MASTER'S THESIS | LUND UNIVERSITY 2017

Text classification of short messages

Anton Lundborg

Department of Computer Science
Faculty of Engineering LTH

ISSN 1650-2884
LU-CS-EX 2017-14

Text classification of short messages

(Detecting inappropriate comments in online user debates)

Anton Lundborg
dic12alu@student.lu.se

September 1, 2017

Master's thesis work carried out at Ifrågasätt Media AB.

Supervisors: Pierre Nugues pierre.nugues@cs.lth.se,
Gustav Hjärn, gustav@ifragasatt.se

Examiner: Jacek Malec, jacek.malec@cs.lth.se

Abstract

Almost every large Swedish online newspaper has disabled comments under their articles due to problems with hateful and offensive comments. In this Master's thesis, we explore different ways to detect toxic comments using machine learning. We carry out a comparison of classification algorithms and evaluate a number of different feature sets with the goal of optimizing accuracy for the classification of comments. We carry out the experiment with a manually labeled data set.

The best classifier was logistic regression with the f-score of 0.47 and recall of 0.50. We incorporated the classifier into a moderation tool for comments to help streamline the moderation process.

Keywords: Machine learning, text classification, hate speech, Linear classification, neural network, natural language processing, word2vec

Acknowledgements

I would like thanks Pierre Nugues for being so helpful and committed during this process. I also want to thank Gustav at Ifrågasätt for proposing this idea and providing all the resources needed for this project.

Contents

1 Introduction **7**
- 1.1 Background . . . 7
- 1.2 Related Work . . . 7
- 1.3 Goals . . . 8
- 1.4 Research Questions . . . 9
- 1.5 Constraints . . . 9

2 Theory **11**
- 2.1 Machine Learning . . . 11
  - 2.1.1 Linear Classification . . . 11
- 2.2 Algorithms . . . 12
  - 2.2.1 Support Vector Machines . . . 12
  - 2.2.2 Logistic Regression . . . 12
  - 2.2.3 Neural Networks . . . 12
- 2.3 Features . . . 14
  - 2.3.1 Bag-of-words Model . . . 14
  - 2.3.2 Character n-gram Model . . . 15
  - 2.3.3 Word2vec . . . 16
- 2.4 Evaluation Metrics . . . 16
  - 2.4.1 K-fold Cross Validation . . . 16
  - 2.4.2 Confusion Matrix . . . 16
  - 2.4.3 Precision and Recall . . . 17
  - 2.4.4 F-score . . . 17
  - 2.4.5 Inter-Rater Agreement . . . 18

3 Method **19**
- 3.1 Business Understanding . . . 20
- 3.2 Data Understanding and Preparation . . . 20
  - 3.2.1 Swedish Language data Set . . . 20
  - 3.2.2 Annotation . . . 21

3.2.3 English Language Data set . . . 22
3.3 Modeling - Feature Extraction and Engineering . . . 23
3.3.1 N-gram Features . . . 24
3.3.2 Linguistic Features . . . 24
3.3.3 Sentiment Feature – Version 1 . . . 24
3.3.4 Sentiment Feature – Version 2 . . . 25
3.3.5 Spellchecker Using SALDO . . . 26
3.3.6 Syntactic Features . . . 26
3.3.7 Embedding Derived Features . . . 27
3.3.8 Other Features . . . 27
3.3.9 Feature Selection and Elimination . . . 28
3.4 Evaluation Methods . . . 28

4 Results **31**
4.1 Annotation . . . 31
4.2 Feature Selection . . . 32
4.3 Classifier Comparison . . . 32
4.3.1 Neural Network . . . 32
4.4 Feature Comparison . . . 33
4.5 Comparing Data Sets . . . 34
4.6 Baseline Performance . . . 34
4.7 Tuning the Decision Limit . . . 34

5 Application **37**

6 Discussion **41**
6.1 The Annotation Process . . . 41
6.2 Machine Learning . . . 42
6.2.1 Features . . . 42
6.2.2 The Algorithms . . . 42
6.3 Applications . . . 42

7 Conclusion **45**
7.1 Future work . . . 45

Bibliography **47**

Chapter 1
Introduction

1.1 Background

Ifrågasätt is a company developing an online debating platform in the form of a plug-in module that can be included on websites. This platform is intended to be used by companies active in the media business, such as online newspapers, blogs and similar. In Sweden, almost every large newspaper has disabled comments under their articles due to problems with hateful and offensive comments. The idea of Ifrågasätt is to perform some kind of text analysis when a user publishes a comment, and decide instantly if the comment is either threatening, explicit, or inappropriate in some other way. The text analysis of comments is currently done by simply checking if the comment contains words from a predefined list of unsuitable or explicit words. The purpose of this Master's thesis is to explore ways to perform the analysis of comments in a more advanced way.

In this thesis, we have investigated the possibility to use natural language processing (NLP) and machine learning (ML) algorithms to identify inappropriate comments. The research covers the entire process: Collecting and annotating data, feature extraction and engineering, evaluating the performance of different algorithms, and lastly integrating the algorithm into the Ifrågasätt moderation tool to help streamline the moderation process.

1.2 Related Work

Yin et al. (2009) applied machine learning techniques to detect harassment online. In this paper, they define harassment as:

> Communication in which a user intentionally annoys one or more others in a web community.

Using a combination of term frequency–inverse document frequency(TF/IDF), sentiment and contextual features, they were able to achieve the f-score of 0.481. They suggest fur-

ther research to incorporate more types of features, such as user information, as well as improving on the ones used in their research.

Dinakar et al. (2011) performed ML experiments on Youtube comments trying to detect cyber-bullying. They find that binary classifiers for each individual topic (sexuality, race, intelligence) perform better than multiclass classifiers. They mention the fact that each comment is treated as an individual, possibly missing context from the related article and previous comments.

Dadvar et al. (2012) investigated whether gender information improves the accuracy detecting cyber-bullying. Their result shows that classification accuracy can improve by incorporating gender information. However, the improvement was not more than one percent for the f-score.

Dadvar et al. (2013) showed how user-based features can improve the accuracy. Examples of features used in the study are: history of user activity, frequency of profanity in previous comments by the user as well as the average length of comments and the age of the user.

The work presented by Nobata et al. (2016) is centered around finding the state-of-art method for detecting abusive user content online. They train a regression machine learning algorithm with years of labeled user comments from Yahoo news. Their result shows that a combination of word and character *n*-grams, linguistic, syntactic, and distributional semantics features yields the highest accuracy. However, character *n*-grams performs almost on par with all the features combined.

Mehdad and Tetreault (2016) performed experiments on the same data set as Nobata et al. (2016). They managed to improve the f-score by simply using a character *n*-gram model (1...5) and a support vector machine with Naive Bayes (NB) features as classifier. They used a Support vector Machine (SVM) variant using NB log-count ratios as feature values which consistently performed well across different tasks and data sets. They released an implementation in Python. [1]

In this thesis, manual annotation of comments is needed to be able to train a supervised machine learning algorithm. Annotation is a very time consuming and expensive process, which is why it is important to explore ways to speed up the annotation. Experiments on manual annotation made by Marcus et al. (1993) show a significant increase in annotation speed when providing annotators with automatically tagged data. The annotators are then asked to correct instead of tagging unlabeled data. The speed increase was as much as 20 minutes vs. 44 minutes per 1,000 words. The tagging process includes 87 simple tags which is far more than the four tags used in this our thesis. This probably explains why the increased efficiency was so large.

1.3 Goals

The goal of this thesis is to streamline the moderation process of user-generated content online by using machine learning algorithms. The idea is to have a method which requires nothing but previously annotated comments to function, i.e. no need to have updated lists of *bad words* or phrases. The algorithm should also perform better as the number of manually annotated comments increase, which is basically the whole idea of machine learning:

[1] https://github.com/mesnilgr/nbsvm/blob/master/nbsvm.py

Using knowledge about previous data to predict the things about new data. This thesis will explore ways to detect and sort comments based on how toxic they are. Hopefully, the results could contribute to similar application where real time text classification of short messages is needed.

1.4 Research Questions

The research questions we would like to answer can be summarized as:

- Which algorithms give the best accuracy for this problem?
- How can pre-processing and feature extraction help improve the accuracy?
- How can the final classifier be used to help streamline the moderation process and decrease the number of toxic comments online?

1.5 Constraints

Previous research has to a very large degree been centered around the English language, while in this thesis, we limit ourselves to the Swedish language. We are limited to a certain amount and quality of data, which in turn limits how well the algorithm can perform. To successfully train the algorithm, we need to annotate the data set. When annotating data, we need to look at how well two humans agree annotating the same data in order to know how much ambiguity there is in the data set. Probably the largest limiting factor is the quality of the annotated data set, where the data set is not only small, but also houses a lot of ambiguity and errors. Given more time and money, a more diverse set of comments could have been collected, as well as having annotators trained for the task perform the annotation. This would probably have given a better inter-annotator agreement score and data set with less errors.

