Algorithmically Supported Moderation in Children’s Online Communities

by

Flora Tan

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Masters of Engineering in Electrical Engineering and Computer Science
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2017

© Massachusetts Institute of Technology 2017. All rights reserved.
Algorithmically Supported Moderation in Children’s Online Communities

by

Flora Tan

Submitted to the Department of Electrical Engineering and Computer Science on February 3, 2017, in partial fulfillment of the requirements for the degree of Masters of Engineering in Electrical Engineering and Computer Science

Abstract

The moderation of harassment and cyberbullying on online platforms has become a heavily publicized issue in the past few years. Popular websites such as Twitter, Facebook, and YouTube employ human moderators to moderate user-generated content. In this thesis, we propose an automated approach to the moderation of online conversational text authored by children on the Scratch website, a drag-and-drop programming interface and online community. We develop a corpus of children’s comments annotated for inappropriate material, the first of its kind. To produce the corpus of data, we introduce a comment moderation website that allows for the review and label of comments. The web-tool acts as a data-pipeline, designed to keep the machine learning models up to date with new forms of inappropriate content and to reduce the need for maintaining a blacklist of profane words. Finally, we apply natural language processing and machine learning techniques towards detecting inappropriate content from the Scratch website, achieving an F1-score of 73%.

Thesis Supervisor: Andrew Sliwinski
Title: Research Scientist

Thesis Supervisor: Mitch Resnick
Title: LEGO Papert Professor of Learning Research
Acknowledgments

I would like to thank my advisor Andrew Sliwinski for giving me the chance to work on Scratch for my Masters of Engineering thesis. Working on this project has been an incredibly rewarding experience. I am very grateful for his patience, guidance, and intellectual support.

I would also like to thank the entire Scratch team for being a constant source of motivation, support, and most importantly, fun. I extend the greatest thanks to Mitch Resnick for gifting me this incredible opportunity to work alongside the Scratch team. I thank Natalie Rusk for first allowing me to work alongside the Scratch team as a UROP student during my first year at MIT. I could not have asked for a better way to conclude my studies at MIT than where I first started — with the Scratch team.

Furthermore, I would like to thank Colby Gutierrez-Kraybill for helping me onboard to Scratch. I would also like to thank the Scratch Moderation team, especially Eric Schilling, Christian Balch, and Sarah Otts, for all of their help in annotating the comments data and providing feedback throughout the development of “Scratch TFC.” Their sense of contagious fun throughout my time at the lab has made it an indelible experience, and I’m very happy to have had the chance to work alongside them.

Lastly, I would like to thank my friends and family for their endless support. I could not have made it this far without them and for that, I am immensely grateful.
# Contents

1 Introduction ......................................................... 15
   1.1 Experimental Group ........................................... 16

2 Background .......................................................... 21
   2.1 Online Moderation .............................................. 21
   2.2 Part-Of-Speech (POS) Tagging ................................. 23
   2.3 Classification Techniques .................................... 24
      2.3.1 Self Harm and Bullying ................................. 24
      2.3.2 Spam ..................................................... 26

3 Data Collection ...................................................... 29
   3.1 Data Collection Overview .................................... 29
   3.2 Scratch Comments Moderation Website ........................ 31
      3.2.1 Technology Stack ....................................... 31
      3.2.2 Database ................................................. 32
      3.2.3 Application Pipeline ................................... 34
      3.2.4 Deployment ............................................... 35
   3.3 Initial Dataset .................................................. 35
   3.4 Preparing Data .................................................. 35
   3.5 Open Source ..................................................... 36

4 Algorithmic Models .................................................. 39
   4.1 Data Preprocessing ............................................. 39
List of Figures

1-1 Moderation Process ............................................. 17
1-2 Number of moderation tickets over time ..................... 19
3-1 kys ................................................................. 31
3-2 Scratch Comments Moderation Website ...................... 32
4-1 Lancaster stemming ............................................. 41
4-2 WordNet stemming ............................................. 42
4-3 Examples of applying Twitter POS Tagging on Scratch comments . 43
List of Tables

3.1 Comments Table ................................................. 32
3.2 Results Table .................................................... 33

4.1 NLTK Sentiment Analysis Applied to Scratch Comments ........... 44

5.1 Maximum Entropy Results ........................................ 50
5.2 SVM Results ..................................................... 50
5.3 Naive Bayes Results .............................................. 50
5.4 Random Forest Results .......................................... 51
5.5 Model Comparison by F1-score ................................. 52
Chapter 1

Introduction

The advent of social media has triggered a surge in online user-generated content. Many social media websites, including Facebook and YouTube, strive to provide the best possible environment for their users by publishing Community Guidelines. To ensure visitors feel safe, these websites flag and erase inappropriate posts.

A recent article by The Verge magazine [2] revealed the surprisingly manual moderation policies that these websites employ. An October 2014 Wired story [1] documents how front line moderators in the Philippines operate in modern-day sweatshops. This massive labor force handles “content moderation” - the removal of offensive material - for U.S. social networking websites, including tech monoliths such as Facebook and Google. The moderators actively screen all posts generated by the website, examining each post individually and removing any offensive material, including pornography, abuse, and profanity. These websites also allow users to contribute to moderation by flagging posts that they find inappropriate, bringing the posts to the attention of moderators.

It is crucial that these websites keep internet bullies, criminals, racists, and spammers at bay. In recent years, Twitter has been a platform where highly publicized accounts of sexual harassment have taken place, many of which involved female celebrities [3]. Indeed, Twitter has been heavily criticized for its inability and slowness in addressing the proliferation of harassment and abuse on its website [3]. A report by The Verge in 2015 [3] documents that the then CEO, Dick Costolo, released a frank
acknowledgement about the company’s failure in addressing abuse and how it has been causing the website to “lose core user after core user by not addressing simple trolling issues that they face every day.”

Perhaps one of the most important user groups of online communities that must be considered are children. The advent of social network sites, online gaming platforms, and online communication, as well as the ease of accessibility to these online platforms, has largely impacted the social lives of children in the United States [4]. Research has shown that youth tend to be earlier adopters than adults of digital communications and authoring capabilities [4]. The high usage of these social networking sites, however, potentially exposes children to obscene or harmful content, and leaves them vulnerable to cyberbullying and internet predators.

Considering the ease which false identities can be created online and the particular vulnerabilities that children present, there is a clear necessity for moderation tools that will respond quickly to abusive behavior towards children. Furthermore, the exponentially growing amount of user-generated content on these social media platforms presents a requirement for serious improvements in current moderation processes to make them more efficient and not as labor intensive.

1.1 Experimental Group

Children have become central participants in defining online literary practices, including nuanced social norms for social network activities and casual forms of online speech [4]. As a result, their unique form of language has caused moderation of children’s comments to be a unique problem.

In this thesis, we will examine content moderation for Scratch, a web-based, drag-and-drop programming language and online community developed by the Lifelong Kindergarten Group at the MIT Media Lab. It is designed for children between the ages of 8 to 14, but is used by people of all ages. Scratch has a massive user base, with over 16 million registered users and over 159 million page views per month [6]. Scratch users can program their own interactive stories, games, and animations.
using the programming editor. The website also allows users to publicly share and collaborate on projects with others in the online community.

