Cyber-Bullying Detection and Classification System

Marc Mailloux
marcmailloux@knights.ucf.edu

Rolando Nieves
rolando.j.nieves@knights.ucf.edu

Maxim Shelopugin
maxim.shelopugin@knights.ucf.edu

1. INTRODUCTION

Among all the forms of bullying and harassment recognized today, bullying via platforms hosted on the Internet has proliferated at an alarming rate. The rate of proliferation has been concerning enough to motivate the United States Centers for Disease Control and Prevention (CSC) to acquire and report data regarding "electronic" bullying (or, as it is more commonly known, "cyber-bullying" via its biennial Youth Risk Behavior Surveillance System (YRBSS) [**?**].

Collection of data from cyber-bullying victims has helped immensely as it pertains to awareness and prevention. It should be possible to further improve in both areas if data collection overcomes the limitation of the "a posteriori" nature of current methods (i.e., prevention methods derived from data voluntarily disclosed by victims after harassment has occurred). An automated system capable of examining electronic communication traffic, identifying and classifying any traffic that could be considered as bullying, could lead to solutions that can either filter out any offensive content automatically, or significantly shorten the response time to an incident, possibly preventing the more dire secondary effects of cyber-bullying.

The report content as follows will begin with a formal statement of the problem the authors attempt to address. Following will be a short survey of works that are related to the system documented in this paper. Then, a discussion of the system's methodology will dive into the inner workings, detailing how the system attempts to solve the aforementioned problem, as well as detailing how the system's performance will be profiled. The following sections covers briefly the risks identified and/or realized during implementation, and how, if at all, they were mitigated. After the discussion on risks, the system's performance is documented by following the system profiling plan detailed in the section covering the system's methodology. The system performance profile is then followed by a short discussion regarding insights gleaned from the system performance profile. After the system performance discussion, the work breakdown and delegation scheme applied during system implementation will be covered. Finally, the paper will conclude with short statements regarding the overall investigation, including possibilities for future enhancements to the work.

2. PROBLEM STATEMENT

The methods presented in this paper document the implementation of a proof-of-concept system that is able to accept short messages in plain text, and classify these messages, identifying those with content that could be considered as cyber-bullying. Leveraging technological advances in Natural Language Processing, along with algorithms borrowed from the areas of Artificial Intelligence and Machine Learning, the system will be trained to recognize and classify cyber-bullying traffic. The classification will go beyond simply identifying bullying traffic, but will also categorize the bullying into three (3) distinct classes:

Cultural Harassment Offensive content primarily focusing on the victim's race or religious beliefs.

Sexual Harassment Amplification of stereotypical behavior for a given gender, or gender identification.

Personal Attacks Ridicule based on a victim's outward-visible attributes, such as appearance, mannerisms, or intelligence.

3. RELATED WORK

There has been much research done around this topic, as evidenced by the body of work on display on the Cyber-Bullying Research Center research summary page [**?**]. Approaches employed to date include relatively simple Bayes type classifiers, as well as more complex deep learning systems (Sweta Agrawal et. al. [**?**]). One existing implementation that serves as an excellent exemplar to this proof-of-concept implementation is the *Anti Bully* project by Michelle Li [**?**]. The work documented in this report differs from *Anti Bully* in two important ways:

- The implementation documented in this paper does not rely solely on Naïve Bayes classification algorithms the way *Anti Bully* does.
- The system will be able to further refine the binary classification done in *Anti Bully* (which only identifies traffic as **bullying** or **not bullying**) by categorizing bullying traffic among one of the classes listed in Section 4.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

The *Anti Bully* system code base does include a good data set which can be used in the proof-of-concept documented in this paper, albeit with some modifications (see Section 6.1).