The focus of the thesis is not limited to just develop and evaluate algorithms, but also to develop integrate the algorithm into a useful application. This wider focus means time has not only been spent on finding ways to improve the model performance, but also researching ways to integrate the classifier in an application.

Looking at different classifiers, we have not developed our own models, but instead used already implemented ones. We have limited ourselves to looking at the ones available through scikit-learn and Keras. Scikit-learn (Pedregosa et al., 2011) is a machine learning library for python including numerous classification algorithm suitable for this project. Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow[2]. Scikit-learn can be used in conjunction with the Keras classifiers making development quicker when one can reuse code.

[2]https://keras.io/

Chapter 2
Theory

2.1 Machine Learning

Machine learning tasks are usually divided into two categories: *supervised* and *unsupervised* learning. In supervised learning, the algorithm is given an input and an output and the goal is to find a mapping between the two which generalizes well to new input. In unsupervised learning, the algorithm is given only an input and the goal is to find some kind of structure in the data. For the problem in this report, we are looking at a supervised learning problem. The goal is to decide for new comments, which class they belong to.

Regardless of which machine learning algorithm is used to solve a problem, one needs to specify what the algorithm should consider as data. The observations in the training data are translated into quantitative or qualitative properties called *features*. Multiple features are combined into a feature vector which is used as input to the machine learning algorithm. The training data is translated into feature vectors which are fed into the algorithm with their corresponding class.

There is a large number of supervised learning algorithms available, and the ones evaluated in this thesis: logistic regression, support vector machine and neural networks, will be described in Section 2.2.

2.1.1 Linear Classification

In this thesis we evaluated three classifiers which are all linear classification algorithms. We first briefly describe what linear classification is.

Given a set of data points, each belonging to one of two classes, the goal is to decide to which class any given new data points belong. A linear classifier makes the decision based on the value of a linear combination of the feature vector. The linear classifiers can be seen as a function $y = f(w, x)$ which maps a set of inputs x to an output y, through a set of weights w. The algorithm tries to find a hyperplane (in the 2-dimensional case, a

line) which maximizes the separation between the two categories. The data points can be seen as p-dimensional vector where the $(p-1)$-dimensional hyperplane tries to separate the data points. The n data points with dimension p are represented by:

$$(\vec{x_1}, y_1), \ldots, (\vec{x_n}, y_n) \tag{2.1}$$

where each $\vec{x_i}$ represents a p-dimensional vector and each y_i is represented by either 0 or 1 indicating which class the point $\vec{x_i}$ belongs. We then want to find the hyperplane which separates the points $\vec{x_i}$ for which $y_i = 1$ from the ones where $y_i = 0$ with the maximum possible margin. For logistic regression and neural networks the convention is to use 1 or 0 to represent the two binary classes, but for SVM, the convention is to use 1 or -1.

2.2 Algorithms

2.2.1 Support Vector Machines

Support vector machines (SVM) are a form of supervised learning algorithm which can be used to solve classification and regression problems (Cortes and Vapnik, 1995). As shown by Joachims (1998), SVM tend to perform well for classification of text because of its ability to generalize into high dimensions, which is often the case with text categorization. SVMs are also known to perform well in cases where the number of features is greater than the number of training samples, which is the case in this thesis.

2.2.2 Logistic Regression

Logistic regression gives an output which is the probability that the given input points belong to a certain class. The output is most often used with a threshold value that decides which class a probability should be assigned to. This threshold value is usually 0.5, but can be set to other values in some applications. Logistic regression is also considered to be a linear classifier. Let us say we have two classes, 1, and 0, by providing a set of data points x, logistic regression can predict which of the classes 1, or 0 is the most probable one, or with other words, can perform classification with a probabilistic outcome. Finding the parameters which estimate the probability of the classes can be done in several different ways. When training the algorithms, one can use different optimization methods to fit the data to a model. The most appropriate optimization method to use depends on how large the data set is and how many features is used, but common ones used are: coordinate descent (implemented in liblinear[1]) and stochastic average gradient (SAG).

2.2.3 Neural Networks

A neural network can be described as a network of layers (input, hidden and output layers) where each layer consists of nodes. Figure 2.1 shows the structure of a neural network. All the nodes of the input layer are connected to all the nodes of the next hidden layer with adjustable weights. The neural network can be seen as a linear classifier if using a linear

[1] https://www.csie.ntu.edu.tw/ cjlin/liblinear/

activation function. Adjusting the weight according to a given set of x and y is what is referred to as *training the network*. Usually the training is done by randomly initializing the weights of the network and using the backpropagation algorithm to adjust the weights with the goal of minimizing the error of the output. The process can be split into two parts: propagation and weight updates.

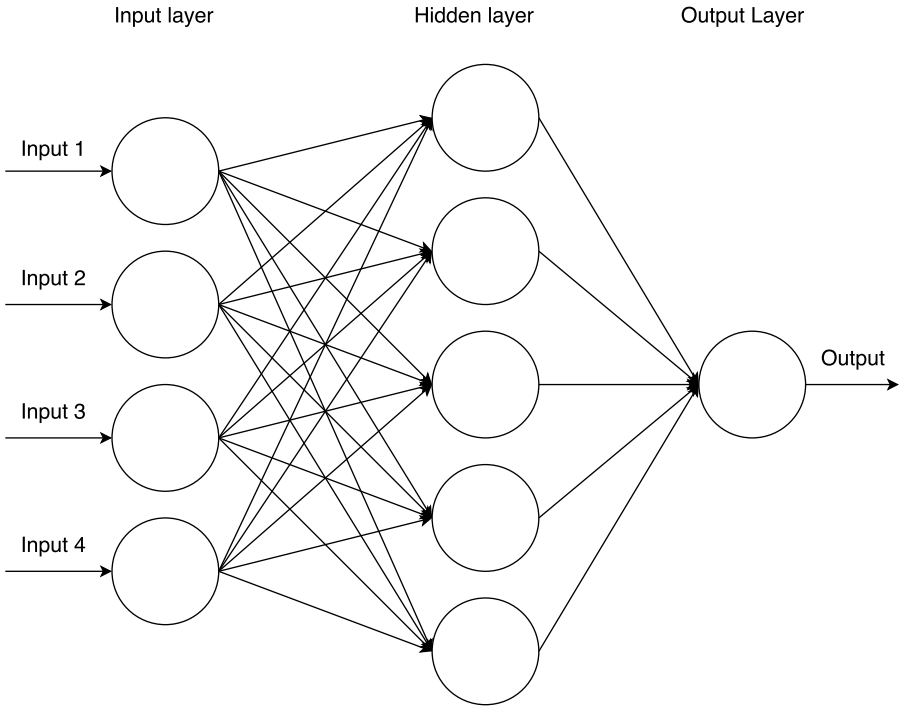

Figure 2.1: Neural Network

Propagation

This process can be repeated several times and one forward + backward propagation is sometimes referred to as *one epoch*. Before running the first epoch, one randomly initiate the weights of the neural network. For each propagation of the network:

1. Run a forward propagation, meaning, feeding the input through the network to generate the network predicted output(s).

2. Run a backward propagation, starting with the actual output and calculating the errors, called the deltas (predictions - actual values), for each weight.

Weight updates

For each weight of the network:

1. The weight's output is multiplied with the delta to get the gradient.

2. The gradient is multiplied with the learning rate and is subtracted from the weight.

The learning rate can be chosen arbitrary and will affect how quickly and accurately the model is learning. A large value will train the model faster, but a small value will do it more accurately.

2.3 Features

One of the most important aspects of machine learning is to decide which features to use. For various types of text classification problems similar to the one in this thesis, the most common baseline approach is the bag-of-words model.

2.3.1 Bag-of-words Model

The bag-of-words (BOW) model considers each message as a set of words that each occurs a certain number of times. The representation of the document is entirely orderless, as each word is treated independently of the previous and upcoming word. As an example, we have a data set consisting of only two messages:

> The cat is better than the dog

and:

> The weather is better than yesterday.

Table 2.1 shows the representation of the two vectors.

Table 2.1: Bag-of-words feature example

	the	cat	is	better	than	dog	weather	yesterday
Vector one	2	1	1	1	1	1	0	0
Vector two	1	0	1	1	1	0	1	1

As the number of samples grow, the number of unique words will increase. Since each unique word is represented by a specific position in the vector, these vectors will naturally grow larger as well. The vector will have the length of the total number of unique words that exists in the data set, but can also be limited to only include the X number of most common words of the data set.

