points into consideration.

Improving the training data. The training data used in this report is one of the big flaws. The main issue of the training data was that it was a twitter training data set. Even though tweets are microblog posts and similar to YouTube comments in length, the language is very different. The difference in language can cause the model to wrongly classify comments. Also, the comments have been automatically generated which mean that they are not guaranteed to be completely accurate.

Adding another class to our models. Neutral comment was in this work classified as either positive or negative. Adding another class would catch a lot of junk comments, comments without sentiment and comments not related to the video. If those factors were excluded from our analyzed data it would give us a more accurate result. And since these removed comments had nothing to do with the video, they wouldn't affect the results very much.

More videos to analyze. Since only 5 channels and 100 videos were selected the overall results are not necessarily indicative of YouTube as a whole. They were all channels with controversial topics and all had a lot of views making them a small subset of the platform. Selecting a more diverse set of videos would give a more accurate result. More analyzed videos would give a more reliable conclusion not only because it is a increase in data, but also more reflective of the entirety of YouTube.

Improvements in these areas would significantly improve the overall results of this study.

# Chapter 6

# Conclusion

There is some correlation between the percentage of likes and percentage of positive comments on YouTube. Though, since the variation is very high it is not possible using the comment sentiment alone to accurately predict the percentage of likes using our method. The answer to the research question, *Can the comments on a YouTube video be used to determine what ratio of the viewers liked or disliked the video using sentiment analysis?* is difficult to answer using only our results. The method needs to improve in order to draw any substantial conclusions. Most importantly does the training data need to improve, irrelevant comments be sorted out and the amount of videos to analyze increase. If these areas are improved upon, a conclusive answer to our research question could be found.

# Bibliography

[1] *A Practical Guide to Sentiment Analysis*. eng. Socio-Affective Computing, 5. 2017. ISBN: 3-319-55394-1.

[2] Ethem Alpaydin. *Introduction to machine learning*. eng. Third edition.. Adaptive computation and machine learning. 2014. ISBN: 0-262-32574-8.

[3] E. Cambria et al. "New Avenues in Opinion Mining and Sentiment Analysis". eng. In: *Intelligent Systems, IEEE* 28.2 (Mar. 2013), pp. 15–21. ISSN: 1541-1672.

[4] Michael R. Chernick. "Correlation, Regression, and Logistic Regression". eng. In: *The Essentials of Biostatistics for Physicians, Nurses, and Clinicians*. Hoboken, NJ, USA: John Wiley and Sons, Inc., Oct. 2011, pp. 95–126. ISBN: 9780470641859.

[5] A Finn, N Kushmerick, and B Smyth. "Genre classification and domain transfer for information filtering". English. In: *Advances In Information Retrieval* 2291 (2002), pp. 353–362. ISSN: 0302-9743.

[6] *For Academics - Sentiment140*. 2018. URL: `http://help.sentiment140.com/for-students` (visited on 05/27/2018).

[7] Yoav Goldberg. *Neural network methods for natural language processing*. eng. S.l.], 2017. ISBN: 9781627052986. URL: `http://portal.igpublish.com/iglibrary/search/MCPB0000900.html`.

[8] *How many words are there in the English language*. 2018. URL: `https://en.oxforddictionaries.com/explore/how-many-words-are-there-in-the-english-language/` (visited on 05/29/2018).

[9] Thorsten Joachims. *Learning to classify text using support vector machines: Methods, theory and algorithms*. Vol. 186. Kluwer Academic Publishers Norwell, 2002.

[10] Bing Liu. *Sentiment analysis and opinion mining*. eng. Synthesis digital library of engineering and computer science. San Rafael, Calif. (1537 Fourth Street, San Rafael, CA 94901 USA): Morgan and Claypool, 2012. ISBN: 1-60845-885-7.

[11] *Logistic Regression Image*. 2015. URL: https://commons.wikimedia.org/wiki/File:Exam_pass_logistic_curve.jpeg (visited on 05/23/2018).

[12] Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[13] Preslav Nakov et al. "Developing a successful SemEval task in sentiment analysis of Twitter and other social media texts". In: *Language Resources and Evaluation* 50.1 (Mar. 2016), pp. 35–65. ISSN: 1574-0218. DOI: 10.1007/s10579-015-9328-1. URL: https://doi.org/10.1007/s10579-015-9328-1.