A library that has proven invaluable in the proof-of-concept implementation documented in this paper is the *Natural Language Toolkit (NLTK)* (Bird et. al. [?]). The primary *NLTK* features leveraged in this proof-of-concept include:

- Split content into sentences and words
- Filter out "Stop Words" (e.g., "the," "a,", "is")
- "Stemming" (i.e., reducing words to their root)

The tutorial by Jason Brownlee titled "How to Clean Text for Machine Learning with Python" [?] was instrumental in learning how to effectively use these *NLTK* features. Indeed, the proof-of-concept implementation documented in this paper incorporates code snippets offered in said tutorial.

Another library that has proven very useful in this proof-of-concept implementation, especially in the area of numerical processing, is the *TensorFlow* library produced by Google, Inc. [?]. The primary list of *TensorFlow* features used in this proof-of-concept are:

- Ready-made constructs for:
  - Artificial Neuron Networks (ANN)
  - Linear and Radial Basis Function-based Support Vector Machines (SVM)
- High performance parallel computing leveraging Graphical Processing Unit (GPU) capabilities

The design decision (as documented in Section 4) regarding the use of Comma Separated Value (CSV)-formatted files as primary input/output led to the inclusion of the Pandas data processing library [?].

Finally, serving as the foundation for nearly all of these external libraries, is the *NumPy* numerical computation library [?].

4. METHODOLOGY

The system, at its core, is very similar to classification systems common in the areas of Artificial Intelligence and Machine Learning. The system combines a classifying apparatus based on machine learning algorithms with a natural language vector representation mechanism.

The inner workings of the system can be described as an information flow, ultimately leading to output artifacts containing the data included in this report. The information flows through the following set of processes (some of which are further described in later subsections, as referenced in each list item):

1. **Pre-processing:** Accept the original input from the data set and prepare the data set text for feature representation (see Section 4.1).

2. **Feature Representation:** Accept the processed input and transform it into vector-based feature representations (see Section 4.3).

3. **Classification and Evaluation:** Train a classifier to predict labels for each of the samples in the data set, compare said predictions to the actual data set labels, and produce metrics describing the classification performance (see Section 4.5). Three independent classifiers are used within this task:
   - ANN classification - A three-layer (one input, one hidden, and one output) Artificial Neuron Network (ANN).
   - SVM classification - A Support Vector Machine using a Radial Basis Function (RBF) Kernel and Linear Kernel.
   - Bayes classification - A Naïve Bayes classifier.

4. **Statistical Significance:** Determine, via statistical tests, which classifier(s) perform better than others (see Section 4.10).

A simple diagram depicting the aforementioned information processing tasks, and how information flows, is shown in Figure 1.

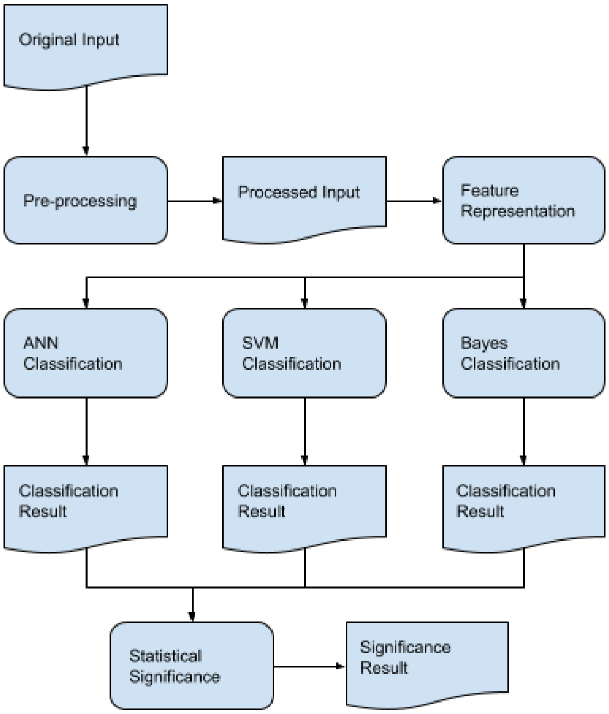

Figure 1: Pipeline depicting the high-level design of the Cyber-bullying Detection and Classification System that combines both control and data flow as the input works its way through the system, eventually leading to the outputs that help evaluate the system's performance.