N-gram model

The N-gram model accounts for word order by counting sequences of words, where N is the number of words to include in a sequence. By including sequences of words, we are able to account for a deeper meaning in the sentences and capture more nuances in text. If using the first sample of the previous example and set $N = 2$, we would get:

Table 2.2: Word *n*-grams feature example

the cat	cat is	is better	better than	than the	the dog
1	1	1	1	1	1

Syntactic *n*-grams

Syntactic *n*-grams are similar to BOW *n*-grams, but instead of creating sequences of word from words occurring next to each other, the sequences are given by the syntactic dependencies of the words in a sentence. The syntactic dependencies are given by a dependency parser. The result of such a parser can be visualized like in Figure 2.2

Figure 2.2: Syntactic dependencies

Using these dependencies, one could as an example create features from the word and its parent and a feature vector for the same example sentence as previously (The cat is better than the dog) would be:

Table 2.3: Syntactic dependencies feature example

the cat	cat is	is ROOT	better is	than better	the dog	dog than
1	1	1	1	1	1	1

2.3.2 Character n-gram Model

Character *n*-grams are similar to word *n*-grams, but instead of creating vector representations of words and word combinations, we create a vector representation based on characters and character combinations. As a result of this, one is able to more easily account for misspellings and obfuscated profanity words such as *idi0t*. As an example we use the text:

one dog, one cat

and 2-gram we would get the following feature vector (representing space-characters with _):

Table 2.4: Character *n*-grams feature example

on	ne	e_	_d	do	og	g,	_o	_c	ca	at
2	2	2	1	1	1	1	1	1	1	1

2.3.3 Word2vec

Word2vec is a word embedding technique presented by Mikolov et al. (2013), capable to capture degrees of similarity between words. In natural language processing, word embedding is the name of techniques used to map words or phrases to vectors of real numbers and word2vec is a technique capable of producing these vectors, where words with similar meaning or context will occur close to each other in the vector space. Word2vec can use two different models to produce the vectors, Continuous bag-of-word (CBOW) or Skip-gram, each of which has some different characteristics. CBOW uses the surrounding words but predictions do not depend on the order of these words. Skip-gram predicts the surrounding words based on the current word. According to Mikolov et al. (2013), the Skip-gram model overall increases the quality of the word vectors, especially in the semantic similarities and for less occurring words, but also increases the computational time needed by approximately a factor of 3.

The resulting word vectors have interesting and somewhat surprising properties. Looking at a word2vec model trained with Swedish twitter messages[2], the 5 closest vectors of the word *Stockholm* are: *sthlm*, *stockholm*, *Göteborg* and *Malmö* and *Uppsala*, meaning the distance in vector space somewhat captures the similarities between the cities (large cities in Sweden). Another example is the 5 closest vectors to the word *katt* which are: *hund*, *kanin*, *vovve*, *kisse* and *katten*. Words which often occur close to each other in sentences will have vectors located close to each other in the vector space.

2.4 Evaluation Metrics

2.4.1 K-fold Cross Validation

K-fold cross validation is a model validation technique used to assess how well the model generalizes to new independent data. For a machine learning problem, the data is usually split into two parts: One training set which is used to train the model and one one test set which is used to evaluate the accuracy/performance of the model. In k-fold cross validation, the data is split into k numbers of equally sized subsets. If we have k subsets, one of the subsets is used as a test set and the remaining $k - 1$ number of subsets are used as a training data. The process is repeated k times, meaning that each subset will be used as a test set once. The results from each fold can then be summed together into a final estimation.

2.4.2 Confusion Matrix

A confusion matrix can be used as a way to visualize the results of a classification algorithm. For the binary case where 1 and 0 is the two possible outcomes, the algorithm can be used to predict whether a test sample is either 0, or 1. As a way to measure how well the algorithm performs, we can count four different metrics, here 1 defined as positive and 0 defined as negative:

[2]https://mattiasostmar.wordpress.com/tag/word2vec/

1. True positive (TP), the algorithm classifies 1 where the correct class is 1.
2. False positive (FP), the algorithm classifies 1 where the correct class is 0.
3. True negative (TN), the algorithm classifies 0 where the correct class is 0.
4. False negative (FN), the algorithm classifies 0 where the correct class is 1.

The confusion matrix is simply these four values visualized in one table, see Figure 2.3

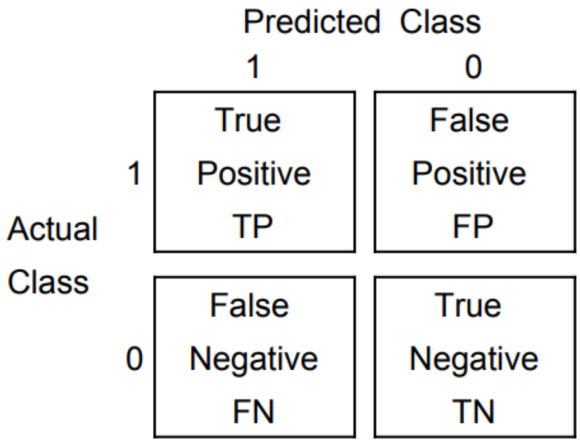

Figure 2.3: Confusion matrix

2.4.3 Precision and Recall

As a way to evaluate the performance of an machine learning algorithm, one can use precision and recall. Precision is defined as:

$$\text{precision} = \frac{TP}{TP + FP} \tag{2.2}$$

And recall defined as:

$$\text{recall} = \frac{TP}{TP + FN} \tag{2.3}$$

A very high precision means that the algorithm classifies almost no inputs as positive unless they are positive. A high recall would mean that the algorithm misses almost no positive values.

2.4.4 F-score

The f-score (or F_1-Score) is the harmonic mean of precision and recall. The f-score is defined as:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2.4}$$

This means that an algorithm with a precision of 1 and recall 0 would still get a f-score of 0.

Micro

Calculating the f-score with k-fold cross validation, it is possible to calculate the average f-score for the k folds in different ways. The results of the Forman and Scholz (2010) study shows that the method with the least bias is micro f1 where the true positives, false positives and false negatives from each fold are summed together and calculated by:

$$F_1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{2.5}$$

2.4.5 Inter-Rater Agreement

Inter-rater agreement is the degree to which two or more raters agree when rating or classifying something. It can be measured in several different ways, where the most basic way is joint probability of agreement:

$$accuracy = \frac{TP + TN}{Allratings} \tag{2.6}$$

The problem with joint probability is that is does not take into account that raters can agree purely by chance. To improve on joint probability, one can use Cohen's Kappa (Cohen, 1960) which is calculated like this:

$$\kappa = \frac{p_o - p_e}{1 - p_e}, \tag{2.7}$$

where p_o is the accuracy and p_e is the hypothetical probability of two raters agreeing. If two raters agree completely, the Kappa score would be 1 and if there is no agreement other than what can be expected by chance, the score would be 0.

Chapter 3

Method

Our process has to a large degree been exploratory, coming up with different ways to process the data, engineer features and evaluate them continuously throughout the process to ensure decisions are made based on results instead of gut feeling. Our work flow was mainly influenced by the the data mining process: *cross industry standard process for data mining (CRISP-DM)*, which can be split into 6 phases: Business understanding, data understanding, data preparation, modelling, evaluation and deployment. The process is iterative and can be seen in Figure 3.1. This chapter will be organized with sections corresponding to each step of the CRISP-DM process (with the exception of *Deployment* which is presented in Chapter 5, named Application).

Figure 3.1: The CRISP-DM Methodology (Jensen, 2012)

3.1 Business Understanding

The goal of Ifrågasätt is to provide an online debating platform where the moderation process can be handled more efficiently. Most online news papers have disabled comments on their websites because of problems with offensive, hateful comments and one part of Ifrågasätt's unique selling point is to solve the problems with toxic comment. It is therefore of greatest importance that Ifrågasätt can detect and remove the toxic comments as quickly and efficiently as possible. In conjunction with Ifrågasätt, we have identified the needs for them:

- Detecting the toxic comment before the user posts the comment, which makes the user aware of their potentially toxic comment. This way users are encouraged to change their comment before posting it, and the moderator load can be decreased.
- Prioritizing the work of the moderators by presenting moderators with the most toxic comments first. This way, the most severe comments can be handled before less severe ones.
- Improving the solution used currently by Ifrågasätt. The text analysis of comments is currently done by simply checking if the comment contains words from a predefined list of unsuitable or explicit words.

3.2 Data Understanding and Preparation

User generated content, like comments, forum post or tweets are available publicly in great numbers online. The problem with this content is that it is mainly un-annotated or that the toxic comments have already been removed, which mean it is not possible apply a supervised machine learning algorithm. To our knowledge, there is currently no public labeled data sets of toxic comments in Swedish language. To successfully experiment with different machine learning methods for detecting inappropriate comments, a labeled data set is needed to train and evaluate the accuracy of these methods. It is important to have a data set consisting of both good and bad examples, which can be hard to find online, as most *bad* comments are moderated and removed from online discussions.

Two different data sets were used; One Swedish language data set, manually labelled by two students at LTH and one in English language data set from a Kaggle competition *detecting insults in social commentary*[1]. The focus will be towards developing a model that works well for the Swedish language, but the English data set will be used as a way to evaluate the models multilingual capabilities as well as way to evaluate to what degree the data is affecting the accuracy of the algorithm.

