Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

The Metadata Troll Detector

Semester Thesis

Stephan Dollberg

dstephan@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Tobias Langner, Jochen Seidel
Prof. Dr. Roger Wattenhofer

January 15, 2015

Abstract

Reddit.com offers a discussion platform for various topics. Given its huge size and its comment based discussion structure it attracts internet trolls. We write a bot that crawls Reddit for new comments and that tries to automatically detect trolling in them. In contrast to conventional solutions we do not only use text analysis of the comments but also take metadata into account. Metadata describes properties such as the degree of the participation in a discussion. For classification, we compile several characteristics of a comment based on data and metadata. These characteristics are used to train the machine learning algorithms of support vector machines to classify each comment. We show that it is generally very difficult to automatically detect trolls and see that metadata based approaches are still inferior to text based ones. However, the combination of both shows promising results.

Contents

Chapter 1

Introduction

With the ongoing expansion of the internet social platforms such as Youtube, Facebook or Twitter continuously gain new members. Once a platform has reached a certain user base not all new users want to participate in a constructive discussion. Instead, their only objective is to invoke other users' emotions and lead the discussion into a different and unrelated direction by directly or indirectly attacking certain groups. They are commonly referred to as trolls.

It is desirable to avoid such unnecessary disruptions and keep discussions clean. Manual intervention of moderators becomes infeasible after a certain user size. As such, several automated approaches have been the aim of research.

In a data analysis competition to find insulting posts in social networks [1] most participants used text analysis techniques [2]. One such technique is to count the words in a text to get a measure of the length of the text. Such counting techniques can be extended or specialized to not count all the words but only a certain subset, for instance, insulting curse words. Another technique is to use word or character n-grams. N-grams describe a sequence of words or characters. For example, the word 2-grams for the sentence "the fox jumps" are "the fox" and "fox jumps". Those n-grams are compared to other known n-grams or used to create statistical models. Others have tried to find trolls by using natural language processing to analyze the sense or sentiment of the text. It is classified by analyzing the sentiment of the used words and by assigning a certain emotion to them. Emotions such as rage or anger are hints for a troll comment [3]. Similar text based analysis has been used in areas that are related to troll detection such as fake online review detection [4, 5] or email phishing [6]. Most social networks or online platforms keep their automated anti-troll mechanisms private to avoid exploitation by trolls even if the rest of their software stack is open source.

1.1 Metadata Based Analysis

All the approaches listed in the previous section focus on the analysis of the actual content of a troll post. However, they ignore the metadata of such a post. Metadata is "data about data". It can be separated into descriptive and structural metadata. Descriptive metadata specifies certain properties of data. Structural metadata describes how the data is related to other data [7]. Taking a comment to a blog post as an example, the data of the comment is the actual text that the user posted. The metadata describes properties such as the author, the creation time and the length of the comment. In addition, the structural metadata explains whether the comment is a direct reply to the blog entry or to another comment.

In this project we also take metadata into account. The idea is that troll posts or the discussions resulting from them show descriptive and structural metadata which is different to non-troll posts. For instance, the comment text might be simple and short and spawn several quick replies which otherwise would appear slower and be more elaborate.

We test our approach on reddit.com (Reddit) which is a news- and link-aggregator. Users can submit links to other websites and subsequently discuss and comment the content of the linked page. They can also reply to other users' comments which results in a tree-like discussion structure. Users can up- or downvote comments and submissions to rate them. Higher rated comments or submissions are more likely to be shown to users than lower rated ones. Reddit is structured into sections called subreddits. They separate different topics such as politics or gaming. This structure fits perfectly for heavy discussions and allows us to extract metadata from the comments and the comment-trees. Figure 1.1 depicts a comment and its family members in a comment tree. For each comment on Reddit we have access to its text, the author, the creation time and to its parent amongst others.