In order to both save required processing time between runs, as well as capture data that eventually is included in this report, some of these processes produce artifacts as their resulting product. The complete list of artifacts in the system follows:

- **Original Input:** The data set as it exists prior to processing.
- **Processed Input:** The data set transformed by the *Pre-processing* task (see Section 4.2).

- **Classification Result:** Set of metrics that describe the classification performance of its corresponding classifier (see Section 4.9).
- **Significance Result:** Statistical analysis derived from the set of results produced by all the system's classifiers (see Section 4.11).

4.1 Pre-processing

The *Pre-processing* process takes the original data set input and performs the following tasks on it:

- Split records and fields in the Comma Separated Values (CSV) input
- Replace recognized emoji character combinations with text equivalents
- Normalize case in the text within each data set sample
- Tokenize the text into words
- Filter out punctuation in the sample text
- Filter out "stop words" in the sample text
- "Stem" the words present in the sample text

4.2 Processed Input

As output, *Pre-processing* creates a CSV file with a record for each original sample, but with the sample's text portion transformed per the aforementioned tasks.

The motivation behind having the *Pre-processing* flow process produce its output in the form of an intermediate artifact, as opposed to the more traditional in-memory data structures, can be summed up in two points:

- The time required to pre-process the data set text is not negligible, averaging around 15 minutes for the whole data set.
- Write-compile-test cycles on the downstream processes is much more efficient with the pre-processor's result readily available.
- Downstream processes are not dependent on the *Pre-processing* task's run time state.

4.3 Feature Representation

The *Feature Representation* process transforms the pre-processed sample text into vector representations by identifying the unique 3-grams and 4-grams present in the input, then describing each text sample as a vector expressing the appearance frequency of these n-grams. N-grams here are the results of process of splitting the pre-processed chunks of text into the subsets of consecutive words, where n determines the size of a given subset. The process is able to produce vector representations for each text sample based on:

- **3-gram** - The presence count of unique 3-grams in the sample
- **4-gram** - The presence count of unique 4-grams in the sample
- **3 and 4-gram** - The concatenation of the above two results

4.4 Cross-Validation

To establish the correctness of hyper-parameters of the model as well as evaluate the effectiveness of the system, it is decided to split the data set into chunks, such that every split generates two disjoint sub-data sets, which together span the original data set. Such a procedure is known as *Cross Validation*. The idea behind it is to granulate the data and train the models on one part, while testing it on the complimentary part. Repeating this procedure some k amount of times tests the robustness and generalizability of the model, while not posing any need of external data for the task.

4.5 Classification and Evaluation

As previously described, the *Classification and Evaluation* process: (1) accepts the data set after text samples have been transformed into vectors, (2) uses the data set to train a classifier such that it can predict a label for an arbitrary sample, (3) evaluates its predictions against the known sample labels, and (4) produces metrics based on said evaluation.

Although the set of classifiers used in the system is quite diverse, they all share the following design traits:

- Since the data set will not contain an explicit training and test set, the classifiers all use 5-fold cross-validation to produce classification metrics.
- The final classification metrics are averaged over each of the cross-validation runs.
- The classifiers will produce the same set of metrics (see Section 4.9).

4.6 ANN Classifier

Artificial Neuron Network (ANN) classifiers attempt to derive discrete classification labels from input data by extracting and examining latent features in the input samples. The extraction of features is done by feeding the vector feature representation of the input samples through a network of artificial "neurons" that, based on model parameters, produce an activation value based on the provided input. At a very high level, each neuron in the network is applying a linear transformation on the input, then feeding the linear transformation through an activation function that eventually produces the output. Since the input is in the form of vectors, the linear transformation must be vector-based as well, as show in Equation 1. The final neuron output is the result of this linear transformation, passed through an activation function. Equation 2 exemplifies the use of a *sigmoid* as the activation function, just like the activation function used in this proof-of-concept implementation.