[14] A Oksanen et al. "Pro-Anorexia and Anti-Pro-Anorexia Videos on YouTube: Sentiment Analysis of User Responses". English. In: *Journal Of Medical Internet Research* 17.11 (Nov. 2015). ISSN: 1438-8871.

[15] Fred C Pampel. *Logistic regression a primer*. eng. Sage university papers series. Logistic regression. Thousand Oaks, CA: Sage Publications, 2000. ISBN: 1-4522-0761-5.

[16] Bo Pang and Lillian Lee. "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts". In: (Sept. 2004).

[17] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs Up?: Sentiment Classification Using Machine Learning Techniques". In: *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*. EMNLP '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 79–86. DOI: 10.3115/1118693.1118704. URL: https://doi.org/10.3115/1118693.1118704.

[18] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[19] Hans Georg Schaathun. "Support Vector Machines". eng. In: *Machine Learning in Image Steganalysis*. Chichester, UK: John Wiley and Sons, Ltd, Aug. 2012, pp. 179–196. ISBN: 9780470663059.

[20] Peter Schultes, Verena Dorner, and Franz Lehner. "Leave a Comment! An In-Depth Analysis of User Comments on YouTube." In: *Wirtschaftsinformatik* 42 (2013), pp. 659–673.

[21] Stefan Siersdorfer et al. "How Useful Are Your Comments?: Analyzing and Predicting Youtube Comments and Comment Ratings". In: *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. Raleigh, North Carolina, USA: ACM, 2010, pp. 891–900. ISBN: 978-1-60558-799-8. DOI: `10.1145/1772690.1772781`. URL: `http://doi.acm.org/10.1145/1772690.1772781`.

[22] M Taboada et al. "Lexicon-Based Methods for Sentiment Analysis". English. In: *Computational Linguistics* 37.2 (June 2011), pp. 267–307. ISSN: 0891-2017.

[23] Peter D. Turney and Michael L. Littman. "Unsupervised Learning of Semantic Orientation from a Hundred-Billion-Word Corpus". In: (Dec. 2002).

[24] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. "Recognizing Contextual Polarity: An Exploration of Features for Phrase-Level Sentiment Analysis". In: *Computational Linguistics* 35.3 (Sept. 2009), pp. 399–433. ISSN: 0891-2017.

[25] *YouTube-comment-scraper-cli*. 2018. URL: `https://www.npmjs.com/package/youtube-comment-scraper-cli` (visited on 05/19/2018).