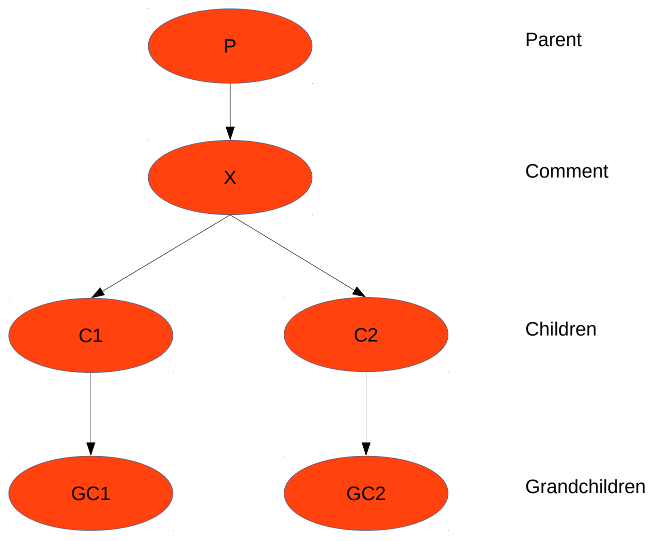

Figure 1.1: *Comment tree showing the family relation of a comment. The comment* ***X*** *has the parent* ***P****, the children* ***C1*** *and* ***C2*** *and the grandchildren* ***GC1*** *and* ***GC2****. Further children of* ***GC1*** *and* ***GC2*** *do also count as grandchildren of* ***X****.* ***C1*** *and* ***C2*** *are siblings.*

Chapter 2

Comment Classification

To classify a comment as trolling or not we compile certain characteristics based on the data or metadata that we have about that comment. Those characteristics are called *features*. Such a feature can for instance be the number of words in a post or the rate at which other people reply to it. The specific features are described in the next section. All features together form the feature vector of a comment. Each feature vector is labeled according to which class (troll or non-troll) it belongs to.

We use the machine learning technique of support vector machines (SVMs) to classify the comments. SVMs take a certain set of correctly labeled feature vectors as input and learn from them. Thereafter, SVMs can be used to classify unlabeled data sets based on their learned knowledge.

SVMs treat the training feature vectors as points in a high dimensional space. This space is separated by one or more hyper-planes that separate the classes best. Test data points are interpreted in the same space and labeled depending on which side of the hyper-plane they are positioned. Figure 2.1 depicts such a simple separation of two classes.

In Figure 2.1 we see that the separating hyper-plane is linear. However, SVMs do also support the use of kernel functions that produce a non-linear hyper-plane. SVMs can be tuned by modifying the SVM specific cost parameter C or by adjusting the parameters of the kernel. For the best performance we standardize our dataset to zero mean and unit variance before passing it to the SVM [8, 9].

2.1 Troll Feature Types

We differentiate between two kinds of features. The first kind describes properties of the actual comment. These features can be dynamic or static. A dynamic feature might change while the actual post doesn't change. An example for this is the structure of a comment tree that changes when somebody replies to a comment. Static features, such as the word count, stay the same once the

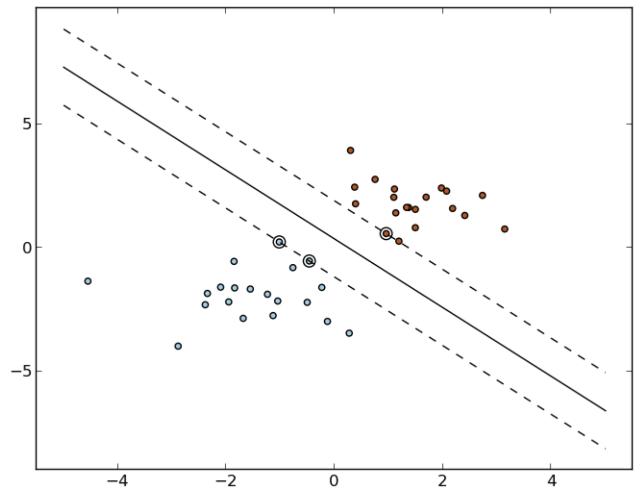

Figure 2.1: *SVM hyper-plane separation of two classes. The hyperplane perfectly separates the two class types (blue and red). Test data points are labeled depending on which side of the hyperplane they are located.* [1]

comment has been created.

The second kind refers to properties coming from the user that posted the comment. This kind of feature is independent from the actual comment and could already be generated before the user posts his next post. It describes the history of the user's previous posts.

2.2 Troll Features

Static Comment Features

Word Count As the name suggests, the Word Count defines the words per comment. The comment text is split by whitespace and the length of the resulting word list is the word count.

Bad Word Count Similar to the Word Count, the word list is compared to a predefined list of "bad" words and matches are counted.