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{1}$$

$$\mathbf{y} = \sigma(\mathbf{z}) \tag{2}$$

Where:

$\mathbf{z}$ - Linear transformation result

$\mathbf{W}$ - Scale factor matrix, usually referred to as weights. Part of the model parameter collection.

b - Linear bias. Part of the model parameter collection.

y - Activation result; neuron output

σ - Sigmoid activation function

The ANN classifier used in this proof-of-concept was designed to contain a total of three (3) layers: a input layer, one (1) hidden layer, and an output layer.

The input layer would be wide enough (i.e., contain enough neurons) to accept each element in the data set feature vectors. Thus, if the feature vectors can be described as $\mathbf{x} \in \mathbb{R}^n$, then the input layer contains exactly n neurons.

The hidden layer in this fully-connected ANN was designed to contain a total of 500 neurons. In effect, the input $\mathbf{x} \in \mathbb{R}^n$ that passes through this layer is reduced to $\mathbf{h} \in \mathbb{R}^{500}$. Given that the dimensionality of the feature vectors for each tweet in this proof-of-concept is a minimum of $70,000$, the hidden layer does a very effective job of extracting the pertinent information from each tweet.

The output layer in the network contains a neuron for each of the classification labels. For this proof of concept, there are four (4) possible labels, so the output layer produces values of $\mathbf{y} \in \mathbb{R}^4$. The output layer's result is then transformed using a *classification cross-entropy* function that eventually turns the output into a probability distribution, with each distribution element representing each of the classification labels. The label with the highest probability is deemed to be the network's *prediction*.

4.7 Naïve Bayes Classifier

Naive Bayes is a popular in its simplicity probabilistic classifier which is used across many fields within computer science. In its essence the model learns the conditional probabilities between the class and a classified instance, with assumed independence between the blocks of such instance. The label is assigned based on the greedy rule, such that the class which yields the highest probability is chosen as an assigned class. A more formal exposition of the concept is shown in Equation 3.

$$\begin{aligned}\hat{y} &= \arg\max_{j\in J}(P(y_j|X)) \\ &= \arg\max_{j\in J}(P(y_j)P(x_0|y_j) * P(y_j)P(x_1|y_j)\dots P(y_j)P(x_i|y_j))\end{aligned} \tag{3}$$

Where:

$\hat{y}$ - a predicted label

arg max - a function which returns the argument which yields the maximum value of the given function

X - an input vector which consists of entries $x_0 ... x_i$

J - is a set of all classes

As it is seen in the formula, the prediction of a true label is based only on the prior distribution and the conditional probability of every single element in the vector to the given label. Such simplicity attracts the attention of developers from many fields, however the assumption of independence hinders the algorithm. Although it achieves good performance in some "bag of words" tasks, it is known to under-perform in sequential data analysis since sequential data assumes dependence. In this project Gaussian version of Naïve Bayes is used, with pre-determined probabilities of priors (due to the unbalanced data set). Different architectures were tested, resulting in higher accuracy, but since the main metric to be optimized is *Recall*, Gaussian Naïve Bayes was chosen as the final model.

4.8 SVM Classifier

Support Vector Machines (SVM) are a supervised learning model that can be applied for regression or classification. The SVM constructs what is called a hyper-plane or a set of hyper-planes to separate each of the given observations. The equation for a hyper-plane for p-dimension can be seen in Equation 4.

$$\mathbf{w}\mathbf{x}_i + \mathbf{b} = 0 \tag{4}$$

Where:

w - is the normal vector to the hyper-plane

b - is some bias

$\mathbf{x}_i$ - belongs to some observation and a p-dimensional vector

---

A linear SVM goal is to classify all training data, as seen in Equation 5

$$\begin{aligned} wx_i + b &\geq 1 \; if \; y_i = 1 \; belongs \; to \; class \; 1 \\ wx_i + b &\leq -1 \; if \; y_i = -1 \; belongs \; to \; class \; 2 \\ y_i(wx_i + b) &\leq 1 \; for \; all \end{aligned} \tag{5}$$

