Online Abuse Detection

Dinesh Kannan  Rajkumar Murukeshan
kannan.d@husky.neu.edu  murukeshan.r@husky.neu.edu

ABSTRACT
With increasing personal content and opinions on social web platforms, there is an important need to protect their owners from abuses and threats. With the user bases of popular platforms like Reddit, Facebook etc. clocking over 500 million and growing, a time-efficient and privacy protective solution to tackle cyberbullying is an automated one that understands a user’s comment and flags them if inappropriate. In this paper, we attempt to tackle this problem of an online abuse detection system. We experiment with a series of our own models and compare them with techniques that were tried already. Our deep learning models perform better than other traditional approaches that exist for this problem. We also analyze and understand the nature of the conversations through the performance of our models.

I. INTRODUCTION
Today, almost everyone has a social profile online, innocently putting all their personal information for anyone to see. Online interactions like VoIP calls, text chats, sharing ideas/opinions on a public forum are more prevalent in the recent times. A significant portion of online participants are kids and teenagers. One in five 8 to 11 year olds and seven in ten 12 to 15 year olds has a social media profile[1]. While there are useful conversations, hate and abuse is widespread in these online interactions. A study finds that almost 1 in 4 young people have come across racist or hate messages online[2]. The most disturbing fact of all is that 1 in 3 children have been a victim of cyberbullying[3]. This is a clear indication that these online interactions should be controlled by the owner/administrator of the platform to eliminate or at least to minimize online harassment. The only way such a control can be implemented, at a large scale without compromising individual privacy, is by automating the process of understanding the language to detect potential abuse in a comment and reporting users if they are found to be misbehaving.

We aim to solve this problem by solving the most important and difficult part in the aforementioned process, understanding a comment and telling if it is safe or not. This requires a great deal of natural language processing / understanding and a significant amount of machine learning. An efficient implementation of such a system takes in a piece of text (a comment) as input and produces a binary label, ‘abuse’ or ‘not-abuse’ as output. An example is depicted in Table 1.

<table>
<thead>
<tr>
<th>Message</th>
<th>Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thank you for your contribution. You did a great job!</td>
<td>not-abuse</td>
</tr>
<tr>
<td>People as stupid as you should not take this task</td>
<td>abuse</td>
</tr>
</tbody>
</table>

Table 1: An example of inputs and outputs to our system

II. REFERENCE / RELATED WORK:
Online harassment statistics discussed in the previous section are from studies conducted very recently. This is a clear indication that not much work has been done to prevent online abuse automatically. Very little has been done commercially and academically.

One of the early known efforts in abuse classification dates back to 2009 where, Dawei Yin and his colleagues explored a context based approach[4]. Their technique employed content features, sentiment features and context features of a comment. They documented that a supervised-learning approach using an SVM classifier with n-grams proved to be better that conventional methods using TF-IDF.

One other approach that seems to be inspired from commercial rule-based spam filtering using blacklists is seen in the work[5] of S.O.Sood, et.al in 2012. They seem to use blacklists along with an edit distance metric. They showed that their approach was better for online profanity detection compared to the ones which existed earlier.

Yahoo seems to automatically moderate online abuses, which they documented in a recent publication[6]. Yahoo used a combination of parser, lexical and syntactic features to train a classifier using supervised machine learning techniques.

Google Jigsaw recently published a paper[7] recently that used data from the Wikipedia Detox project[8]. The paper discusses the effectiveness of logistic regression and multi-layer perceptron for abuse classification. The paper also compares their approach with a human baseline. Methods discussed in this paper is actually put to use in Google’s Perspective API - an API that takes in a piece of text and returns if the text is an abuse.

All the published works mentioned above seem to focus more on the data through progressive feature selection. There is no effort put in to explore deep learning techniques to detect abuses in online comments. Deep learning methods often require little to no feature engineering. However, there has been some significant work done in the past that employs deep learning for text but for completely different problems.


Ji Young Lee and his colleague worked on building a sequential short-text classifier[10]. Their classifier employed the use of a recurrent neural network. They prove that their model achieves state-of-the-art results on three different datasets for dialog act prediction.

**III. NOVELTY:**

Our work takes a more recent, relevant and manually annotated dataset of online user comments from the Wikipedia Detox project and explores different deep learning architectures. We also craft a convolutional layer with an RNN to capture higher level representations. No other work to detect abuses has studied the importance of these higher level representations.