Compression Ratio For this feature the comment body is compressed. The Compression Ratio specifies the ratio of the absolute length of the compressed

[1]Figure taken from Scikit-learn [9].

comment to the length before compression.

We use the zlib library for compression [10]. It is possible to pass a predefined dictionary to the compressor that is expected to compress well. An alternative to the standard approach without passing a dictionary is to first create a dictionary by counting the most frequent words in a number of comments that are known to be trolling. The expectation is that troll comments compress better than non-troll comments.

All Capital Word Count The All Capital Word Count is a counter of how many words the comment contains that consist of capital letters only.

All Capital Word Count Ratio The All Capital Word Count Ratio is calculated by dividing the All Capital Word Count by the absolute Word Count.

Dynamic Comment Features

Total Number Of Replies The Total Number Of Replies is the sum of all children and grandchildren.

Average Reply Word Count The Average Reply Word Count defines the average of the Word Counts of all children and grandchildren of a comment.

Regular User To One Time User Ratio We look at all the authors of the children and grandchildren of a comment. The users are categorized into either being a one time user of the current subreddit or a regular user. A regular user is characterized by having more than a certain threshold of posts in the given subreddit. An alternative is to not look at the count of posts in the subreddit but instead at the overall number of posts.

Children Count The Children Count is the absolute number of direct replies to a comment. This can be seen as the breadth of the first level of a comment tree. For the comment in figure 1.1 the Children Count is two.

Length Of The Deepest Comment Tree Branch In comparison to the Children Count which describes the breadth of a comment tree this feature defines the depth. It is defined as the length of the longest branch of a comment tree. Looking at figure 1.1 both branches are equally long with a length of two.

Children Count Growth Rate The Child Count Growth Rate describes the rate at which the Children Count is initially growing. We define it as the number of replies that were created in a certain time span after the comment was posted. If the timespan reaches into the future we assume that no further replies would be generated. This is a pessimistic assumption as we prefer to rather classify a troll comment as not being trolling than the other way around.

Deepest Branch Growth Rate The Deepest Branch Growth Rate is similar to the Children Count Growth Rate with the difference that it measures how fast the length of the deepest branch increases.

User Based Features

User Up/Down Ratio The upvote to downvote ratio is calculated by taking the average of the scores of all posts from a user. Comments with an absolute score smaller than two are ignored as we assume them to be not judged by the community.

Chapter 3

Reddit Bot Design

Reddit offers a HTTP API that eases the creation of an automated bot. The bot is designed to consist of two parts. On the one hand side there is the crawler and on the other side is the troll detector that analyzes and classifies new comments using the SVM technique. The full design is depicted in figure 3.1.

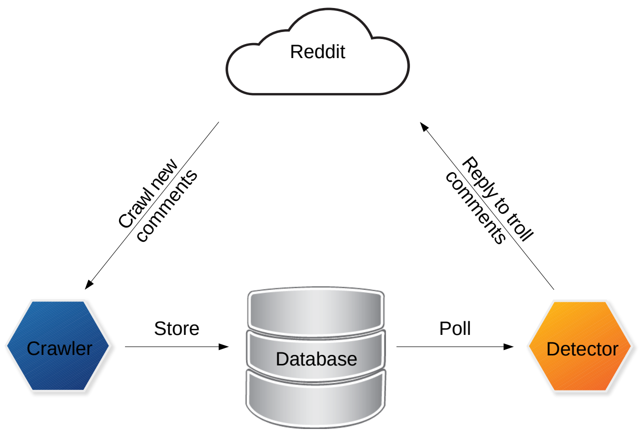

Figure 3.1: *Design of the Reddit Bot. The crawler stores new comments from Reddit in a database. The troll detector reads these new comments from the database and analyzes them. If a troll comment is found, then a reply to Reddit is posted.*

The crawler polls the API every two seconds (adhering to the Reddit rate limiting standards) to get the latest comments in a list of selected subreddits. New comments are stored in a local database. If a new comment is from a submission which we have not seen yet or has a parent that we did not store yet, we retrieve the submission and recursively fetch all the unseen parent comments. This way we can locally compute the comment tree and compile our features.