Then to maximize the margin M of the hyper-plane as shown in Equation 6. where

$$M = \frac{2}{|w|} \tag{6}$$

Thus minimizing Equation 7 subject to the stipulated conditions.

$$\begin{aligned}\Phi(w) &= \frac{1}{2}w^t w \\ y_i &= (wx_i + b) \; for \; all \; i\end{aligned} \tag{7}$$

So we can test new classification points using Equation 8.

$$f(x) = wX + b = \sum \alpha_i y_i x_i^T x_j \tag{8}$$

Find $\alpha_1 ... \alpha_n$ such that the $Q(\alpha)$ term in Equation 9 is maximized, given the stipulated restrictions.

$$\begin{aligned} Q(\alpha) &= \sum \alpha_i - \frac{1}{2}\sum \alpha_i y_i x_i^T x_j + b \\ \sum \alpha_i y_i &= 0 \\ \alpha_i &\geq 0; for \; all \; \alpha_i \end{aligned} \tag{9}$$

SVMs are also subject to the kernel trick where we able to change the decision function and used for non-linear cases. Traditionally we have Linear, Polynomial of power p, and

Gaussian or Radial Basis Function as kernel functions. Mathematically they can be represented as shown in Equation 10 for a Linear kernel, Equation 11 for a Polynomial kernel, and Equation 12 for a Radial Basis Function kernel.

$$K(x_i, x_j) = x_i^T x_j \tag{10}$$

$$K(x_i, x_j) = (1 + x_i^T x_j)^p \tag{11}$$

$$K(x_i, x_j) = exp(-\frac{\|x_i^T x_j\|^2}{2\sigma^2}) \tag{12}$$

Thus modifying Equation 8 as seen in Equation 13.

$$f(x) = \sum_{i=0}^{n} \alpha_i y_i K(x_i, x_j) + b \tag{13}$$

Where:

$\mathbf{x}_i$ - belongs to some observation and a p-dimensional vector

σ - is a free parameter

Another notion relative to SVM multi-class classification are training the SVM one vs one and one vs rest. One vs one is will compare each class separately one by one and will create more boundaries between classes chooses the class with the most votes. This method has limitations in its computationally complexity since all classes are being compared to each other. One vs rest takes one class and compares it to the the rest of the classes chooses the strongest prediction.

4.9 Classification Result

As output, the *Classification and Evaluation* process creates the *Classification Result* artifact. This artifact is a CSV-formatted file that describes the performance of a classifier using the following metrics: (1) Precision, (2) Recall, and (3) F1 score

The results produced by the individual classifiers are stored in an external artifact in order to serve two use cases:

- Import the resulting metrics easily into this report.
- Feed the metrics into the *Statistical Significance* process.

4.10 Statistical Significance

The *Statistical Significance* process ingests the results from each of the classifiers, derives a 95% confidence interval around the *Recall* score of each performance run, and based on the confidence interval determines which (if any) classifier and feature representation combination is is "statistically better" than the others (i.e., rejecting the "null hypothesis" in the comparison). The significance test is done by ranking the classifier *Recall* scores from best to worst, then determining whether the confidence interval of the better performers sees any overlap with the intervals derived from the worse performing classifiers. Those that do not exhibit an overlap with worse performers are identified as "significantly better" than them.

Establishing the 95% confidence interval is done by assuming a binomial distribution over the scores, using that assumption to calculate an estimate for the variance of the distribution (as shown in Equation 14), and then using the appropriate *z-score* (which, for 95%, happens to be 1.96) in order to establish the lower and upper bounds of the confidence interval (as shown in Equation 15).

$$\hat{\sigma} = \sqrt{\frac{\hat{r}(1 - \hat{r})}{n}} \tag{14}$$

$$[(\hat{r} - 1.96 \times \hat{\sigma}), (\hat{r} + 1.96 \times \hat{\sigma})] \tag{15}$$

Where:

$\hat{\sigma}$ - Estimated variance

$\hat{r}$ - Estimated recall score

n - Total number of samples in the data set

4.11 Significance Result

The *Statistical Significance* process, as its output, creates the *Significance Result* artifact. This artifact is a CSV-formatted file that describes the 95% confidence interval around the *Recall* score of each classifier run. The artifact is designed for easy inclusion into this report.