# Appendix A

# Graphs from individual channels

## A.1 SVM

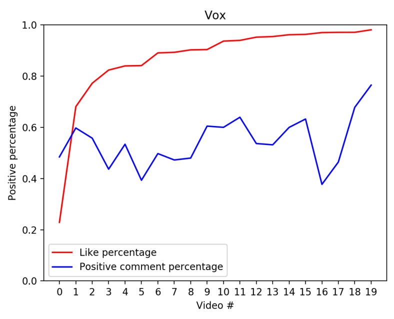


Figure A.1: Vox videos using SVM

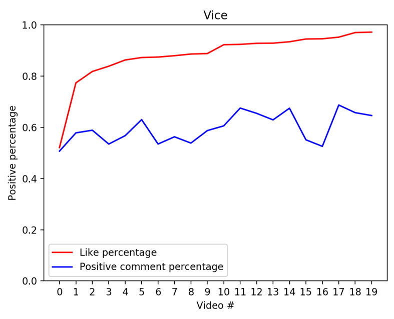


Figure A.2: Vice videos using SVM

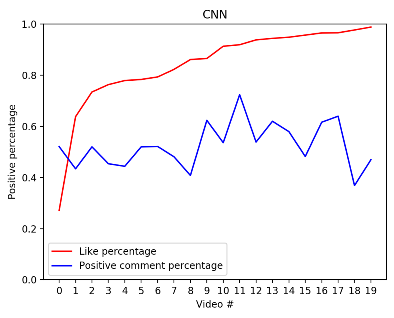


Figure A.3: CNN videos using SVM

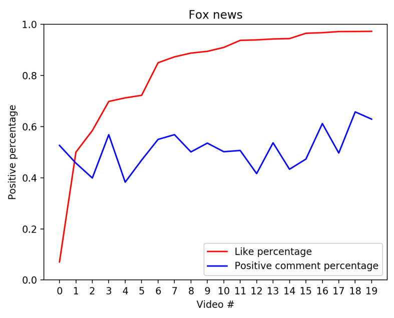


Figure A.4: Fox videos using SVM

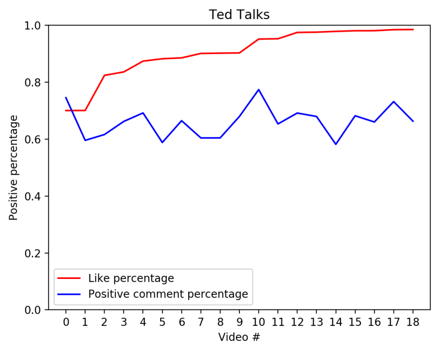


Figure A.5: Ted talks videos using SVM

## A.2 Logistic regression

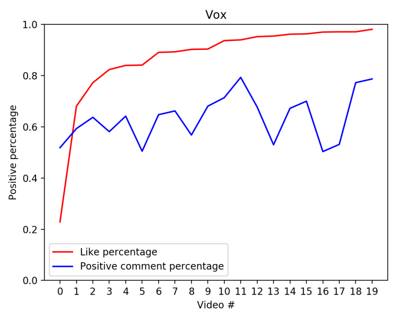


Figure A.6: Vox videos using LR

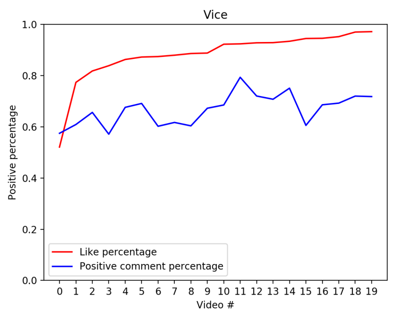


Figure A.7: Vice videos using LR

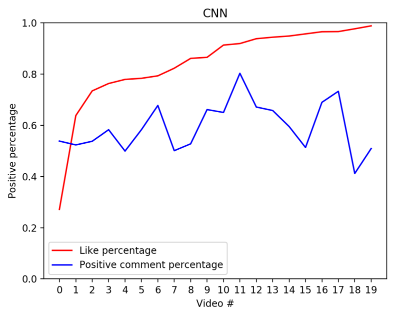


Figure A.8: CNN videos using LR

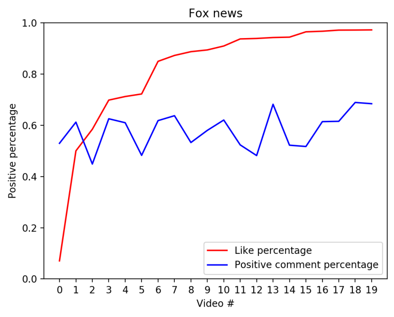


Figure A.9: Fox videos using LR

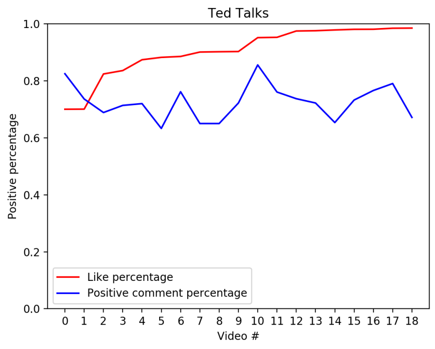


Figure A.10: Ted talks videos using LR

## A.3 Naïve Bayes

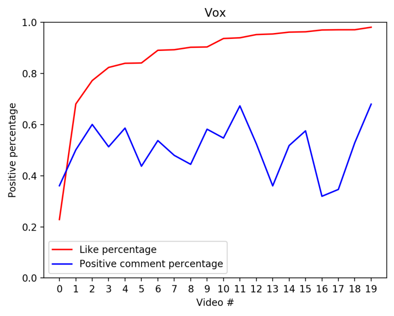


Figure A.11: Vox videos using naïve Bayes

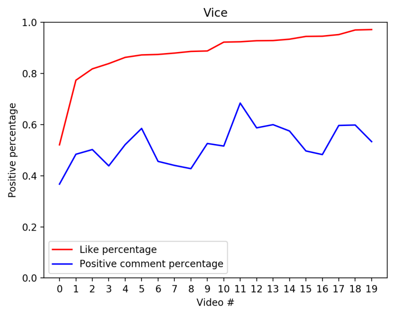


Figure A.12: Vice videos using naïve Bayes

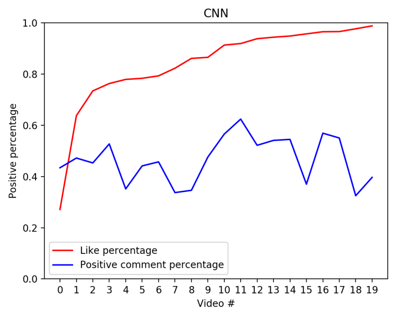


Figure A.13: CNN videos using naïve Bayes

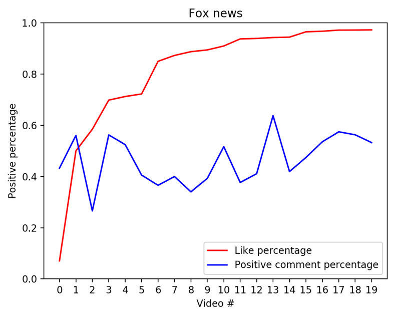


Figure A.14: Fox videos using naïve Bayes

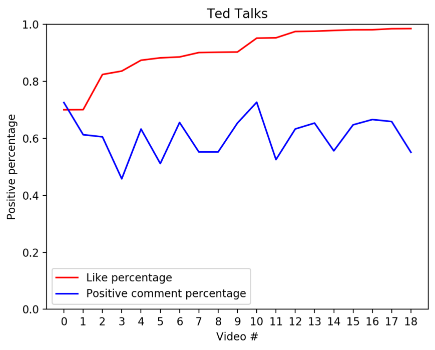


Figure A.15: Ted talks videos using naïve Bayes

# Appendix B

# Source code

```
from sklearn.feature_extraction. text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import csv
import json
import random
import glob
import numpy

data = []
data_sentiment = []

voxLikeRatio = [0.6806713818220205, ... , 0.9367713757769192]
viceLikeRatio = [0.862965397717086, ... , 0.9287232826608756]
tedTalkLikeRatio = [0.975995367, ... , 0.9652550822577964]
foxLikeRatio = [0.5001088455119367, ... , 0.9650676335239158]
cnnLikeRatio = [0.9659727499941237, ... , 0.27143643368189324]

# mste vara samma ordning som filnamns−arrayen
likeRatioList = [voxLikeRatio, viceLikeRatio, tedTalkLikeRatio,
 foxLikeRatio, cnnLikeRatio]
channelNames = ['Vox', 'Vice', 'Ted Talks', 'Fox news', 'CNN']
superList = []
```

```
#Byt algoritm har om du vill anvanda de andra
model = LogisticRegression()
#model = MultinomialNB()
#model = svm.LinearSVC()