3.2.1 Swedish Language data Set

When looking for potential unlabeled data sets online, one of the few sites which still allows comments was the Swedish blog / news site Avpixlat. This led us to collect comments from Avpixlat. This site has been described as a racist, hateful, xenophobic and

[1] https://www.kaggle.com/c/detecting-insults-in-social-commentary

right wing site by other Swedish, and international medias and it is one of the few Swedish news-websites which allow (anonymous) user comment on articles. Their policy on which comments are allowed is not very restrictive, meaning a lot of hateful and inappropriate comments are still available online. This fact is of great importance if one want to have a representative data set. Avpixlat use a commenting platform called Disqus[2], which provide API:s to download comments without spending to much time building an advanced HTML parser.

A set of 12,500 comments downloaded from articles written in the period 1 Jan 2017 - 15 Feb 2017. The data contains a lot of platform-specific information related to the user, e.g. the number of up- and down-votes. To ensure an algorithm which can be applied in other places than Disqus, all the information except the time stamp and the actual text of the comment was removed from the data set during pre-processing. Previous attempts have been made, trying to to benefit from data such as age, gender and how long the person has been registered as a user (Dadvar et al., 2013), but since this information is not present in the data set, we chose not to investigate this type of meta-data any further.

These comments are unlabeled and the training data must first be manually labeled to train a machine learning algorithm to detect bad comments.

3.2.2 Annotation

Classes

In order for a machine to accurately classify comments into different classes, the classes must be properly defined. As a baseline, straight up illegal comments should always be detected. However some comments might not be illegal, but would be considered inappropriate by most persons. To be able to differentiate between these comments, we defined four different categories: illegal, inappropriate, SPAM and OK, with the hope of capturing different types of toxic comments in the different categories.

According to Swedish law, some written comments can be considered unlawful. (Åklagarmyndigheten, 2016) describes four different types of crimes which can be expressed in text:

1. **Slander**, oral or written, occurs when someone describes another person as a criminal or leaves derogatory information about the person.

2. **Offensive** comments occurs when judgments or accusations is directed against a person.

3. **Hate speech** occurs when spreading public statements about a group of people with allusion to race, color, national or ethnic origin, religious belief or sexual orientation.

4. **Unlawful threats** occurs when the threatened person feel fear for their lives or health. This may include threats of personal assault.

These four crimes are all considered illegal by our definition. Inappropriate comments are comments that are not illegal but contains things like racism, sexism, hatred and bullying. The third category called SPAM includes comments which are incoherent, has bad

[2]https://disqus.com/

grammatical structure or could be considered general SPAM. The fourth category is called OK and essentially contains the remaining comments not beloning in any of the other three classes.

Manual annotation process

The annotation process was made during one day (8 hours) by two LTH students in their fourth year. They are not by any means educated or experienced within the area of comment moderation, but are still reasonably educated and skilled in the Swedish language. The annotators, called person A and person B were given the same set of 500 comments to annotate together with instructions defining what comments belong in each class. The purpose of letting them annotate the same comments were to be able to measure the inter-rater agreement, i.e to what degree their annotation decisions match. After they had finished the first 500 comments they were each given a set of 6,000 unique comments each to classify, which is far more comments than one would expect a person to annotate during one working day (but rather too many than too few). At the end of the day, a total of 3,448 unique comments had been annotated, where person A annotated 1,259 comments and person B annotated 2,189 comments during the same period of time.

The data set used to train the machine learning algorithms consists of all the uniquely annotated comments. The first 501 comments were annotated by both person A and B, which is why we chose only to include the 501 comments from person A in the final data set. The combination of these comments gave us a total of 3,949 comments.

Evaluation

We calculated the inter-annotator agreement between the two annotators using Cohen Kappa and f-score. The Cohen Kappa was calculated using the scikit-learn method *cohen_kappa_score*. When calculating the f-score, person A's set were chosen as the gold standard and person B's set were considered the classifier. The results were not as satisfactory as we hoped, which is why we tried to group the three *bad* categories (Illegal, Inappropriate and SPAM) as one category and calculated the scores for this binary annotation as well. These results were more acceptable to use for training.

Tool

To use the annotators time as efficiently as possible, we developed a basic graphical interface in which the annotators were able to classify the comments in a quick way, see Figure 3.2.

3.2.3 English Language Data set

The website *Kaggle* runs programming contests to crowdsource machine learning solutions, and usually makes data sets available for the public during and after the competition. From the competition *Detecting insults in social commentary* (Kaggle, 2012) Kaggle released a data set consisting of 6,591 labeled training data. In addition to the training data

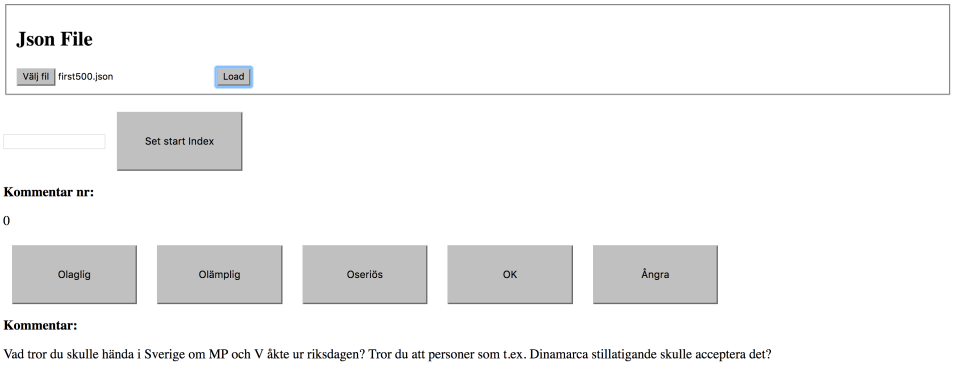

Figure 3.2: A graphical interface for comment annotation.

they also released a verification set of 2,232 labeled comments. The goal of the competition was to detect comments that are insulting to a person who is a part of the larger blog/forum conversation, but not insults directed to non-participants. Insults could contain profanity, racial slurs, or other offensive language, but often times, they don't. Comments which contain profanity or racial slurs, but are not necessarily insulting to another person are considered not insulting.

The two data sets were combined into one data set consisting of 8,823 labeled comments. The comments consist of the following information:

Table 3.1: Kaggle data point

Insult	Date	Comment
0 or 1	20120618192155Z	But how would you actually get the key out?

The label is either 0 meaning a neutral comment, or 1 meaning an insulting comment. Since the comments in the Kaggle data set are already labeled, there is no need to manually annotate the comments.

3.3 Modeling - Feature Extraction and Engineering

Finding the appropriate features for a ML-problem is no trivial task. The process often includes a lot of trial and error trying to identify what patterns in the data are representative of each class. Finding features is often considered an iterative process and in our case the process was exploratory.

We started by implementing a baseline, where previous work has shown that character *n*-grams are quite efficient features when identifying hate speech in English user online content (Nobata et al., 2016). In their discussion section, they mention that:

> Given how powerful the two *n*-gram features were in English, these would probably fare well in other languages given enough training data.

Nobata et al. (2016) also conclude that adding more features, such as linguistic, syntactic and distributional semantics features, can increase the f-score with a few percent, but the character *n*-gram still provides a very good baseline. This section will describe the different features and give some background to *why* they were tried. Some ideas for features were discarded at an early stage and are just mentioned briefly at the end of the section.

3.3.1 N-gram Features

We started out by trying different lengths of the *n*-grams, 1-3 word *n*-grams and 1-6 character *n*-grams. We also experimented with removing stop words, which are the most common words in a language and tried different maximum number of words or characters to use.

3.3.2 Linguistic Features

With linguistic features, the idea is to capture things like bad grammar, and patterns which are not captured by character or word *n*-grams. Looking at the data set, a lot of the comments were written with an incorrect sentence structure, and often contained unreasonably many or few punctuation. This is why the following features were crafted:

- Number of words and characters.
- Number of uppercase words and number of uppercase words per number of words.
- Number of uppercase characters and number of uppercase characters per number of characters.
- Longest word and average word length.
- Number of one letter tokens and number of one letter tokens per number of words.
- Number of punctuation, spaces, exclamation marks, question marks, at signs and commas.

When looking at which features are most indicative of being a toxic comment we found that the number of exclamation marks, number of one letter tokens and number of uppercase words per number of words were the three strongest ones. The strongest indicative of non-toxic comments were the number of words used in the comment.

3.3.3 Sentiment Feature – Version 1

The simplest approach to perform sentiment analysis is based on the assumption that certain words are more indicative of positive or negative sentiment. We therefore chose to explore a lexicon based approach similar to the baseline explored by Pang et al. (2002). They manually chose 7 positive and 7 negative words from the training and test sets which seemed representative for each polarity. The decision to whether a sample was positive or negative was then made by counting the number of words existing in each of the lists and simply considering a sample positive if the number of positive words are more than the

number of negative ones. The approach we chose is rather similar, but the lists of positive and negative words are much longer, and is not based on the training and test set. The list of words used were presented in a recent paper by Nusko et al. (2016) where they started with a small set of core words that is expanded semi-automatically using the lexical-semantic relations of the network structure in SALDO. SALDO is an extensive electronic lexicon for modern Swedish language.[3]The result is a lexicon consisting of 2,067 words or expressions with a polarity score (-4 to 4) (and a confidence score varying between 0.25 and 1). As a baseline version, we use a basic lexicon look up method where each word is stemmed and looked up in the lexicon. For all words found in the lexicon, we summed all the negative polarity scores and all the positive polarity scores. These two scores were then both divided with the number of words in the comment yielding two different features.