To maintain the integrity of the community and ensure that the user base is abiding by the Community Guidelines, Scratch employs moderation strategies to flag down inappropriate content throughout the website. Currently, Scratch employs Cleanspeak [5], a platform for filtering profanity and moderating user-generated content. It is equipped with automatic filters for profanity, and for further filtering, allows moderators to input new words or phrases. The tool will report any content it determines as inappropriate. Moderators and administrators of the Scratch website review the content flagged by the Cleanspeak system as well as any content flagged by users of the website.

Figure 1-1: A diagram of the moderation process currently in place on the Scratch website.

While Cleanspeak does a good job at filtering specified words or phrases, it often misses content because of common spelling errors and/or interjections in between phrases (example 1), because it is unable to identify new spam that it has not seen
before (example 2), because it is unable to distinguish bullying behavior (example 3), or because users find unique ways of bypassing the filter through trial and error (example 4). The brute force solution that the moderators employ - that is, using a blacklist of bad words or phrases - mandates an automated way of keeping the moderation tool up-to-date with new forms of inappropriate content that appear on the Scratch website.

1. Example 1. An example of a user-generated comment with spelling errors.

   foooouuuwuck u beecitch

2. Example 2. An example of spam left as a user comment.

   Hello, members, [url=http://www.mbtshoesdiscount.org/]MBT shoes[/url] gradually become popular. The key to the [url=http://www.mbtsale.org/]MBT[/url] is patented sole construction, at the heart of which is the soft Masai Sensor. MBT takes Masai barefoot technology. It is said that [url=http://www.mbtsale.org/]MBT shoes[/url] are the world smallest gym.

3. Example 3. An example of cyberbullying.

   Please shut up about your newest project. It will explain why you are such an Idiot!

4. Example 4. An example of abuse that uses asterisks instead of swear words.

   ... I was begging for mercy. because I almost F***ING KILLED MYSELF!!!!!! You started being mean! So did she! That's why!!!!! JUST LEAVE ME ALONE! I HATE YOU!

Moreover, the Cleanspeak system has a high false positive rate of 80%, where a false positive is a discrepancy between what the filter (or Scratch user) reports to the moderator, versus what the moderator ultimately decides. This means that 80% of the comments reported to moderators result in no action taken. With the scale at which Scratch operates drastically increasing, there has been a corresponding exponential increase in the amount of user-generated content on the website. Currently, over three million comments are created per month; in a year’s time, this number is on track
to rise to four million comments per month. Consequently, the amount of content that the Scratch Moderation team has to review is greater than ever before. In total, there are currently over 101 million comments on Scratch [6]. The high false positive rate of the Cleanspeak filter and the constraint for Scratch to scale accordingly calls for a more accurate detection system.

The objective of this thesis is to explore Natural Language Processing (NLP) techniques to moderate online comments on the Scratch website. The overarching goal is to build an NLP tool based on a unique corpus set — comments generated by the young users of the Scratch website. We hypothesize that natural language processing will allow us to determine context of speech and more accurately flag down inappropriate material. This project will examine part-of-speech (POS) tagging for informal, arbitrary texts; clustering and classification systems that will identify inappropriate text and spam; then finally, some groundwork for an in-house data pipeline.
Chapter 2

Background

The rise and prominence of social media platforms such as YouTube, Facebook, and Twitter has brought alongside it a clear need for moderation. Surprisingly manual forms of moderation policies - and until recently, lack thereof — persist to this day. In this thesis, the content of interest is a unique study, as it considers the moderation of children’s language and associated culture of the Scratch community.

There have been recent developments in the employment of statistical machine learning techniques towards automated moderation systems. This project aims to develop a new profanity and abuse detection algorithm that will incorporate natural language processing techniques with supervised machine learning strategies. More specifically, we are interested in examining work involving part-of-speech (POS) tagging, textual feature extraction, as well as classification techniques.

2.1 Online Moderation

Most existing websites rely on manual moderation of inappropriate content. Classically, this administrative role is given to a group of moderators who have special privileges that allow them to delete and remove messages [7]. Moreover, most systems use a post-moderation strategy such that submissions to the website are reviewed after being published.

Like staff at Twitter, Facebook, and YouTube, the moderators of Scratch are
responsible for enforcing the website’s rules, and consequently influence everyday interactions amongst users by determining what content is allowed on the website. The Community Standards and Guidelines for Twitter [8], Facebook [9], and YouTube [10] show that they have rules in place against hate speech and harassing people based on race, gender, or religion, just as Scratch does [11].

These social media sites also employ distributed moderation, in which users are given privileges to enable the community to self-moderate in a reactive fashion, such that they can alert moderators when they find inappropriate content [12]. By allowing users to report, these websites empower their moderation groups with extra resources, given the sheer amount of content created on a daily basis. For example, Facebook has reported receiving 1 million user reports on a daily basis [7].

While there have been many attempts to develop better systems that detect abusive content, these social media websites ultimately defer to humans to review complaints. Indeed, tech giants including Google and Facebook continue to outsource moderation to contracted workers in the Philippines and other countries [1]. Existing detection techniques, which are divided into rule-based and statistical techniques, suffer limitations: rule-based techniques rely on manually maintained blacklists of bad words, and statistical techniques are often limited by the lack of labeled examples and the cost of labeling a large-enough number of examples [12]. Most attempts to detect abusive language in these media outlets using machine learning are imperfect, with accuracies in the 70% range, and the highest performing model, developed by Yahoo Researchers in April 2016 and trained on comments on Yahoo News and Yahoo Finance, accomplishing an F1-score of 78.3% [13].

Rather than supplanting the work of the moderators, this project intends to aid their efforts in reviewing reported material by developing a new online harassment detection system that has a lower false positive and false negative rate, as well as a new pipeline that allows the system to be responsive to new content.

In May 2016, Twitter introduced a new tool to combat spam and abuse in broadcasts to Periscope [14], a live-streaming app that it acquired. They released a comment moderation system that empowers the user community to report and vote on
comments that they consider to be spam or abuse. The decision as to whether the
comments feature inappropriate content is deferred to a majority vote, and the author
of a comment that is determined to be abusive will be temporarily disabled.

In this thesis, we propose a similar system of reviewing comments. While the
Periscope tool is intended to engage the community in helping to improve comment
moderation through consensus [14], we introduce a system of reviewing comments
designed for the Scratch Moderation team. Our goal is that the labeling of com-
ments through this pipeline will help the moderation tool react to any new forms of
inappropriate content that appear on the Scratch website.

2.2 Part-Of-Speech (POS) Tagging

POS tagging assigns parts of speech to words (and other tokens), such as noun, verb,
adjective, etc. [15]. POS tagging has often been used for sentiment analysis tasks
[13]. For this reason, we consider POS tagging as it might help the models learn
contextual clues in sentences.

One of the most relevant research on POS tagging that has been done involves
conversational text on Twitter. Researchers at Carnegie Mellon University released
a paper on POS Tagging for Twitter, describing how they evaluated 1,827 annotated
tweets and achieved an accuracy of 90% [16]. Their tagset comprised of various tokens
seen mainly in social media, including URLs and e-mail addresses, emoticons, Twitter
hashtags, of the form #tagname, and Twitter at-mentions of the form @user. We
can use this as a reference since many conversations on the Scratch website similarly
use idioms such as emoticons and hashtags.