with open('sentimentAnalysisDataset2.csv') as f:
    reader = csv.reader(f)
    i = 0
    for row in reader:
            i += 1
            data.append(row[3])
            data_sentiment.append(row[1])
            if i == 500000:
                break

vectorizer = CountVectorizer(
    analyzer = 'word',
    lowercase = False,
)

features = vectorizer.fit_transform(data)

# Trnar data med 80% av filen och sparar resten till validation
X_train, X_test, y_train, y_test = train_test_split (
        features,
        data_sentiment,
        train_size =0.80,
        random_state=1234)

model = model.fit(X=X_train, y=y_train) # Trnar modellen
y_pred = model.predict(X_test)

# Hmtar alla filer fr varje kanal
pathVox = '/Documents/Skola/Kex/vox/*.json'
pathFox = '/Documents/Skola/Kex/Fox/*.json'
pathVice = '/Documents/Skola/Kex/Vice/*.json'
pathTedTalks = '/Documents/Skola/Kex/TedTalks/*.json'
pathCNN = '/Documents/Skola/Kex/CNN/*.json'
filesVox=glob.glob(pathVox)
```

```
filesFox=glob.glob(pathFox)
filesVice =glob.glob(pathVice)
filesTedTalks=glob.glob(pathTedTalks)
filesCNN = glob.glob(pathCNN)

# mste  vara samma ordning som likeRatio−arrayen
files  = [filesVox,  filesVice,  filesTedTalks,  filesFox,  filesCNN]

keyErrorCounts = 0
likeRatioIndex = 0

for files in files:          #Loop fr  kanalerna
    commentRatiosTuples = []
    commentRatios = []
    for fileName in files:  #Loop fr  varje video
        justComments = []
        commentList = []
        print(fileName)
        commentsData = json.load(open(fileName))
        commentList = list(commentsData)

        #Extraherar kommentarerna ur JSONfilen
        for k in range(0,len(commentList)):
            try:
                comment = commentList[k]['text']
                justComments.append(comment)

            except KeyError:    #Hanterar tomma kommentarer
                keyErrorCounts += 1

        youtubeFeatures = vectorizer.transform(justComments)
        predictions = model.predict(youtubeFeatures)

        pos = 0
        neg = 0

        for k in range(0,len(justComments)):
            if predictions[k] == '1':
                pos += 1
            elif predictions[k] == '0':
                neg += 1