There are several problems with this approach, as it only captures the words found in this particular lexicon. The language used in social media is often filled with slang which is not found in this list. It would also fail to capture any negations like *not stupid* or intensifiers like *very stupid*. Also, a comment can be toxic without containing a lot of negative words, or the first sentence could be toxic and the remaining sentences in the comment could consist of a lot of positive words.

3.3.4 Sentiment Feature – Version 2

The second version is also based on the assumption that certain words are more indicative of positive or negative sentiment, but we are utilizing another set of words. The list of words consist of 1,000 negative and 1,000 positive words generated by Östmar (2016). These words have been generated using a word2vec model trained on 47 million Swedish tweets, using 150 dimensional vectors to represent the words. Words not occurring more than 30 times were discarded from the model, as well as two word expressions or names such as *inte bra* or *Svenska Dagbladet*. The list of negative and positive words were generated by asking the Word2Vec-model to find the 1,000 words, which have the closest distance in vector space to the 3 negative (idiot, svin, lögnare) and 3 positive words (förebild, hjälte, hjältinna).

The actual features implemented which use this data are the following:

- The number of negative words.
- The number of positive words.
- The number of positive words divided by the total number of words.
- The number of negative words divided by the total number of words.

Initial test showed that this using these sentiment features performed better than the first version

[3]https://spraakbanken.gu.se/resurs/saldo

3.3.5 Spellchecker Using SALDO

SALDO is a Swedish word association lexicon used in different types of language research. When looking through the data set, we found that a lot of the comments which were classified as *bad* had a lot of misspellings. We therefore wanted to find an easy way to measure the number of words which have not been correctly spelled and capture this property as a feature. A pretty simple way to measure the number of misspellings is to perform a look-up of each word in the SALDO lexicon. If the word does not exits, count it as a misspelling, otherwise, count it as a correctly spelled words. This approach is likely to not be 100% accurate, but will mostly likely capture the comments with a lot of misspelled words. We derived two different features from this method: The number of misspelled words and the number of misspelled words divided with the number of words in total.

3.3.6 Syntactic Features

As mentioned earlier, word *n*-grams can fail to capture the true relation between words, since the tuples created from the sentences are essentially the word and the word next to it, rather than the parent or child of the word. The idea is that crafting features using a dependency parser can capture more complex relations in a sentence, and complement a baseline solution using word or character *n*-grams.

We ran the entire data through the Langforia language analysis pipeline (Klang and Nugues, 2016) that produces a dependency tree which returns the parsed text in a tab separated format:

Table 3.2: Vilde data format

form	cpostag	feats	id	lemma	pos	head	deprel
I	ADP	_	1	i	PP	5	RA
den	DET	UTR|SIN|DEF	2	den	DT	4	DT
blå	ADJ	_	3	blå	JJ	4	AT
himlen	NOUN	UTR|SIN|DEF|NOM	4	himmel	NN	1	PA
bor	VERB	PRS|AKT	5	bo	VB	0	ROOT
fåglar	NOUN	UTR|PLU|IND|NOM	6	fågel	NN	5	SS

This information is best represented by a graph:

For the features we have built, the most important fields are the id, which gives the position of the word in the given sentence, pos which gives the *part of speech*, head which gives the id of the parent of the word.

All words were lower cased. Using this information, we crafted features consisting of different tuples. Concretely, the different tuples implemented can be seen in the list below, including examples of how the tuples would look like for the given example in Table 3.2 and Figure 3.3.

- **Word + Parent**, i_bor, den_himmel, blå_himmel, himmel_i, bo_ROOT, fågel_bo.
- **Word + Grandparent**, i_ROOT, den_i, blå_i, himmel_bo, bo_ROOT, fågel_ROOT.

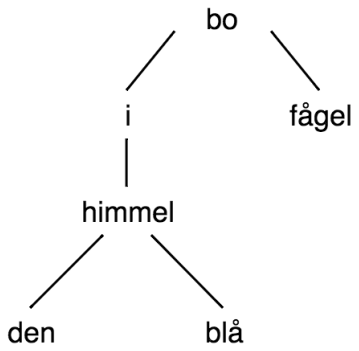

Figure 3.3: Dependency parsed data visualized in a graph.

- **Word + POS of parent**, i_VB, den_NN, blå_NN, himmel_PP, bo_ROOT, fågel_VB.
- **Word + POS of grandparent**, i_ROOT, den_PP, blå_PP, himmel_VB, bo_ROOT, fågel_ROOT.
- **Word + Children(s)**, i_himmel, den_, blå_, himmel_den_blå, bor_fågel, fågel_.

3.3.7 Embedding Derived Features

The same trained word2vec model as presented in Section 3.3.4 was used to craft these features. The idea here is that these vectors potentially carry some type of information about the words which are not captured by the other features. However the challenge is to find a way to design a feature based on the word2vec vectors. The first method we tried was as follows:

1. Split the comment into an array of words.
2. For each of the words, get the 150 dimensional vector from the word2vec model.
3. Sum all the vectors and divide with the number of vectors. This single 150 dimensional average of all the vectors was then used as a feature.

The second method:

1. Split the comment into an array of words.
2. For each of the words, get the 150 dimensional vector from the word2vec model.
3. For each of the vectors, calculate the sum of all the coordinates.
4. Pick the vector with the smallest and largest sum and average them. This single 150 dimensional average of the two vectors was then used as a feature.

3.3.8 Other Features

Other features have also been explored and discarded due to poor performance and will not be presented in the results.

Timestamp

We looked at using the time-stamp to create features measuring which day of the week the comment was posted as well as which hour of the day. However these features turned out to have the same accuracy as simply deciding on pure chance.

Bad words

The solution used by Ifrågasätt before exploring this thesis was to use a word-list of manually defined bad words and expressions. Using the same list of bad words, we tried to construct a feature which is simply the total number of bad words and expressions in the comment.

Extended bad words

We used the trained word2vec model to get the 5 word closest words in vector space to all the words in the bad word list. By extending the original list, the idea was to get a more complete list. However, by manually inspecting the list, we concluded that there is simply too much noise/errors for it to be useful as a feature.

3.3.9 Feature Selection and Elimination

When stacking several features, some feature will increase and other will decrease the accuracy. Feature selection methods can be used to identify and remove unnecessary or redundant features from the model. Using few features is usually desirable, since it reduces the complexity of the model (hence decreasing the time needed to train the model) and gives a better understanding of the model and data.

We implemented a forward selection algorithm to extract the best features. In Forward selection we tried each feature individually and chose the feature which give the best f-score. We then added the feature which improves the f-score the most until the f-score does not improve any more when adding features. This method is not guaranteed to give the global optimum, but will provide a reasonably good result without having to try every combination of features.

To further ensure we have the best combination of features, we perform a backward elimination on the features given by the forward selection. Backward elimination is basically the opposite of forward selection, removing one feature at the time and choosing the subset which improves the f-score the most until the f-score can't be improved further.

3.4 Evaluation Methods

Evaluation of different features and classifiers was done in several different ways:

- Measuring precision, recall and f-score using 10-fold cross validation.
- Plotting learning curves
- Computing the confusion matrix

The final classifier evaluation was performed by testing each feature individually, all together, and with the features from the feature selection with logistic regression and SVM. For the neural network, we used all features except the dependency parser features and experimented with one and two hidden layers. For the first layer, we tried 128, 256, 512, 1,024 and 2,048 nodes and for the second layer 0, 64, 128, 256, 512, 1,024 nodes yielding 5 * 6 = 30 different combinations. Each of these combinations were run with 5, 10, 20 and 50 epochs evaluating the f-score for each and every one of these.

For the best performing classifier, we experimented with the decision variable. This decision variable is usually set to 0.5, but by changing this variable, a different ratio between precision and recall can be achieved. Depending on the use case of the classifier, a tradeoff where i.e. a very good precision is needed but recall is not as important.

Chapter 4

Results

4.1 Annotation

Two persons annotated the same set of 501 comments. Measuring the multiclass inter-rate agreement of the two persons, the agreement rate is 0.820 and the Cohen's Kappa score is 0.466. If merging the three categories illegal, inappropriate and SPAM into one category, the agreement rate increased to 0.860 and the Cohen's kappa score increased to 0.553.

If considering the annotation of person A as the source of true labels and computing the f-score compared to the annotations of person B we get an f-score of 0.639. These results should be kept in mind when looking at the accuracy of the classifiers, since one could not expect to have a classifier performing better than the annotators. One could also note that the there is approximately 19% bad and 81% good comments in the data set used for training and evaluation of classifier performance.