In 2013, the researchers published another paper that iterated upon their previous
work, increasing the performance of the POS tagging to 93% [17]. They aim to
directly analyze the syntax of online conversational text on its own terms, instead of
attempting to normalize the text into normal written English. The authors claim that
many of Twitter’s unique linguistic phenomena are due not only to its informal nature,
but also a set of authors that heavily skews toward younger ages and minorities. As
a result, they leave the unique word forms in Tweets untokenized. Likewise, Scratch is unique in that the authors of the comments are generally children or young adults so it is crucial to determine the tagset that will be used for this project.

In terms of the corpus of interest, there has been considerable work done by Sagae et. al. in 2007 [18] on annotating child language transcripts for grammatical relations. They worked on annotating the English section of the CHILDES database with grammatical relations, and used the corpus to develop a data-driven parser, which are both freely available online. Initial investigation of the CHILDES database and the parser is helpful in understanding the scope of the NLP tasks.

2.3 Classification Techniques

For this project, we will be interested in detecting three forms of inappropriate content: spam, bullying, and self harm. There has been ongoing, separate research for these three types of content.

2.3.1 Self Harm and Bullying

Previous work on the detection of cyberbullying has been done at the MIT Media Lab. In 2011, Dinar, Reichart, and Lieberman [19] worked on Ruminati, a project that focused on detecting sensitive topics from YouTube comments. Using the YouTube API, they scraped comments from controversial videos surrounding sexuality, race & culture and intelligence. The comments were grouped into clusters, and then hand annotated. They curated 1500 instances for each of the three categories to form the training data.

Their feature space design was two-fold: general features that were common for all three labels, and specific features that are used to predict a particular label. The general features consisted of TF-IDF (term frequency inverse document frequency) weighted unigrams, the Ortony lexicon of words denoting negative connotation, and a list of profane words and frequently occurring POS bigram tags across the 3 datasets. The label specific features consisted of unigrams and bigrams, including frequently
used forms of verbal abuse or stereotypical utterances, specific to the label. They used this towards three different supervised machine learning models: JRip, J48 (Decision Trees), and Support Vector Machine (SVM), as well as a Naive Bayes classifier.

Two experiments were done using these models. In the first, a binary classification was done on each of the three datasets for each of the labels. In the second, a multiclass classifier was trained upon an aggregate dataset consisting of all three datasets. The results showed that the multi-class classifiers underperformed compared to binary classifiers. The JRip classifier performed the best in terms of accuracy, scoring 80.20% for sexuality, 68.30% for race, and 70.39% for intelligence, while the SVMs significantly higher kappa values suggested a higher reliability across all labels.

In a later work, Dinar, along Picard and Lieberman, produced a paper on Common Sense Reasoning for Detection, Prevention, and Mitigation of Cyberbullying. The authors [16] present an approach for detecting cyberbullying. They demonstrate the limitations of using purely supervised machine learning classification. Using a dataset of YouTube comments and flagged content on FormSpring, they were able to use classical supervised machine learning approaches, including Support Vector Machines (SVM), J48 (Decision Trees), and Naive Bayes, to flag down user generated content that contained recurring patterns of slang and profanity. However, these models were unable to detect instances that did not contain explicitly negative words and instances that required semantic reasoning to interpret.

To remedy this, they constructed an innovative common sense knowledge base called BullySpace that encodes knowledge about bullying situations. The knowledge base incorporated stereotypical knowledge regarding sexuality and gender roles, for the purpose of detecting subtle, indirect forms of verbal abuse. The specific behavior they attempt to detect is one in which a user may attribute characteristics of the opposite sex to an individual. Using this dataset, the authors were able to perform analysis and evaluate the model on real-world instances reported by users on the Formspring social networking website. They tested real instances of Formspring comments and had three annotators either agree or disagree with the label that the model produced.
2.3.2 Spam

Supervised machine learning techniques have been shown to be effective in detecting email spam [21]. Previous work [21] has shown that features of spam can be extracted from comments. Examples of such features include: the length of a comment, since in the real world, normal comments are usually short and to the point; similarity to the content the comment was written about; and ratio to popular words and propaganda.

In 2013, research was done by Martinez-Romo and Araujo [22] on detecting spam tweets on Twitter. One of the features that the researchers extracted was the divergence between the ten tweets posted just before and after the spam tweet, since in most cases the topic or the terms used in the spam tweets have no relationship to the adjacent posts. Additionally, they used content-specific attributes as features. They use metrics such as the number of URLs per number of words, number of hash-tags per words, number of words from a list of spam words, number of tweets posted in the thread by the same user, and time since the last tweet posted by the same user. These types of features may be directly applicable to the detection of spam on the Scratch website. Using an SVM, the researchers were able to achieve an accuracy of 92.2%, which suggests that such comment based features may be similarly promising for our model.

In 2014, another paper was released by Rdulescu, Dinsoreanu, and Potolea [23] that examined spam detection for comments on YouTube and the Daily Telegraph. They extracted features including the number of links in the comments, the number of white spaces in the comments, number of sentences in the comment, number of punctuation marks in the comment, word duplication, stop words ratio, number of non ASCII characters, the number of capital letters, and number of new lines. They suggest that these attributes are highly correlated with spam, since many spammers try to get the attention of readers using capitalized letters and different kinds of symbols. With these features, they were able to achieve accuracies of 74% for decision trees, 90% for SVM, and 94% for Naive Bayes. In the last stage of their implementation, they realized that post-comment similarity and topic similarity helped
eliminated comments that are not related to a specific context. Adding these features helped increase the accuracy of the decision tree classifier from 74% to 90%; the other models maintained the same accuracy. Overall, the results of this paper and that on the research of spam tweets [22] largely suggest the importance of examining features in comments that may indicate potential spam as well as detecting the divergence of the content of the comment from the context of adjacent posts on the website to obtain good performance.
Chapter 3

Data Collection

An essential part of this project involves collecting and curating a dataset of user profile comments from the Scratch website. We introduce a new Scratch comments moderation website that serves as a data pipeline for constructing the dataset. This website is intended to allow the Scratch moderators to label comments. The resulting corpora will be used as input to train the models.

The moderation website is also designed to help reduce the need for keeping a blacklist of inappropriate words. By keeping the moderation website updated with recently posted comments from Scratch and continuing the effort of labeling these comments, we can ensure that the machine learning models will be up to date with any new forms of online conversational speech amongst the users of Scratch.

3.1 Data Collection Overview

The dataset is constructed of historical comments found on the Scratch website. The comments were exported from the research database, divided by a visibility flag.

The machine learning task at hand is a classification problem: given a comment, the machine learning model must choose the correct class label. As such, the initial goal was to divide the comments into three labels: (1) censor, if it was flagged by the moderation tool and eventually censored from the Scratch website; (2) investigate, if it was flagged but ultimately not censored; and (3) close, if it was neither flagged
nor censored. However, a close investigation of the comments brought to light an ambiguity between comments that were ultimately censored from the website and comments that were left visible on the website.