        commentRatio = pos/(pos+neg)
        commentRatiosTuples.append((fileName[-8:-5],
            commentRatio))

    commentRatiosTuples = sorted(commentRatiosTuples, key=
        lambda tup: tup[0]) #sorterar listan efter id p  videon
    for x in commentRatiosTuples: #Extraherar bara ration och tar
        bort id:et eftersom den nu r  sorterad
        commentRatios.append(x[1])

    commentRatioLikeRatioTuples = []
    for i in range(0, len(commentRatios)):
        commentRatioLikeRatioTuples.append((likeRatioList[
            likeRatioIndex][i], commentRatios[i]))

    for x in commentRatioLikeRatioTuples:
        superList.append(x)

    commentRatioLikeRatioTuples = sorted(
        commentRatioLikeRatioTuples, key=lambda tup: tup[0]) #
        sorterar efter like/dislike-ration och  hller  ihop motsvarande
        kommentarratio
    res1, res2 = zip(*commentRatioLikeRatioTuples) #gr tv  listor
          frn  tuplesena

    xValues = range(0, len(commentRatios))
    axes = plt.gca()
    axes.set_ylim ([0,1])

    print (numpy.corrcoef(res1,res2)) #Skriver ur korrelationen
    plt.plot(xValues, res1, 'r', label = 'Like percentage')
    plt.plot(xValues, res2, 'b', label = 'Positive comment
        percentage')
    plt.xticks(xValues)
    plt.ylabel('Positive percentage')
    plt.xlabel('Video #')
    plt.legend()
    plt. title (channelNames[likeRatioIndex])
    plt.show()

    likeRatioIndex += 1

superList = sorted(superList, key=lambda tup: tup[0])
superRes1, superRes2 = zip(*superList) # gr tv listor frn
    tuplesena

xValues2 = range(0, len(superList))

#Skriver ur korrelationen
print ('correlation: ', numpy.corrcoef(superRes1,superRes2))

axes = plt.gca()
axes.set_ylim ([0,1])
axes.set_xlim ([0,1])
plt. suptitle ('Logistic regression')
ax1 = plt.subplot(1,2,1)
ax1.plot(xValues2, SuperRes1, 'g')
ax1.plot(xValues2, superRes2, 'b')
ax1.set_ylim ([0,1])
plt.ylabel('Positive ratio')
plt.xlabel('Video #')

ax2 = plt.subplot(1,2,2)
ax2. scatter (superRes2, SuperRes1)
ax2.set_ylim ([0,1])
ax2.set_xlim ([0,1])
plt.ylabel('Like percentage')
plt.xlabel('Positive comment percentage')

plt.show()
print('accurary: ', accuracy_score(y_test, y_pred))
```

www.kth.se