Table 4.1: Annotation of identical 501 comments

	Person A	**Person B**
OK	402	406
Oseriös	36	32
Olämplig	48	40
Olaglig	15	23
Total Bad	99	95
Total	501	501

Table 4.2: Annotation of different comments

	Person A		**Person B**		**A + B**	
Total	1,259		2,189		3,949	
OK	884	70%	1,905	87%	3,191	81%
Oseriös	91	7%	43	2%	170	4%
Olämplig	257	20%	196	9%	501	13%
Olaglig	27	2%	46	2%	88	2%
Total Bad	375	30%	285	13%	759	19%

4.2 Feature Selection

The feature selection and backward elimination process yielded two different set of features for logistic regression and SVM.

Logistic regression

- Character *n*-grams
- Word2vec
- Spellchecker
- Sentiment 1

SVM

- Character *n*-grams
- Word2vec
- Word + Grandparent

4.3 Classifier Comparison

Logistic regression and SVM were evaluated with the same method. We ran forward selection and backward elimination for each of the classifiers individually to get the best features. The neural networks evaluations was made without the syntactic features since the input layer was simply too large to be able evaluate different layer sizes, numbers and epochs within a reasonable time frame. The precision, recall and f-score for each of the three classifiers is presented in Table 4.3.

4.3.1 Neural Network

The experiments with different number of layers, layer sizes and number of epochs resulted in a certain combination of these parameters performing best. The winning set of

Table 4.3: Classifier comparison

	precision	**recall**	**f-score**
Logistic regression	0.44	0.50	0.47
SVM	0.44	0.49	0.47
Neural network	0.44	0.43	0.44

parameters was a neural network with two layers, the first hidden layer with 1,024 nodes, the second hidden layer with 128 nodes and trained with 20 epochs.

4.4 Feature Comparison

The following results are tested on the Swedish language data set. In Table 4.4 each of the feature have been evaluated individually and is presented with precision, recall and f-score.

Table 4.4: Evaluation of all features individually, together and after feature selection

	Logistic regression			**SVM**		
	P	R	F1	P	R	F1
character *n*-gram 1-2	0.33	0.54	0.41	0.33	0.53	0.41
character *n*-gram 1-3	0.38	0.51	0.44	0.38	0.50	0.44
character *n*-gram 1-4	0.41	0.47	0.44	0.41	0.45	0.43
character *n*-gram 1-5	0.44	0.43	0.44	0.45	0.42	0.43
character *n*-gram 1-6	**0.45**	0.40	0.42	**0.46**	0.38	0.42
word *n*-gram 1-1	0.32	0.44	0.37	0.32	0.44	0.37
word *n*-gram 1-2	0.36	0.35	0.35	0.36	0.35	0.35
word *n*-gram 1-3	0.39	0.29	0.33	0.39	0.28	0.33
manual	0.27	0.39	0.32	0.28	0.29	0.28
sentiment 1	0.20	**0.64**	0.30	0.20	**0.64**	0.30
sentiment 2	0.28	0.56	0.37	0.28	0.52	0.37
spellchecker	0.25	0.35	0.29	0.24	0.34	0.28
word2vec average	0.30	0.60	0.40	0.30	0.60	0.40
word2vec max min	0.24	0.55	0.33	0.24	0.55	0.33
dependency Tuples	0.26	0.34	0.30	0.26	0.34	0.30
dependency grand tupels	0.25	0.37	0.30	0.25	0.36	0.30
dependency POS	0.31	0.42	0.35	0.31	0.41	0.35
dependency Grand POS	0.30	0.41	0.34	0.30	0.41	0.34
dependency Children	0.27	0.38	0.31	0.27	0.39	0.32
all Features	0.44	0.49	0.46	0.44	0.49	0.47
feature selected features	0.44	0.50	**0.47**	0.44	0.49	**0.47**

4.5 Comparing Data Sets

To verify that the methods used for the Swedish data set are working well, we test the two data sets with the same features and algorithm to see how the accuracy compares. Results from SVM and logistic regression were really similar, which is why the comparison of data sets is only presented for logistic regression in Table 4.5

Table 4.5: Comparison English and Swedish data set, logistic regression with C = 0.01

	Swedish	**English**	
	f-score	f-score	diff
character *n*-gram 1-2	0.412	0.591	30%
character *n*-gram 1-3	0.44	0.616	29%
character *n*-gram 1-4	**0.439**	0.631	30%
character *n*-gram 1-5	0.435	0.638	32%
character *n*-gram 1-6	0.423	0.637	34%
word *n*-gram 1-1	0.372	0.57	35%
word *n*-gram 1-2	0.354	0.579	39%
word *n*-gram 1-3	0.327	0.575	**43%**
manual	0.315	0.423	26%
C (1-5) W(1-2) Manual	0.431	**0.643**	33%

4.6 Baseline Performance

Before starting this project, a-word-list based solution was used to detect inappropriate comments. The solution is not based on machine learning, but simply classifies a comment as bad if it includes a word or phrase from a predefined list. We used this method on the Swedish data set to get the precision, recall and fscore of this solution. The results can be seen in Table 4.6

Table 4.6: Baseline method results

	Precision	Recall	f-score
Baseline solution	0.58	0.03	0.05

4.7 Tuning the Decision Limit

The logistic regression classifier gives a probability that a certain sample belongs to a certain class. By default, this limit is set at 0.5, but by changing the value for this limit, one can tweak the ratio between precision and recall. In Figure 4.1, one can see that precision gets better and recall worse when raising the limit. The opposite can be observed when lowering the limit, recall gets better but precision gets worse.

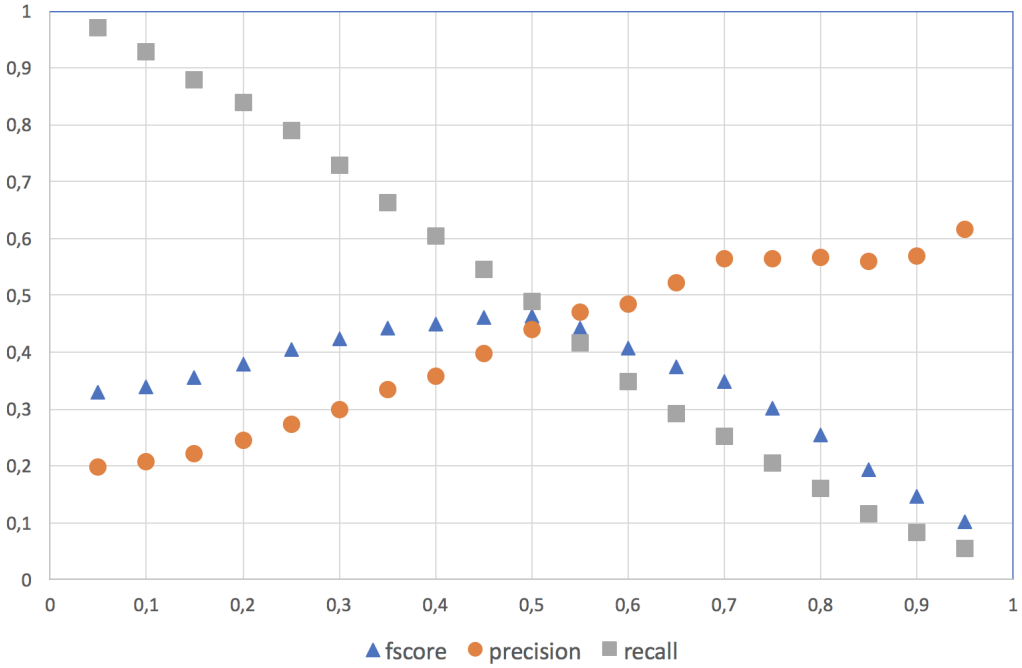

Figure 4.1: Precision, recall and f-score for different decision limits.

Chapter 5
Application

The algorithm developed in this thesis must be incorporated into some sort of application to be useful. Considering the accuracy of the classifier at this stage, automatic removal of comments is not desired since the ratio of OK comments being removed would simply be too high. That is why we have decided to incorporate the classifier into the moderation tool of Ifrågasätt rather than using it for automatic removal of comments or showing the score for the users. As seen in Figure 5.2, the annotators are given an option to show the toxic score for the comments sorted to show the comments with the highest score first. As seen in Figure 5.3, one also has the option to filter out all the comments with a score higher than e.g. 0.6. This way, the moderators can start by annotating the comments with high toxic scores and work their way down.

As seen in Figure 5.1, the classifier runs on a separate server which the main application server queries with every new comment in the system. The response from the server will be the toxic score for that particular comment, which is saved in the database and can be retrieved by moderators when needed.

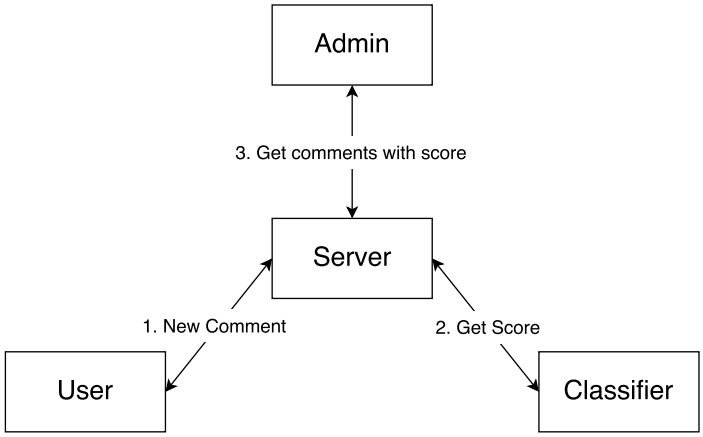

Figure 5.1: Architecture of the system with the classifier connected.