5. RISK MANAGEMENT

One risk tracked by the team which was eventually realized pertained to the distribution of samples among all classes. Originally, the team identified a risk where said distribution would be too lopsided to achieve effective classification. A survey of the data set after completing the enhanced labeling is shown in Table 1.

Label	Count	Percentage
Non-Bullying	7, 203	81.7%
Cultural	260	2.9%
Sexual	264	3.0%
Personal	1, 090	12.4%
TOTAL	8, 817	100.0%

Table 1: Data set survey immediately following enhanced labeling

As evidenced by the survey, two of the labels amount to just 3.0% each of the total sample set. In order to address this extremely uneven distribution, the team has decided to duplicate the "Cultural" and "Sexual" harassment samples in the data set, such that the new make-up of the data set looks as shown in Table 2.

Label	Count	Percentage
Non-Bullying	7, 203	77.1%
Cultural	520	5.6%
Sexual	528	5.6%
Personal	1, 090	11.7%
TOTAL	9, 341	100.0%

Table 2: Data set survey after duplication of "Cultural" and "Sexual" harassment samples.

Improving data set diversity by lifting the class label composition percentage floor above 5.0% helped suppress training-induced bias that was liable to emerge in classifiers due to the original lopsided class label distribution amongst data set samples.

6. EVALUATION

The primary focus while evaluating the system's performance is to maximize detection of bullying traffic (i.e., maximize true positives while minimizing false negatives), with the minimization of otherwise innocuous traffic (i.e., maximizing true negatives while minimizing false positives).

Although several metrics were derived from the predictions produced by the classifiers, focusing on the *Recall* classification metric (i.e., the proportion of true positives classified properly) first and foremost will provide a clear assessment regarding the aforementioned behavior we seek from the system. A system that perfectly identifies all cyber bullying instances while not misclassifying any innocuous traffic would exhibit a *Recall* score of 1.0. Although such perfect performance is likely not attainable, tuning of the system will focus on maximizing the *Recall* score.

Similar classification exercises, such as that documented by Sweta Agrawal [**?**], achieved an average Recall score of 0.87. Thus, we expect this proof-of-concept to at least meet this performance baseline.

In order to objectively evaluate the performance of all classifiers used, a statistical significance test is done on the classifier's performance. Deriving a 95% confidence interval around the *Recall* score of each classifier. Classifiers with a superior *Recall* score that do not exhibit overlap with any worse performing classifier's confidence interval are deemed as "significantly better."

6.1 Data Set

The data set used in this proof-of-concept, as alluded to in Section 3, is a modified version of the data set provided by the *Anti Bully* system [**?**]. The data set as provided is labeled, but the labels only contain two (2) classes: **bullying** and **not bullying**. In order to meet the goals for the system as detailed in Section 1, the data set labeling has been enhanced such that the samples labeled as **bullying** are distributed among the categories identified in Section 4.

6.2 ANN Classification Results

The performance of the ANN classifier on the Data Set was evaluated by comparing the network's predictions against the true labels in each test set used. Given the cross-validation approach applied to the data and the classifiers, as detailed in Section 4.5, each cross-validation run produced its own set of metrics. For the purposes of evaluating the classifier, the set of metrics produced in each cross-validation fold were averaged granting equal weight to each fold. Within each fold, however, the metrics were weighed according to the class composition of the cross-validation fold samples. Separate metrics were included, however, based on what feature representation was fed to the ANN classifier. Details on the different feature representations are documented in Section 4.3, and they are identified in the results by the labels that appear in boldface prior to each representation's brief explanation. The results of the ANN classification run are shown in Table 3.

Representation	Precision	Recall	F1
3-gram	0.840	0.868	0.829
4-gram	0.849	0.876	0.838
3 and 4-gram	0.855	0.882	0.852

Table 3: The overall weighted average of the performance metrics derived from ANN classification runs on each feature representation of the data. Average weighing was done by scaling the raw results using the input data set sample presence count.