For example, “I loved this TV-series on HBO” can be chunked as [“I loved this”, “TV-series on HBO”] instead of [“I”, “loved”, “this”, “TV-series”, “on”, “HBO”]. Our work is novel in this way. We use some common machine learning models used previously as a baseline to compare the results of our experiments.
IV. METHODOLOGY:

Data
Quality datasets for this task are very sparse and small due to unpopularity of this research problem. For our experiments, we chose to use the dataset released by Wikimedia for the Wikipedia Detox project. There are currently two distinct types of data included:

1. A corpus of all 95 million user and article talk diffs made between 2001–2015.
2. An annotated dataset of 1m crowd-sourced annotations that cover 100k talk page diffs (with 10 judgements per diff) for personal attacks, aggression, and toxicity.

Since our baselines deal with evaluating personal attacks in online conversations, we only consider personal attacks for our experiments.

Data Preprocessing
Even though we had 1 million annotated comments, they were not ready to be consumed by our models. We had to process the data using regular expression and natural language processing techniques. Our preprocessing efforts are outlined below.

1. Text cleansing and tokenization: The raw dataset was abundant with numerous placeholders and special characters like ‘NEWLINE_TOKEN’ etc. We cleansed the text and tokenized into pure lowercase english words in this step.
2. Label selection using majority voting: Due to the crowdsourced nature of this dataset, one comment was labelled by several workers. Often, a comment had different labels due to the difference in perception of the workers. We chose a single label for a comment using majority voting.
3. Data representation: For each experiment, we had to represent the data differently. This involved frequency counting, tf-idf vectorization, one-hot embedding, word embedding with pre-trained word vectors, sequence padding etc.
4. Train-validation-test split: After preprocessing, we create equal samples for both the categories ‘abuse’ and ‘not-abuse’. For training, tuning and evaluating our classifier, we fix 60% of data for training, 20% for validation and 20% for testing.

After our efforts to improve the quality of data for our experiments, we ended up with 7800 samples (each category) for training and 2600 samples (each category) for validation and testing.

Features
Features and data representation varies for each experiment. They are described separately in the experiments section.

V. METHODS:

Naive-Bayes
We first used a naive-bayes classifier for the task to detect abuse in a comment. The input features were just a bag-of-words. This method can particularly be useful if there is some conditional independence between the features, which is often a poor assumption for text classification tasks. A naive-bayes classifier learns the probability of the features given a class from the training set. During prediction, it uses bayes theorem to predict the probability of the class given the features as shown in Fig 1.
Logistic Regression
A logistic regression classifier explores any linear dependencies in the features. The input features were represented as a tf-idf vector that considers all unigram, bigrams and trigrams. The algorithm learns the coefficients in the linear combination of features and maps it to a probability using the function in Fig 2.

\[
p(C_k | x) = \frac{p(C_k) p(x | C_k)}{p(x)}
\]

*Fig 1: Naive-Bayes prediction*

Multilayer Perceptron - Feed forward neural network
A multilayer perceptron is a feed-forward neural network which takes in data represented in the one-hot vector representation and returns probability of the final labels through a sigmoid activation function. A multilayer perceptron is formally represented in Fig 3.

\[
Pr(Y_i = 1) = \frac{e^{\theta_i^T x_i}}{1 + \sum_{k=1}^{K-1} e^{\theta_k^T x_i}}
\]

*Fig 2: Logistic Regression prediction*

Convolutional Neural Network
A convolutional neural network (CNN) is a class of feed-forward neural network architecture which takes an input and transforms it several times through a sequence of layers. The CNN finally outputs the probability for each class label. CNNs are good for highly local data because they make local connections. This explores if a word in a comment is more dependent on the surrounding words. CNNs for NLP is depicted in Fig 4.

*Fig 4: CNN for natural language processing*[11]
Recurrent Neural Network with LSTM
A recurrent neural network (RNN) is an alternative to the feed-forward approach. A RNN trains in terms of time steps, where at each step it factors in the current word and history of words seen. We use Long-Short-Term-Memory (LSTM) with RNN. LSTMs are modules consisting of a set of gates that adds memory to RNN. This is particularly useful for capturing long term dependencies, holding information about an important time step or word in the past. Because of this property, RNN with LSTMs are greatly useful with sequences like texts. An LSTM can be formally represented as in Fig 5.

\[
\begin{align*}
    f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
    i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
    o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
    c_t &= f_t \odot c_{t-1} + i_t \odot \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
    h_t &= o_t \odot \sigma_h(c_t)
\end{align*}
\]

*Fig 5: Forward pass of LSTM with forget gate*\(^{[12]}\)

Recurrent Neural Network with LSTM and a convolutional layer
Based on our experiments, we found out that RNN with LSTMs take a long time to train and did not show a substantial improvement in accuracy compared to the baselines. Hence, we came up with adding a convolutional layer just before the LSTM modules. This improved speed primarily because the inputs were not individual words but a sequence of logical chunks of words. This higher level representation also worked better to capture long term dependencies for the model to use.