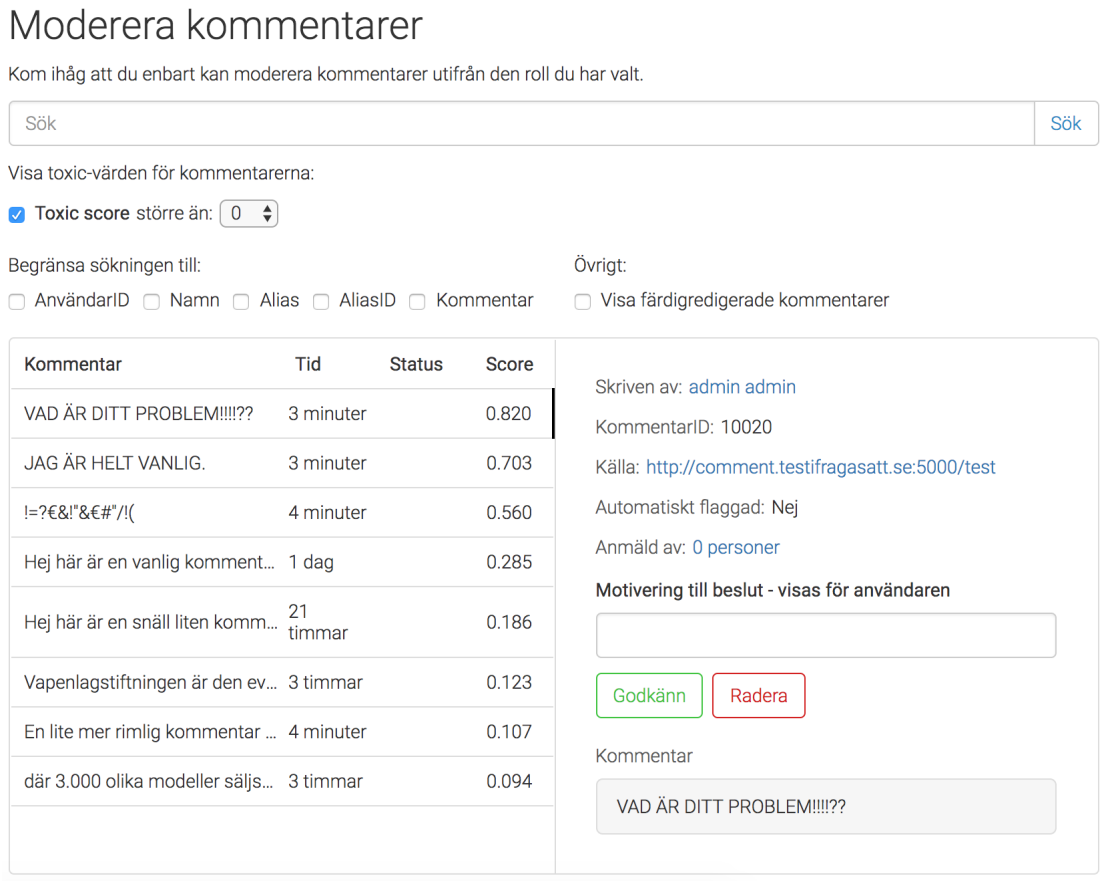

Figure 5.2: View for moderators, showing all un-annotated comments sorted on toxic score.

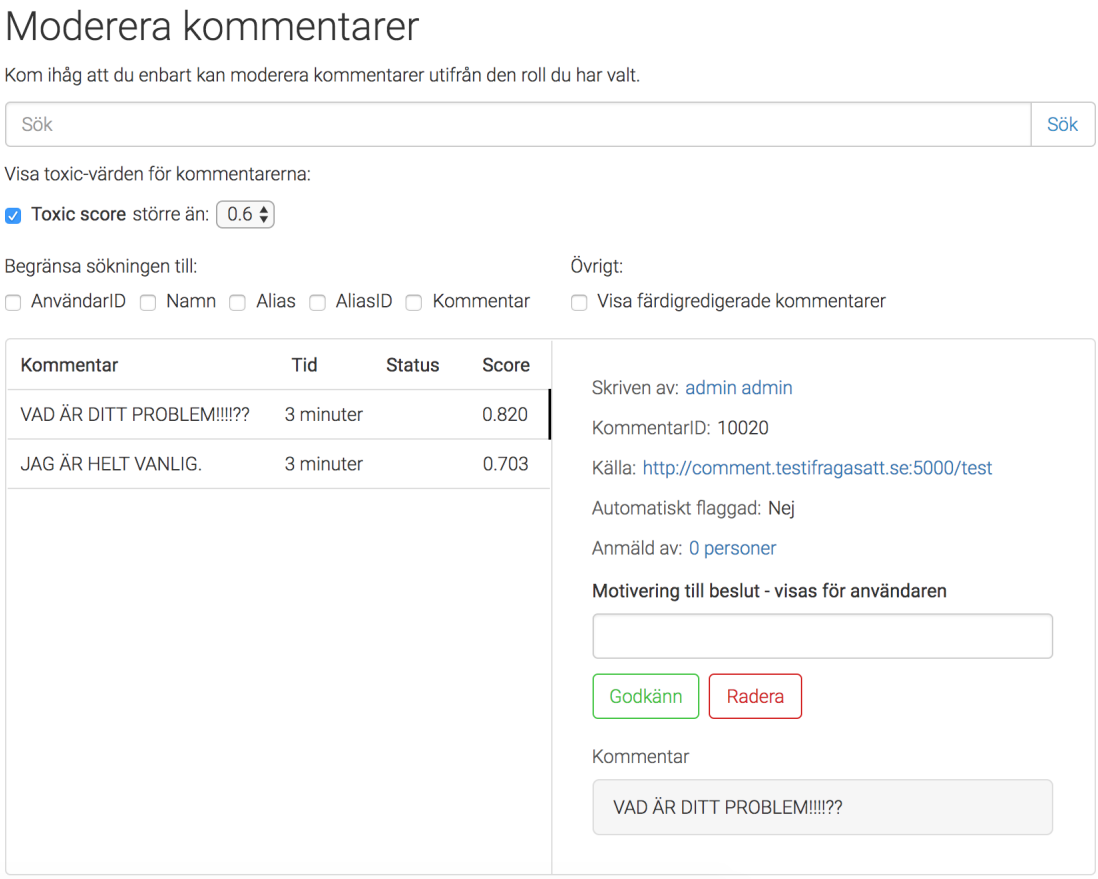

Figure 5.3: View for moderators, filtering out all comments with a toxic score lower than 0.6.

Chapter 6
Discussion

When starting this work, the goal was to find ways to streamline the moderation process by using machine learning techniques, concretely by exploring which algorithms, pre-processing and feature extracting gives the best accuracy for a classification problem. Logistic regression and SVM turned out to have really similar performance for this given data set, however logistic regression scored slightly higher and has the advantage of providing a probability score describing the certainty that a comment belongs to one of the two classes. Due to the relatively low f-score, the classifier could not be used for automatic removal of comments without removing too many *OK* comments, but by utilizing this score, a more practically useful application was implemented.

When comparing the method to previous work, the same type of features and algorithm is used, but one wonders, why is the performance not on par with Nobata et al. (2016).

6.1 The Annotation Process

When we looked at the results of the multi class inter-rater agreement, the result was simply not good enough to be useful. We therefore discarded the idea of labeling bad comments in three different categories, and went for the binary case instead (OK vs not OK). The company also realized that they do not need that level of detail, but rather have a binary classification which is also makes the moderation process faster.

Despite a very similar method compared to Nobata et al. (2016), the resulting f-score is not at the same level (0.82 vs 0.47). If presuming all the implementations measurements are done correctly, the reason for the low score is probably due to the data set. The Cohen Kappa measurements of the Inter-rater agreement showed rather large difference in how the two raters classified the comments. This means that the data set has a lot of ambiguity, where really similar comments are rated differently. Comparing the inter-rater agreement of the Yahoo Evaluation data set and our data set, 0.843 vs 0.553, it is quite clear that a high inter-rater agreement, helps to achieve a higher f-score. The size of the Yahoo

Evaluation data set was 2,000 comments, which is lower than our data set. This indicates that the size of the data set is not the main problem, but rather the level of ambiguity. These assumptions were confirmed by our test where we used our features (the ones which are not language specific) and algorithms to train and evaluate the performance of the Kaggle data set. When testing the same models at the Kaggle data set, we reach approximately a 30% higher f-score. Looking at the learning curve we could see that the increase in f-score is not due to the Kaggle data set being slightly bigger, but most likely because the data set has less errors.