1. Example of a comment that was flagged by Cleanspeak, yet still visible on the website.

   *i really wish i could meet you in person... wouldn’t it be funny if we actually could though? :D*

2. Example of a comment that was flagged by Cleanspeak and censored by a Scratch moderator.

   *depending on what state you live in we might be able to meet next weekend:) i found out i’m going out of state for a few days next week:no clue where yet but be prepared it might be where you live. ( i get to pick where so i’ll be able to visit you and meet you in person)*

In the above example, the two comments are very similar in content, yet only one was censored. The ambiguity in the dataset demonstrated the necessity of establishing a “pristine” grouping of the three labels of interest. To that end, we introduce a website that allows the Scratch moderators to review the ambiguous comments, and give a label amongst “Censor,” “Investigate,” “Close,” as described earlier, and also “Skip” if the moderator is unsure of how to categorize the comment.

We instruct the moderators to judge the comment by answering the following question: “Would I want the filter to add this comment to the ticket queue for a moderator to review?” A comment should be labeled “Censor” if, based on the content alone, it should certainly be censored from the website. However, if a comment cannot be labeled as “Censor” adequately without any context, ie. if it is a link to another website, or there is some situational context that is required, the comment should be labeled as “Investigate.” A comment should be labeled as “Close” if it should remain visible on Scratch.
Figure 3-1: The current profanity filter on the Scratch website requires manual entry of new inappropriate phrases. One example of this was the introduction of the phrase “kys,” or “kill yourself.” The term started to propagate throughout the internet in early 2016. The Cleanspeak system was unable to recognize it as harmful content until the phrase was identified and manually added to the blacklist. With the introduction of the Scratch Comments Moderation Website, we hypothesize that the machine learning models will be able to automatically adapt to new forms of inappropriate content.

3.2 Scratch Comments Moderation Website

In this section, we describe the technical mechanics of a website for data collection. This website is designed for the Scratch Moderation team in an effort to produce the data set that will be used to train the machine learning moderation tool. It is also motivated by the need to maintain the proposed moderation tool in the future, so that it can automatically learn any new internet lingo that appears on Scratch. This system allows for reviewing and classifying comments from the website.

3.2.1 Technology Stack

The website is comprised of Node.js and Express on the back end, and React and Webpack on the front end. All code is written in Javascript.
3.2.2 Database

We setup a MySQL database instance for storing the data. The database contains two tables, with the following schemas:

Table 3.1: Comments Table

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>bigint(20)</td>
</tr>
<tr>
<td>comment_id</td>
<td>bigint(20)</td>
</tr>
<tr>
<td>comment_text</td>
<td>blob</td>
</tr>
<tr>
<td>available</td>
<td>tinyint(1)</td>
</tr>
</tbody>
</table>

Comments Table

The comments table includes comments exported from the Scratch website that will be labeled by the moderators. Each row in the table includes a primary id \( id \), which is automatically incremented; a \( comment_id \) and corresponding \( comment\_text \); and
Table 3.2: Results Table

<table>
<thead>
<tr>
<th>column</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>comment_id</td>
<td>bigint(20)</td>
</tr>
<tr>
<td>user_id</td>
<td>varchar(100)</td>
</tr>
<tr>
<td>user_input</td>
<td>enum('Close','Investigate','Censor','Skip')</td>
</tr>
<tr>
<td>time</td>
<td>datetime</td>
</tr>
</tbody>
</table>

a binary value *available*, which corresponds to whether the comment is available for labeling, i.e. it has not yet been assigned a label, or unavailable, if it has been given a label.

We use a consensus system for establishing the final label for any given comment. The consensus check for an individual comment is completed by querying the Results table and counting the number of “Censor”, “Investigate”, and “Close” instances that the moderators gave the comment. We do not use the Skip label to calculate a consensus vote. If there is majority on one of these three labels, the comment’s availability in the Comments table is set to False to indicate that the comment’s label has been finalized according to the majority vote and that it is no longer available to be added to the website’s queue for labeling. This consensus check is completed on comments as they are labeled by the moderators and the data stored in the Results table.

**Results Table**

The results table captures all labels made by the Scratch moderators as they go through the queue of comments on the website. This includes the *comment_id* corresponding to the id of the comment they were labeling, *user_id* to capture the name of the moderator, *user_input* to capture the label that the moderator chose for the comment in question, and the *time* that they labeled the comment.

Each time a user labels a comment, a new row is added to the Results table that captures this information. After the row is added, we query the Results table to determine whether a majority vote for the comment has been established. This keeps the Comments table up to date with the comments that are still available for review.
### 3.2.3 Application Pipeline

#### Data

The initial data import consists of comments from the Scratch website between March 1, 2016 and June 1, 2016 and between October 1, 2015 and January 1, 2016, which are peak periods of user activity. Each comment is labeled with a specific “visibility” level, including “visible” if it is visible on the Scratch website, “censbyfilter” if the comment was flagged by the Cleanspeak filter and eventually censored by a Scratch moderator, “censbyadmin” if the comment was directly censored by a Scratch moderator, and “markedbyfilter” if it was a false positive marked by the Cleanspeak filter.

These three groupings capture informative datapoints. Allowing the moderators to review false positives made by the Cleanspeak filter (the “markedbyfilter”) will help to create a cleaned corpus of data, since there is also the chance of human error, ie. the moderators forgot to take action on some of the comments that the Cleanspeak filter flagged, or the moderators inconsistently applied rules throughout the process.

Because we are working with potentially personally identifiable information, we make sure to anonymize the data before extracting information from the comments.

#### Comment Queue

The moderation website loads comments to the queue one at a time for the user to label. A check is performed such that the comments that are loaded are ones that the user has not seen yet. This is accomplished by selecting a random available comment, and checking the Results table to see that there is not yet an entry corresponding to the user having labeled the comment already.

#### Labeling Data

A comment remains available to label by moderators until a consensus is decided upon, where the consensus is determined using a voting scheme on the moderator’s input. When a consensus is decided, the comment is given a final label, and set to Unavailable in the database. All comments in the database that are still available for
labeling are added to the queue for review. The resulting set of labeled instances of comments in the database is to be used as the input for the machine learning models.

3.2.4 Deployment

We deploy the website using AWS Elastic Beanstalk. We also add a MySQL database to the environment to load the queue of comments to be labelled, as well as to capture the moderator’s input. To authenticate the user, we use the “scratch-auth” utility, which allows our tool to use the same authentication system that the moderation team uses daily.

3.3 Initial Dataset

While this system does use a consensus model to decide on a final label for each comment, for the purposes of this project, we needed an initial dataset to use to train and test the machine learning models and establish a baseline. We established this initial dataset by pulling all of the comments from the Results table and joining them with the Comments table to obtain a set of data consisting of the comment_id, comment_text, and user_input fields. The particular size of the data set is on the same order of magnitude as those that were used in the projects described in the Previous Works section; this approximates to around a few thousand labeled instances as training data.

Of note is that this resulting data includes comments that have not yet been given a final label. In the long term, future iterations will not have to use prematurely labeled comments, but comments that have been given a final label after a consensus vote. However, for this initial set of data, we make this concession.

3.4 Preparing Data

Because the comments pulled as described in the previous section have not yet been given a consensus backed label and because the website currently has no option that
allows moderators to fix accidental misclicks or mislabels, we found it necessary to revisit the comments and their corresponding labels. Across the machine learning model iterations, we produced a list of comments that the model incorrectly classified, where the moderator’s label is compared to as the source of truth. This list often had a few mislabels.