The troll detector on the other hand is constantly polling the database for new comments from the crawler. Whenever the detector sees a new comment it rechecks all the comment's parents. This is necessary because the dynamic features of the parents change. The SVM initially learns from a manually classified dataset. However, in contrast to offline testing the learning process is slightly different. Instead of learning from the feature vectors of the marked comments only, we add additional feature vectors for each point in time at which a reply is added to a comment. These additional data points have the same static features but different dynamic features. Using this learning scheme we do train the SVM to detect trolls as soon as possible. In comparison, the offline learning method only learns from the state in which all trees have fully evolved. If a comment is positively identified as trolling, it is marked in our database so that it will no longer be checked. In addition, we post a reply on reddit indicating that we have identified the comment as trolling.

Splitting the bot into two separate entities gives us several advantages. At first, we can easily let both parts run independently. This enables us to crawl Reddit while not necessarily having to analyze the comments. Second, by running on two different IP addresses we can avoid being rate limited from a single IP address. Finally, the design allows us to distribute the two components to multiple machines. This allows for a setup with a cheap virtual machine for the crawler and stronger machines for the database server and the troll detector. In addition, it would allow splitting up the two roles into multiple crawler and detector instances with each being responsible for a certain set of subreddits.

Chapter 4

Results

To test our approach we first create a ground truth by marking around 400 reddit comments from several subreddits as being trolling or not. We judge a comment as trolling if it is overly provocative or hateful against a certain group.

Our marked dataset consists of an equal amount of trolls and non-trolls to avoid any bias or variance caused by an uneven split. Concerning the SVM configuration parameters, we use the default $C = 1$ setting together with the default kernel (radial basis function [8]) of our implementation. Tuning of these parameters does not show any improvements.

To evaluate the prediction performance of our SVM we perform an N-fold cross-validation test. In an N-fold cross-validation test the data set is split up into N equal sized sets. After the split, the SVM is trained on $N - 1$ sets and tested on the remaining test set. This process is repeated N times until each set has been the test set once.

Given the cross-validation results we measure the classification quality by looking at the metrics accuracy, precision, recall and f_1-score. They are defined as follows. Troll comments are interpreted as a positive classification while non-troll comments are negatives ones.

$$accuracy = \frac{\text{true positives} + \text{true negatives}}{\text{size of the dataset}}$$

$$precision = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$recall = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

$$f_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

The accuracy describes the percentage of how many comments we classify correctly. Precision and recall(sometimes called sensitivity) put focus on the behavior of the classifier concerning positives. Precision can be interpreted as how much more correct than incorrect positive classifications are returned. This is especially important to us because we prefer false negatives over false positives. The reason for this is that in our scenario false negatives do not have any consequences compared to false positives. In other areas such as cancer classification both error types are equally unacceptable. On the other hand, the recall describes how much percent of the positives are classified correctly. Finally, the f_1-score is the harmonic mean of the recall and precision. Its advantage over the two metrics is that it is not possible to misleadingly modify the score. For instance, by simply classifying everything as negative it is possible to get a very high precision. However, the f_1-score would be very low because it also takes the recall into account which is very small in this case [11].

4.1 Offline Results

Doing a five-fold cross-validation on our marked dataset we obtain classification results as depicted in Figure 4.1.

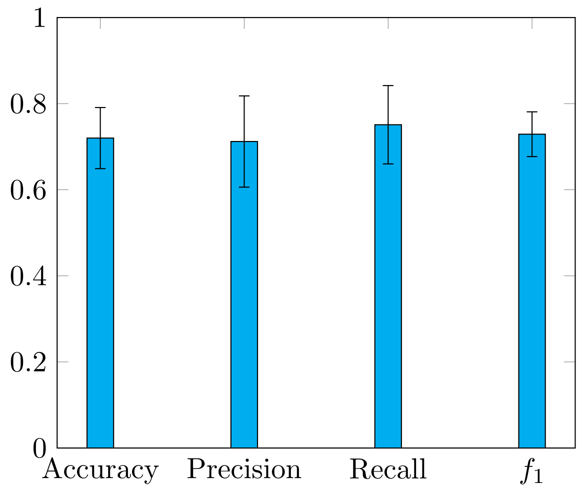

Figure 4.1: *Overall Performance Estimation of our classifier using all features (with 95% confidence interval).*

We see that all metrics are around the seventy percent mark. While this is definitely better than random classification it leaves room for improvements and is around ten percent worse than what was reported for the technique that analyzes the emotions of a text [3]. As precision and recall are in the same range we see that we do not have any bias in either of the classification directions. Using only comments from a single subreddit shows no improvements in performance.