So how to deal with the low inter-rater agreement? More precise instructions and use professional annotators instead of students. Even though the students were given rather precise instructions on what is OK and what is not, their opinions could differ. One could also extend the instructions with some example of edge cases where it is clear how comments should be rated.

6.2 Machine Learning

6.2.1 Features

As previously shown for English data sets, character *n*-grams work really well on their own for our data set as well. The feature selection process for logistic regression and SVM both had character *n*-grams and word2vec as the most important features.

The results of the feature selection gave a slightly better result for logistic regression but almost no significant increase for SVM. Despite only a small accuracy increase, one could argue that feature selection is important anyway. Fewer features with the same accuracy means a model which is simpler, easier to explain and easier reason about. On the other hand, a more complex model might capture more complex patterns and possibly be useful when the data set grows. It is therefore recommended to do feature selection when retraining the model with more data.

6.2.2 The Algorithms

Conducting experiments with neural networks were far more time consuming than experiments with SVM and logistic regression. If we had more time, we would probably have explored more parameters and types of neural networks.

6.3 Applications

The integration of the classification algorithm in the application is just one way how the classifier can be used. It would be possible to show the score to the user before it posts the comment as a way to remind the user that others may find their comment inappropriate. The user would always be allowed to post the comment despite a high score, but by making the score visible to the user, the number of inappropriate comments could possibly be decreased before even reaching the platform and moderators.

The solution implemented with the toxic score in the moderation tool is a way to ensure that it works as expected before releasing it in some form for the users. If the results are satisfactory, the next step would be to use the score earlier in the process, i.e. show it in real time for the user during typing, or notify the user when it publishes a comment with a high toxic score. Another solution could be to let users filter the comments based on the score, and hiding comments with a score higher than a certain threshold.

Chapter 7
Conclusion

We have completed a data mining process including gathering and annotation data, evaluating different features and classifiers, and finally implementing an application which utilizes the output from the classifier to streamline the moderation of comments. The feature engineering was done in an exploratory and iterative way, which most likely helped coming up with and implementing features. The annotated Swedish data set was the largest limiting factor for the accuracy of the classifier. Logistic regression with character *n*-grams, word2vec, spellchecker and sentiment 1 turned out to be the winning classifier. It scored a f-score of 0.47, precision of 0.44 and recall at of 0.49. This translates into 49% of the *not ok* comments are found, and that 44 % of the comments classified as *not ok* really are *not ok*. Comparing to the solution previously used, the f-score increase is huge, an increase from 0.05 to 0.47.

Overall, we are satisfied with the final performance of the classification algorithm, as well as the application utilizing the output of the algorithm. Comparing the solution developed by us with the one word list solution used by Ifrågasätt, we can proudly say we now can detect a much larger amount of *not ok* comments. The research questions defined in the beginning of the report are answered and the goal of streamlining the moderation process is fulfilled.

7.1 Future work

The most limiting factor is by far the data set, and the thus the moderation process must get better for the classifier to perform better. More research into the the annotation process could help sort out how can one achieve a better inter-rater agreement. One could look into what factors are important when designing guidelines and processes for moderation?

One could explore ways to do the moderation process differently by moderating each sentence individually. Detect which part(s) of the comment is the problematic part to make the moderation faster for moderators. This would require annotations sentence by sentence

which could be quite time consuming.

The current solution implemented for the application relies on manual retraining when new comments are added to the system. It would be interesting to implement a model which retrains every time a new comment enters the system to faster to adopt to changes in language and decrease the time spent on manual retraining.

Word embedding in the form of word2vec proved to be useful, but no more tests or investigations with word embedding for entire sentences or comments has been done. This could possibly help increase the accuracy. One could also explore more ways how one can use word embedding to construct features in more creative ways.

Bibliography

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Dadvar, M., de Jong, F., Ordelman, R., and Trieschnigg, D. (2012). Improved cyberbullying detection using gender information. In *Proceedings of the Twelfth Dutch-Belgian Information Retrieval Workshop (DIR 2012)*, pages 23–25, Ghent, Belgium. University of Ghent.

Dadvar, M., Trieschnigg, D., Ordelman, R., and de Jong, F. (2013). Improving cyberbullying detection with user context. In *European Conference on Information Retrieval*, pages 693–696. Springer.

Dinakar, K., Reichart, R., and Lieberman, H. (2011). Modeling the detection of textual cyberbullying. In *The Social Mobile Web, Papers from the 2011 ICWSM Workshop, Barcelona, Catalonia, Spain, July 21, 2011*.

Forman, G. and Scholz, M. (2010). Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement. *ACM SIGKDD Explorations Newsletter*, 12(1):49–57.

Jensen, K. (2012). IBM SPSS modeler CRISP-DM guide, ftp://public.dhe.ibm.com/ software/analytics/spss/documentation/modeler/18.0/en/modelercrispdm.pdf.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Kaggle (2012). Detecting insults in social commentary, https://www.kaggle.com/c/detecting-insults-in-social-commentary.

Klang, M. and Nugues, P. (2016). Langforia: Language pipelines for annotating large collections of documents. In *26th International Conference on Computational Linguistics (COLING), 2016*, pages 74–78.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Mehdad, Y. and Tetreault, J. (2016). Do characters abuse more than words? In *Proceedings of the SIGdial 2016 Conference: The 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y., and Chang, Y. (2016). Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 145–153, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Nusko, B., Tahmasebi, N., and Mogren, O. (2016). Building a sentiment lexicon for swedish. *Linköping Electronic Conference Proceedings*, 126(006):32—-37.

Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

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.

Yin, D., Xue, Z., Hong, L., Davison, B. D., Kontostathis, A., and Edwards, L. (2009). Detection of harassment on web 2.0. *Proceedings of the Content Analysis in the WEB*, 2:1–7.

Åklagarmyndigheten (2016). Förtal och förolämpning, https://www.aklagare.se/ombrottsligheten/olika-brottstyper/fortal-och-forolampning/.

Östmar, M. (2016). Filtering out negative and positive words, https://github.com/mattiasostmar/notebooks/blob/master/filter_out_negative-positive_words.ipynb.

EXAMENSARBETE Text classification of short messages
Detecting inappropriate comments in online user debates
STUDENT Anton Lundborg
HANDLEDARE Pierre Nugues (LTH)
EXAMINATOR Jacek Malec (LTH)

Bättre debatter online med hjälp av maskininlärning

POPULÄRVETENSKAPLIG SAMMANFATTNING **Anton Lundborg**

Fler och fler tidningar online inaktiverar sina kommentarsfält på grund av att diskussioner urartar. Detta arbete har utforskat maskininlärning som ett sätt att förenkla och effektivisera moderatorernas arbete.

I mitt examensarbete har jag tittat på hur maskininlärning kan användas för att identifiera vilka kommentarer som är olämpliga i ett kommentarsfält. Tidigare har man matchat kommentarer mot ordlistor med svordomar för att att fånga upp de värsta kommentarerna. När jag testade metoden med ordlistor visade det sig att enbart några få procent av de dåliga kommentarerna hittas medan majoriteten passerar obemärkta förbi. Med maskininlärning drar man nytta av kommentarer som modererats tidigare och använder vetskapen om huruvida en kommentar är lämplig eller olämplig för att träna algoritmen. Den tränade algoritmen används sedan för att bedöma lämpligheten i nya kommentarer.

Efter att ha experimenterat med olika typer av algoritmer och språktekniska sätt att extrahera olika egenskaper ur text, kom vi fram till en algoritm. Den metoden som fungerade bäst lyckades hitta ungefär hälften av alla olämpliga kommentarer, men fångar samtidigt upp ungefär lika många kommentarer som egentligen är lämpliga. För att sätta dessa siffror i ett sammanhang så skulle det innebära att man genom att gå igenom 20% av de totala antalet kommentarer kan hitta 50% av de olämpliga kommentarerna.

Algoritmen ger dessutom varje kommentar ett tal mellan 1 och 100, där 100 representerar en kommentar som är olämplig och 1 en lämplig kommentar. På så sätt kan man prioritera vilka kommentarer som modereras först och öka chanserna att de "värsta" kommentarerna tas bort först. I detta arbete tog jag fram ett grafisk gränssnitt där moderatorerna får möjligheten att både filtrera och sortera kommentarerna på talet som algoritmen räknade fram. Talet skulle även kunna användas på flera andra sätt, tex att författaren av kommentaren i realtid kan se hur lämplig kommentaren bedöms vara medan den skrivs.

Förhoppningsvis kan maskininlärningen göra att moderatorernas arbete blir lättare och mer effektivt, samt även på sikt bidra till att diskussioner i kommentarfält kan få fortgå utan att förstöras av hat-, hot- och trollkommentarer. Det finns ekonomiska incitament för tidningarna att ha kommentarsfält eftersom fler återkommande användare ger fler sidvisningar och reklamintäkter. Dessutom finns ett demokratiskt värde i att man låter läsare ifrågasätta och tycka till om innehållet i tidningarna.