Regular meetings with the Scratch moderation team were established to clean the corpus of comment data. The moderators would review the list of comments that the machine learning model incorrectly classified to determine if there were any accidental errors in their labeling. If they found any ambiguous comments, the comments were discussed at length and, after a consensus was established, given final labels.

This process was repeated at length for multiple sessions and brought many issues to light, the main being uncovering inconsistencies in current views amongst the moderation team. One particular example is the issue of the word “sucks.” We found inconsistencies in the way that comments containing the word was labeled. For example, the comment “you know what I think I’ll just do it now my hw sucks” was Closed by a moderator, while the comment “hi, u suck” was Censored by a moderator. In the first comment, the user is complaining about the arduousness of a homework assignment, whereas in the second comment, the user is provoking another Scratch user. In a similar vein, the comment “its so boringgg” was a point of contention. The moderators revealed that if such a comment were found on a user’s project, it would be considered unconstructive and censored, but if it were found anywhere else on the Scratch website it would be fine. These reviews revealed the importance of context in determining whether comments are appropriate.

3.5 Open Source

We intend to eventually release the dataset under an open source license after the completion of the project. The Scratch website currently has a page called “Scratch for Developers, which hosts open source projects currently maintained by the Scratch Team at MIT. Once the dataset has been cleaned and anonymized, we would like to
share it publicly on the developer page.
Chapter 4

Algorithmic Models

The core component of this thesis involves constructing machine learning models that will be used to perform language processing tasks automatically. We construct this as a binary classification problem. The models will, given a user comment as input, classify whether the comment is inappropriate or not. We employ several supervised machine learning techniques towards the task of classifying Scratch comments as inappropriate or appropriate.

The eventual goal for these machine learning models is that each comment that the model determines to be inappropriate will be flagged on Scratch and added to the ticket queue for the Moderation team to review.

4.1 Data Preprocessing

The data set used to train and test the machine learning models are extracted from the website’s MySQL database, as explained in Chapter 3. We combined the “Investigate” labels with the “Censor” labels. We do so because by definition, the “Investigate” label is designated for comments that require further review. We want the model to flag all comments that the moderators want to review and that will eventually be added to the moderation ticket queue. This condenses the dataset into two classes: “Censor” and “Close” labels.

The dataset is pre-processed, features are extracted, and the models are built with
the resulting dataset and features as input.

4.1.1 Bag of Words Representation

To detect forms of abuse, harassment, and bullying, we use natural language processing techniques to extract textual information as features for the models. We turn the text into a bag of words representation. This representation entails assigning each unique word occurring in any comment of the training set with an integer id, then counting the number of occurrences of each word and setting that as the value corresponding to the word’s integer id [24].

The corpus is thus turned into a matrix where each row represents a comment and each column represents a word occurring in the corpus, where the row entries correspond to the word counts in a given comment. Using Scikit-Learn’s utilities, we extract numerical features from the comments by (1) tokenizing the strings, (2) counting the occurrences of tokens for each comment, and (3) normalizing and weighting the importance of each token.

4.1.2 Tokenization, Token Counting, and Stop Words

The goal of tokenization is to break up a sentence or paragraph into specific words, or tokens [24]. For instance, we can consider comments on a word level basis, multiple word basis (in the form of bi-grams, tri-grams, etc.), as well as on a character level basis. We employ Sci-kit Learn’s CountVectorizer to both tokenize and count the number of occurrences of these words, phrases, and characters. The CountVectorizer converts the text input to a matrix of token counts. The rationale here is that commonly appearing characters and / or words may be strongly correlated with inappropriate content, and that the counts of each token may be informative in classification.

We also use CountVectorizer to remove stop-words from the corpus. Stop-words are defined as commonly used words such as “the”, “a”, “an”, etc. and are determined to have little effect as a feature [25]. Removing stop-words also helps to remove any
redundancy in the data. Since the corpus is tokenized and extra features will be extracted, the dimensionality of the data will be very large. By using a stop list, we can significantly reduce the number of tokens that the system has to store.

4.1.3 Stemming and Lemmatizing

We also use stemming and lemmatizing to reduce different forms of a word into their base form [26], and to further reduce dimensionality of the feature space. To do this, we use NLTK’s advanced text processing mechanisms including the WordNet lexicon for lemmatization, and the LancasterStemmer for stemming.

Stemmers remove morphological affixes from words, leaving only the word stem [26]. Stemming usually is a crude heuristic process that chops off the ends of words; in the process it often includes the removal of derivational affixes [26]. The most common algorithm for stemming English is the Porter algorithm [26], however we find from initial experimentation of the data that the Lancaster algorithm provides greater performance.

On the other hand, lemmatization uses a vocabulary and morphological analysis of words to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma [26]. Unlike stemmers, lemmatizers require a complete vocabulary and morphological analysis to correctly lemmatize words.

To accomplish this, we supplement the comment’s text with the corresponding
Figure 4-2: An example of how the WordNet stemmer completes lemmatization using part of speech tags.

<table>
<thead>
<tr>
<th>Word</th>
<th>Stemmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>(love, v)</td>
<td>love</td>
</tr>
<tr>
<td>(loved, v)</td>
<td>love</td>
</tr>
<tr>
<td>(loving, n)</td>
<td>loving</td>
</tr>
<tr>
<td>(loving, a)</td>
<td>loving</td>
</tr>
<tr>
<td>(choose, v)</td>
<td>choose</td>
</tr>
<tr>
<td>(chose, v)</td>
<td>choose</td>
</tr>
</tbody>
</table>

Part of Speech tags to help the lemmatizer correctly lemmatize the word into its base form. Knowing the part of speech of a word helps intelligently remove the suffix of the word into its lemma form. For example, the word “loves” is a Verb and is lemmatized to “love,” its base form. The WordNet stemmer, when given (word, part-of-speech tag) pairs, collapses tense, aspect, and number marking [27].

### 4.1.4 TF-IDF Weighting

Given the size of the comment corpus, we would like to assign greater weight to words that carry more information. By using just the counts of tokens, we find that common words such as “a,” “is,” etc. will have high count occurrences, and therefore carry very little meaning about the contents of the comments. To re-weight the count features we use the TF-IDF transform. TF-IDF means term-frequency multiplied by inverse document-frequency [28]. TF-IDF weighting assigns a higher weight to tokens that occur many times within a small number of comments, and a lower weight to terms that occur in all comments [28]. This selective weighting thus assigns more weight to rarer yet more interesting terms.
4.1.5 Sci-kit Learn Pipelines

We leverage Sci-kit Learn’s pipelines for text feature extraction. Pipelines allow one to chain estimators together, in particular the CountVectorizer transformation that tokenizes the text corpora and counts the tokens, with the TF-IDF transformation that reweights the token counts [28]. It provides ease for standardizing and streamlining the pre-processing and evaluation of the various classifiers we experiment with.

4.2 Part of Speech (POS) Tagging

We conduct part of speech tagging on the acquired dataset. This will help extract features from the corpus data, which should aid the classifier in determining how to label inputs. In particular, we use the POS tags to help with lemmatizing.

We use Carnegie Mellon University’s Twitter Part of Speech Tagger [16]. We hypothesize that the informal text that characterizes Tweets is similar to that of Scratch comments, and that the Part of Speech tagger should be similarly successful in tagging Scratch comments.