4.2 Feature Comparison

To evaluate which of our features performs best we again run our cross-validation test. However, the SVM only uses a single feature and not all of them. For comparison we look at accuracy and f_1-score.

Feature	Accuracy	f_1-score
User Up/Down Ratio	0.69	0.67
Compression Ratio	0.68	0.71
Word Count	0.67	0.73
Average Reply Word Count	0.64	0.69
All Capital Word Count Ratio	0.60	0.66
All Capital Word Count	0.58	0.61
Children Count Growth Rate	0.55	0.48
Total Number Of Replies	0.53	0.68
Bad Word Count	0.52	0.25
Children Count	0.50	0.51
Deepest Branch Growth Rate	0.48	0.53
Regular User To One Time User Ratio	0.46	0.45
Length Of The Deepest Comment Tree Branch	0.45	0.43

Table 4.1: *Classification accuracy and f_1-score with only one of the features.*

Looking at the results from Table 4.1 we see that the strongest features are text analysis related features such as the Word Count or the Compression Ratio. Pure metadata features such as the Children Count or the Children Count Growth Rate are only minimally better than random classification while some are even worse(Deepest Branch). However, features that combine text analysis with structural metadata such as the Average Reply Word Count show that it is worth to not only look at the comment itself but also at related data.

It would be interesting to see how more advanced text based techniques perform in this combination. In addition, another possible further research topic is to see how features perform that put a comment in comparison to its siblings.

Finally, for the growth rates a time span of one or ten hours proves to be the best duration. Using a specialized dictionary for the Compression Ratio does not show any improvements.

4.3 Online Results

We do also perform a test run of our troll detector bot on Reddit. The reply we post back to Reddit once we find a troll asks the community to judge our classification by up- or downvoting the reply.

The feedback is mixed. While we agree with most of the upvotes on the correct classifications and downvotes on the incorrect ones there are some debatable cases. Some of the classifications are heavily downvoted from the community even if we would say by human judgment that the posts are indeed trolling.

From this we learn two important points. First, we see that a good ground truth is very important. Second, some online communities, such as Reddit, live from their trolls as they define the culture of the platform. Therefore, if troll detection is desired on such a platform, the ground truth should be setup by the community itself.

Finally, the online test makes it clear that features that rely on structural metadata have a disadvantage for use in trolling prevention. Structural properties first have to develop themselves with a growing comment tree. As such, they are not useful to prevent an active troll attack but can be used to retrospectively detect users that have successfully trolled in the past.

Bibliography

[1] Kaggle.com: Detecting insults in social commentary. http://www.kaggle.com/c/detecting-insults-in-social-commentary (2012) Accessed: 15.01.2015.

[2] Kaggle.com: Detecting insults in social commentary - discussion. http://blog.kaggle.com/2012/09/26/impermium-andreas-blog/ (2012) Accessed: 15.01.2015.

[3] Cambria, E., Chandra, P., Sharma, A., Hussain, A.: Do not feel the trolls. (2010)

[4] Jindal, N., Liu, B.: Opinion spam and analysis. In: Proceedings of the 2008 International Conference on Web Search and Data Mining. WSDM '08, New York, NY, USA, ACM (2008) 219–230

[5] Dave, K., Lawrence, S., Pennock, D.M.: Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In: Proceedings of WWW. (2003) 519–528

[6] Fette, I., Sadeh, N., Tomasic, A.: Learning to detect phishing emails. In: Proceedings of the 16th International Conference on World Wide Web. WWW '07, New York, NY, USA, ACM (2007) 649–656

[7] Press, N.: Understanding Metadata. National Information Standards Organization Press (2004)

[8] Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning: data mining, inference and prediction. 2 edn. Springer (2009)

[9] 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., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12** (2011) 2825–2830

[10] Deutsch, P., Gailly, J.L.: Zlib compressed data format specification version 3.3. RFC 1950 (May 1996)

[11] Powers, D.M.W.: Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. Technical Report SIE-07-001, School of Informatics and Engineering, Flinders University, Adelaide, Australia (2007)