VI. EXPERIMENTS:
Evaluation metrics
There are a lot of metrics to use for classification. We decided not to complicate our analysis by calculating a lot of metrics. Our primary metric is the f1 measure, a balanced precision and recall measure. An f1 score is the harmonic mean of precision and recall used to assess the precision - recall tradeoffs and it is a popular metric for classification tasks. Our balanced f1 score is calculated as seen in Fig 6.

\[
F_1 = 2 \cdot \frac{1}{\frac{1}{\text{recall}}} + \frac{1}{\frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}
\]

*Fig 6: F1 score - a harmonic mean of precision and recall*

Training details
We found that training our RNN model with LSTM to be slower and less accurate. We experimented to improve accuracy and speed by just adding a convolutional layer just before our LSTM modules. This was primarily done to explore impact of high level representations of the words with LSTMs.

We trained our models in batches with a batch size of 128. We tuned our architectures over several iterations and validated using a fixed and held out validation set. We found that the optimal dropout rate is
0.2 optimized with an adam optimizer. We trained for 5 epochs (data iterations). The plots of validation accuracy and test accuracy vs epochs are provided in Fig 7 and Fig 8.

We note that while training, our CNN model did not require additional iterations to see improvements over our validation set. The test accuracy starts to fall right away after the first epoch for all three models.

![Validation accuracy vs epochs](image)

**Fig 7: Validation accuracy vs epochs**

![Test accuracy vs epochs](image)

**Fig 8: Test accuracy vs epochs**

**Baselines**

The results for abuse detection with traditional machine learning approaches are provided in Table 2. The results were quite as expected. Naive-Bayes performed poorly, this can partially be due to the incompetent conditional independence assumption of our data. Multilayer perceptron performed better than logistic regression, indicating that the features are not necessarily linearly dependent.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive-Bayes</td>
<td>0.72</td>
<td>0.92</td>
<td>0.812</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>0.74</td>
<td>0.89</td>
<td>0.813</td>
</tr>
<tr>
<td>Multilayer Perceptron</td>
<td>0.81</td>
<td>0.85</td>
<td>0.83</td>
</tr>
</tbody>
</table>

**Table 2: Baseline performance**
**Results and analysis**
For the deep learning part, CNN is a clear winner with the highest F1 score. RNN with LSTM performed poorly but gained improvement in F1 score when a convolutional layer was added before LSTM. This indicates that higher level representations of words worked well with LSTM to recapture long term dependencies instead of individual words. Hence, we conclude from our experiments that using a CNN model like ours is the best for an abuse detection system.

The final scores are provided in Table 3.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 score</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>0.87</td>
<td>0.83</td>
<td>0.852</td>
</tr>
<tr>
<td>RNN - LSTM and conv layer</td>
<td>0.89</td>
<td>0.79</td>
<td>0.840</td>
</tr>
<tr>
<td>RNN - LSTM</td>
<td>0.86</td>
<td>0.79</td>
<td>0.829</td>
</tr>
</tbody>
</table>

*Table 3: Performance of our models*

**VII. FUTURE WORK:**
In the future, we aim to explore quite a few things. Firstly exploring character embeddings instead of word embeddings. After going through the preprocessed data we found that in many places, especially in ‘abuse’ comments, words did not conform to dictionary words and mixed special characters with words. Character embeddings can better capture these. Secondly, a recently popular approach called ‘fastText’[13] which proved to be extremely effective in text classification tasks, is to be explored for our abuse detection system.

**VIII. REFERENCES:**

(https://www.lse.ac.uk/media@lse/research/EUKidsOnline/EU%20Kids%20III/Reports/NCGMUKReportfinal.pdf)

(https://www.nspcc.org.uk/services-and-resources/research-and-resources/2014/experiences-of-11-16-year-olds-on-social-networking-sites/)