Running the tagger one comment at a time takes much longer because it requires restarting the tagger. To overcome this, we load the entire data file into the tagger and tag the file. The resulting Part of Speech tags are saved to disk and loaded for use during training.
Table 4.1: NLTK Sentiment Analysis Applied to Scratch Comments

<table>
<thead>
<tr>
<th>Comment</th>
<th>Compound</th>
<th>Negative</th>
<th>Neutral</th>
<th>Positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let’s run chara <em>chara growls and throw his blade at the wall in front of u</em></td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>he is stupid</td>
<td>-0.5267</td>
<td>0.63</td>
<td>0.37</td>
<td>0.0</td>
</tr>
<tr>
<td>Awh, thanks ! j3 But I haven’t shown my main OC yet x’D</td>
<td>0.7263</td>
<td>0.0</td>
<td>0.596</td>
<td>0.404</td>
</tr>
</tbody>
</table>

4.3 Bad Word List

We incorporate a check against a list of common inappropriate terms as collated by [32], and append it with phrases from Scratch’s blacklist of terms.

While the profanity blacklist helps to narrow down on typical expletives, it isn’t enough to capture behavior specific to the Scratch website. Regular meetings with the Scratch moderation team helped to solidify the types of behavior they deemed inappropriate. This includes: sharing of personally identifiable information including e-mail addresses, phone numbers, names, and addresses; any hints of flirtatious behavior; links to FlockMod, a free real time online group drawing app that circumvents the review of Scratch moderators; links to Skype, Facebook, Google Hangouts; references to hentai, provocative Anime characters, etc.

4.4 Sentiment Analysis

We also incorporate sentiment analysis in our machine learning models. We leverage the natural language toolkit NLTK and its sentiment analysis tool, SentimentIntensityAnalyzer [33]. This module incorporates the VADER sentiment analysis [34] in identifying sentiment-relevant, word level properties of input text. It quantifies the sentiment of a given text, giving measurements on the overall sentiment of a text, as well as the overall negative, neutral, and positive sentiment of a text.
4.5 Feature Selection

Selecting relevant features has an enormous impact on the learning method’s ability to extract a good model. Although it’s often possible to get decent performance by using a fairly simple and obvious set of features, there are usually significant gains to be had by using carefully constructed features based on a thorough understanding of the task at hand.

4.5.1 Feature Space

We use bag-of-words features to represent the corpus data as an unordered set of words and characters. The features include:

1. **Word unigrams and bigrams.** We use vectors to encode the counts of word unigrams (ie. single words) and word bigrams (ie. sequence of two words).

2. **Character unigrams, bigrams, etc.** We also examine character counts, where character unigrams are counts on single characters, and character bigrams are counts on sequences of two characters.

3. **TF-IDF weighting.** The TF-IDF, or term frequency times inverse document frequency, is a measure of the importance of a word in a document within a collection of comments, thereby taking into account the frequency of occurrence of a word in the entire corpus as a whole and within each comment.

4. **Lexical features.** We use a number of handcrafted features that we hypothesize to capture common traits of spam and abusive content.

   (a) Length of comment

   (b) Number of words

   (c) Number of characters

   (d) Number of all-capital words

   (e) Longest word
5. **Sentiment analysis.** We incorporate NLTK’s SentimentIntensityAnalyzer [33] for sentiment analysis. Given a comment, the analyzer returns an overall polarity score, which is a value indicating sentiment strength. The score is positive for positive sentiment, and negative for negative sentiment. We also use the overall positive, neutral, and negative scores that the analyzer outputs.

Since explicit verbal abuse involves the use of stereotypical slang and profanity as recurring patterns, those aspects lend themselves nicely to supervised learning algorithms. We also hypothesize that instances of cyberbullying where the abuse is more indirect and does not involve the use of profanity or stereotypical words are likely to be misclassified by supervised learning methods.

### 4.5.2 Regularization

L1 regularization uses a penalty term which encourages the sum of the absolute values of the parameters to be small [30]. L2 regularization encourages the sum of the squares of the parameters to be small [30]. It has frequently been observed that L1 regularization causes many parameters to equal zero, so that the parameter vector is sparse. This makes it a good candidate in feature selection, in which less informative features should be ignored.

We use L1 regularization for feature selection to prevent overfitting on the data. As supported by [30], using L1 regularization for feature selection is effective for
supervised learning. Since we have only a moderately sized sample of data, we need to use regularization, which encourages the fitted parameters to be small [30], in order to overcome overfitting.

### 4.5.3 Univariate Feature Selection

In addition to using regularization, we use univariate feature selection to select the best features. In particular, we use Scikit-Learns SelectPercentile [35], which removes all but the highest scoring percentage of features. We use F1-score as the basis for measuring how good a feature is.

### 4.6 Supervised Machine Learning Models

We adopt a bag-of-words supervised machine learning classification approach to identifying the sensitive theme for a given comment. As a baseline, we investigate the use of four supervised learning algorithms, including Maximum Entropy classifier, Support Vector Machines (SVMs), Naive Bayes, and also Random Forest. We use Python’s Scikit-Learn library to develop these models.

Our investigation of previous related work on NLP tasks has shown that the SVM and Naive Bayes algorithms work well as baselines for detecting spam and bullying behavior, hence we include these in our study as a baseline.

Maximum Entropy is a probability distribution estimation technique widely used for a variety of natural language tasks, including part-of-speech tagging, sentiment analysis, and text segmentation [36]. We include this in our experiments to determine how applicable it might be towards the task of text classification.

Random forests, introduced by Breiman in 2001 [37], are aggregated classifiers composed by ensembles of trees independently induced. The classification of a new instance is made by a voting system, where the instance is classified by each individual tree and the class votes are counted [37]. Since its introduction, the random forests framework has been extremely successful as a classification and regression method [38]. Although it has not been a commonly used model for text classification, we
choose to include it in our evaluation given the performance the model has been known for in other more general purpose classification tasks.

Each of the classifiers are trained and evaluated in the same fashion. The classifiers are fed the pre-processed comment corpora as input, trained on the data, and then cross-validated. We also tune the hyper-parameters of the estimator by leveraging Scikit Learn’s GridSearchCV, which exhaustively considers all parameter combinations [31]. Given the shortage of the amount of data we work with, cross-validation helps to leverage all of the datapoints we have.

After the classifiers are fit to the comments and their labels, they are serialized to disk and used to make predictions in the future.
Chapter 5

Evaluation

This part will examine what combination of features and clustering algorithms will lead to the highest accuracy. It will also take into account scalability, as the model would ideally be integrated into Scratch’s infrastructure. A unique feature of our project is the goal of making our algorithmic model practical for live production use. While these algorithms must be accurate, they also need to be adequately fast, and scalable, as Scratch is used extensively around the world.

5.1 Data Evaluation

We use a dataset comprised of 2565 labeled instances of Scratch comment data. 1280 comments are labeled “Censor” and 1286 comments are labeled “Close.” The “Censor” labels combine the “Investigate” and “Censor” labels. We use an 80% training and 20% testing split for each of the four models.

5.2 Bag of Words Models

Below are charts detailing the performance of each of the four supervised machine learning models: Maximum Entropy (Logistic Regression), Support Vector Machine, Naive Bayes, and Random Forest.
### Table 5.1: Maximum Entropy Results

<table>
<thead>
<tr>
<th>Maximum Entropy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word n-grams</td>
<td>0.69</td>
<td>0.68</td>
<td>0.67</td>
</tr>
<tr>
<td>char n-grams</td>
<td>0.6</td>
<td>0.6</td>
<td>0.59</td>
</tr>
<tr>
<td>sentiment</td>
<td>0.57</td>
<td>0.57</td>
<td>0.54</td>
</tr>
<tr>
<td>lexical features</td>
<td>0.6</td>
<td>0.57</td>
<td>0.49</td>
</tr>
<tr>
<td>all features, lemmatizing</td>
<td>0.65</td>
<td>0.65</td>
<td>0.65</td>
</tr>
<tr>
<td>all features, lemmatizing w/POS</td>
<td>0.7</td>
<td>0.69</td>
<td>0.69</td>
</tr>
<tr>
<td><strong>all features, stemming</strong></td>
<td><strong>0.73</strong></td>
<td><strong>0.73</strong></td>
<td><strong>0.73</strong></td>
</tr>
<tr>
<td>all features, no normalizing</td>
<td>0.7</td>
<td>0.7</td>
<td>0.7</td>
</tr>
</tbody>
</table>

### Table 5.2: SVM Results

<table>
<thead>
<tr>
<th>SVM</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word n-grams</td>
<td>0.29</td>
<td>0.54</td>
<td>0.38</td>
</tr>
<tr>
<td>char n-grams</td>
<td>0.29</td>
<td>0.54</td>
<td>0.38</td>
</tr>
<tr>
<td>sentiment</td>
<td>0.62</td>
<td>0.6</td>
<td>0.54</td>
</tr>
<tr>
<td>lexical features</td>
<td>0.56</td>
<td>0.57</td>
<td>0.56</td>
</tr>
<tr>
<td>all features, lemmatizing</td>
<td>0.6</td>
<td>0.6</td>
<td>0.6</td>
</tr>
<tr>
<td>all features, lemmatizing w/POS</td>
<td>0.58</td>
<td>0.58</td>
<td>0.57</td>
</tr>
<tr>
<td>all features, stemming</td>
<td>0.58</td>
<td>0.58</td>
<td>0.58</td>
</tr>
<tr>
<td><strong>all features, no normalizing</strong></td>
<td><strong>0.61</strong></td>
<td><strong>0.61</strong></td>
<td><strong>0.6</strong></td>
</tr>
</tbody>
</table>

### Table 5.3: Naive Bayes Results

<table>
<thead>
<tr>
<th>Naive Bayes</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word n-grams</td>
<td>0.65</td>
<td>0.64</td>
<td>0.62</td>
</tr>
<tr>
<td><strong>char n-grams</strong></td>
<td><strong>0.71</strong></td>
<td><strong>0.68</strong></td>
<td><strong>0.67</strong></td>
</tr>
<tr>
<td>sentiment</td>
<td>0.25</td>
<td>0.5</td>
<td>0.33</td>
</tr>
<tr>
<td>lexical features</td>
<td>0.25</td>
<td>0.5</td>
<td>0.33</td>
</tr>
<tr>
<td>all features, lemmatizing</td>
<td>0.63</td>
<td>0.53</td>
<td>0.43</td>
</tr>
<tr>
<td>all features, lemmatizing w/POS</td>
<td>0.59</td>
<td>0.51</td>
<td>0.37</td>
</tr>
<tr>
<td>all features, stemming</td>
<td>0.58</td>
<td>0.52</td>
<td>0.42</td>
</tr>
<tr>
<td>all features, no normalizing</td>
<td>0.71</td>
<td>0.53</td>
<td>0.4</td>
</tr>
</tbody>
</table>
Table 5.4: Random Forest Results

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word n-grams</td>
<td>0.65</td>
<td>0.65</td>
<td>0.64</td>
</tr>
<tr>
<td>char n-grams</td>
<td>0.69</td>
<td>0.69</td>
<td>0.69</td>
</tr>
<tr>
<td>sentiment</td>
<td>0.57</td>
<td>0.55</td>
<td>0.51</td>
</tr>
<tr>
<td>lexical features</td>
<td>0.56</td>
<td>0.55</td>
<td>0.53</td>
</tr>
<tr>
<td>all features, lemmatizing</td>
<td>0.7</td>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>all features, lemmatizing w/POS</td>
<td>0.69</td>
<td>0.69</td>
<td>0.69</td>
</tr>
<tr>
<td>all features, stemming</td>
<td>0.7</td>
<td>0.7</td>
<td>0.69</td>
</tr>
<tr>
<td><strong>all features, no normalizing</strong></td>
<td><strong>0.71</strong></td>
<td><strong>0.71</strong></td>
<td><strong>0.71</strong></td>
</tr>
</tbody>
</table>

5.2.1 Discussion of Results

The Maximum Entropy classifier produced the highest all around scores, with a precision of 73%, recall of 73%, and F1-score of 73%. Combining all of the features and normalizing the text using stemming helped achieve these scores. The results provide support for the use of Maximum Entropy in this particular application of natural language processing towards classifying inappropriate content on Scratch. Moreover, given that Maximum Entropy is a linear model and takes less time to train, it would be favorable to use Maximum Entropy over the next best performing classifier, Random Forest.

Interestingly, the Random Forest classifier also achieved similar results, with a precision, recall, and F1-score of 71%. It performed best with a combination of all features, and no normalization. Employing lemmatizing and stemming degraded performance by 1-2%. More accurate ensembles require more trees, which causes the model to become slower. Thus, there is a tradeoff between accuracy and run-time performance. Given the degree to which the model would have to scale on the Scratch website, the Maximum Entropy classifier, which is a linear model, would be preferred.

Our use of Support Vector Machine gave poorer performance, with a precision and recall of 61% and F1-score of 60%. This is contradictory to previous work that has been done on text classification, as SVMs have historically been known to do well on these tasks. Similar to the Random Forest classifier, we maximized performance
Table 5.5: Model Comparison by F1-score

<table>
<thead>
<tr>
<th>Model</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Entropy</td>
<td>0.73</td>
</tr>
<tr>
<td>SVM</td>
<td>0.60</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>0.67</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.71</td>
</tr>
</tbody>
</table>

by combining all features, and not using normalization. Adding any lemmatizing or stemming degraded performance by a 1-2%.

The Naive Bayes model worked best with simply character n-grams and word n-grams, achieving a precision of 71%, recall of 68%, and F1-score of 67% using only character n-grams as a feature. It appears that sentiment and lexical features did not help the model.

5.2.2 Stemming and Lemmatization

Stemming appeared to help the Maximum Entropy model. Otherwise, the use of stemming and lemmatizing towards the other models did not result in any gains in performance. Supplementing the lemmatizers with POS tags from the comments also did not prove to be fruitful.

There are a few possible reasons for why these normalization techniques did not help performance. Stemming increases the risk of losing information by ignoring the difference between words of different inflectional forms [25]. It increases recall while potentially arming precision [25]. Lemmatizing, which does a full morphological analysis, provides at most a modest gain in recall [25].

5.2.3 Other Features

Some features that did not help performance include contextual features, including the parent comment of the comment in question, how long the author has been a member of Scratch, and the number of moderation actions taken on the author.
5.3 False Positives and Negative Cases

Some of the comments that the models had difficulty with did not necessarily have any inappropriate words or signs of verbal abuse. For example, there were some comments where the Scratch user mentioned their own personally identifiable information, putting them at risk. The models had difficulty registering that these were not appropriate.

The following examples are found using the best performing classifier, the Maximum Entropy model.

1. The model incorrectly labeled these comments as “Close,” that they were appropriate. However these comments are dangerous as they give away personally identifiable information.

   (a) A user giving away personally identifiable information (birthdate, name).
   
   I’m 10, my birthday is on october 21 (21! ya i know) i like minecraft and my actual name is John Smith

   (b) A user giving location information for a meet up.
   
   If you guys won’t to meet me in person come to target by party city, monkey joes and petco

2. The model also had difficulty parsing some comments that did not necessarily have abusive content.

   (a) This comment features a potential case of emotional distress.
   
   to be honest i was controlling myself very well in the gym. normally im crying uncontrollably and shaking and rocking back and forth sometimes screaming... i was very collected. im kinda proud of that

   (b) This comment indicates that occurrence of recoloring of projects, which is forbidden on the Scratch website.

   Anyone who recolors my characters...

---

\(^1\text{Names and identifying details have been changed to protect the privacy of individuals.}\)
3. The model had some false positives on comments that looked similar to spam.

(a) Here the comment has capitalized letters and is very long in length.

\[
I \text{ KNOW WHERE YOU LIVE}.............................
\]

..........................................................................................................
..........................................................................................................
..........................................................................................................
..........................................................................................................
...........................................................................................................
...........................................................................................................
...........................................................................................................
...........................................................................................................
...........................................................................................................You live in a house

on Earth
Chapter 6

Future Work

6.1 Working with communities

Currently, the labeling of comments on the Scratch comments moderation website is carried out only by the Scratch Moderation team. The machine learning algorithms are trained on their labels, as the Moderators are in this sense the source of truth by which the models learn from. Despite the relative ease and speed that they have in labeling comments using the website, machine learning models require a massive amount of data to achieve best results. This begs the question of whether the data labeling can be opened up to people outside of the Scratch Moderation team.

Scratch currently has many initiatives involving public schools throughout the United States. One potential idea would be to reach out to the educators of these schools. Teachers that use Scratch in their curriculum have privileges to moderate their students; their opinions may also align with the Scratch moderators.

We may also consider reaching out to learning communities to collaborate on labeling the datasets. However, there are inherent differences between moderation teams between Scratch and other websites. DIY [39], for example, is another online community for kids that allows them to learn new skills and share knowledge about a wide range of topics. Because these varied topics include Science and Anatomy, the presence of toilet humor, ie. jokes about animal scat, is allowed throughout the website. However, such content would be seen as inappropriate on Scratch.
This brings to question exactly how much dissonance we would be willing to consider. By expanding the data labeling to resources outside of the Scratch Moderation team, there is a potential that the source of truth would become fuzzy. However, according to a Yahoo paper on detecting inappropriate content [13], in which data labeling was outsourced to Amazon Mechanical Turk, it was shown that external workers can produce results comparable to results established in laboratory studies.

6.2 Specializing comments

The data collection piece of this project prompts potential changes in current moderation policies. An inherently unique characteristic of this corpus is the variety of topics that the comments are about. For example, topics range from Anime to Scratch characters to world events and politics. Given the great breadth of topics that these comments feature, it might be useful to designate certain moderators on the Scratch Moderation team to become specialized on certain topics. For example, a moderator could become specialized on anime-centric comments and learn to distinguish which Anime related phrases are inappropriate.

6.3 Revising labels

Currently, the data collection website for labeling Scratch comments lacks a feature for fixing accidental misclicks or mislabels. A new feature for the website would be the ability to revisit the comments that a given Scratch moderator has labeled and review and fix labels. This would place less burden on the data cleaning process.

6.4 Foreign Languages

Currently, the scope of this project considers only comments written in the English language. All text processing techniques were done on English comments, and foreign languages were ignored from the corpus. Given the scale that Scratch operates on and
the various countries that use Scratch, it will be important for further moderation attempts to cover multiple languages as well.

6.5 Expanding towards Scratch Projects and Studios

The corpus of data used for building the machine learning models described in this thesis was taken from comments found on User Profiles. However, there is a clear distinction in the types of comments that are found on User Profiles compared to comments found on Scratch Projects and Studios. For example, the incidence of roleplay, i.e., where the Scratch users pretend to be roleplaying as another character, is very common in Studios, but not as common on User Profiles. The presence of roleplay is a unique trait of online conversations on the website, and demonstrate the distinct culture of Scratch.

Moreover, moderation policies for these three contexts are also distinctive. On Studios and Projects, comments that disparage a User’s work are in violation of Scratch’s community guidelines and are considered inappropriate. Thus, the context of the comments greatly affects moderation actions.

6.6 Scratch POS Tagger

This thesis employs Carnegie Mellon University’s Twitter Part of Speech (POS) tagger to do POS tagging on Scratch comments. There are certain similarities between Twitter comments and Scratch comments, however there is a very distinct difference in terms of the culture of the two social platforms. The disparities inherent in the corpora data may cause the Twitter POS tagger to be not as effective on Scratch comments, so it would be a useful effort to develop an in-house POS tagger trained on Scratch comments.
Chapter 7

Conclusion

In this paper we aim to develop a state-of-the-art method for detecting abusive language in user comments, while also addressing deficiencies in the field. Specifically, this paper has the following contributions:

1. We develop a supervised classification methodology with NLP features. We use and adapt several of the features used in prior art in an effort to see how they perform. This thesis evaluates the use of natural language processing and supervised machine learning strategies towards the detection of inappropriate online comments written by children on the Scratch website. The highest performing combination of features and machine learning models achieves an overall F1-score of 73%, precision of 73%, and recall of 73%. These figures are in line with similar attempts at applying natural language processing techniques towards the classification of profane and obscene online content.

2. We produce a new corpora of several thousand user comments collected from Scratch user profiles. This set includes one judgment per comment, indicating whether the comment is appropriate or inappropriate.

3. This work also introduces a novel comment moderation website which is used as a data pipeline for training the machine learning models. By having the models retrained on new comments reviewed and classified via this tool, we present a unique solution to the issue of maintaining a blacklist of inappropriate words.
and phrases. The webtool will allow the models to adapt to new and evolving forms of internet speech.
Bibliography


[8] https://support.twitter.com/articles/69214


[10] https://www.youtube.com/yt/policyandsafety/communityguidelines.html


[27] http://sentiment.christopherpotts.net/stemming.html


ence on Machine learning (ICML ’04). ACM, New York, NY, USA, 78-.
DOI=http://dx.doi.org/10.1145/1015330.1015435


[32] https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-
Otherwise-Bad-Words/blob/master/en


for Sentiment Analysis of Social Media Text. Eighth International Conference on
Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

feature-selection

for Text Classification.


[38] Misha Denil, David Matheson, Nando de Freitas. Narrowing the Gap: Random
Forests In Theory and In Practice.

[39] https://diy.org/guides/guidelines