6.3 Naïve Bayes Classification Results

The performance of Naïve Bayes is based on the ability of the algorithm to distinguish between the classes of bullying given the training and testing Data Sets. Mainly the focus is to derive some probabilistic characteristics that depict the distribution of samples in Training Data Set which can be generalized to some unknown data - Testing Data Set. Model was run in accordance to the Section 4.5, and the results were averaged over all cross-validation fold samples. The performance of Naïve Bayes can be seen in the Table 4.

Representation	Precision	Recall	F1
3-gram	0.434	0.697	0.463
4-gram	0.426	0.691	0.449
3 and 4-gram	0.434	0.697	0.464

Table 4: The overall weighted average of the performance metrics derived from Naïve Bayes classification runs on each feature representation of the data.

6.4 SVM Classification Results

Using three different SVM model representations as classifiers on the given Data Set, the model was evaluated by comparing the calculated placement of the support vector or prediction vs its actual local or the correct class label. The evaluation process consisted of using a 5-fold cross-validation technique, where metrics were produced for each fold for each model used. The main difference from each model was the kernel that was used, and the notion one vs one or one vs rest. The three kernels used were Gaussian (Radial Basis Function), Linear One Vs One, Linear One Vs Rest. Details on the different feature representations are documented in Section 4.3, and they are identified in the results by the labels that appear in boldface prior to each brief explanation The results of all three models are seen in Table 5.

7. DISCUSSION

7.1 ANN Classification

The ANN classifier performed about as well as was originally expected, as discussed in Section 6, particularly when discussing the prior work done by Agrawal [**?**], achieving in most cases a rounded *Recall* score of 0.87 or better. A deeper dive into the per-class metrics, however, does reveal a curious detail regarding the classifier's performance. For example, Table 6 displays the per-class metrics acquired from the "**3 and 4-gram**" feature representation run, which at first glance appears to be the best performer.

Configuration	Precision	Recall	F1
RBF OVO(3-gram)	0.59	0.77	0.67
Linear OVO(3-gram)	0.854	0.884	0.854
Linear OVR(3-gram)	0.854	0.882	0.854
RBF OVO(4-gram)	0.59	0.77	0.67
Linear OVO(4-gram)	0.86	0.886	0.852
Linear OVR(4-gram)	0.858	0.882	0.852
RBF OVO(3 and 4-gram)	0.59	0.77	0.67
Linear OVO(3 and 4-gram)	0.858	0.884	0.852
Linear OVR(3 and 4-gram)	0.854	0.882	0.852

Table 5: The overall weighted average of the performance metrics derived from SVM classification runs on each feature representation of the data. Among the configurations, "RBF" stands for "Radial Basis Function," "OVO" stands for "One vs One," and "OVR" stands for "One vs Rest."

Class Label	Count	Precision	Recall	F1
No Bullying	1,440	0.880	0.993	0.933
Sexual	105	0.981	0.981	0.981
Cultural	104	1.0	0.942	0.970
Personal	218	0.769	0.138	0.233

Table 6: ANN classification performance metrics, broken down per class label, on the last cross-validation fold of the "3 and 4-gram" feature representation of the data set.

The most surprising fact that emerges from these metrics is not only how well the *No Bullying*, *Sexual Harassment* and *Cultural Harassment* classification performed, but also how poorly the *Personal Attacks* classification performed. The overall metrics were primarily buoyed by the strong performance of the *No Bullying* classification, given that the overall metrics were weighed based on data set composition. A possible explanation for this discrepancy could be that the content in tweets labeled as *Sexual Harassment* and *Cultural Harassment* was very consistent, easily trainable. Whereas the *Personal Attacks* tweets were too similar to content labeled as *No Bullying*, making it difficult to positively identify. This last statement is supported by the fact that the confusion matrix for the run profiled in Table 6 showed that nearly all samples whose ground truth label was *Personal Attacks* produced a predicted label of *No Bullying*.

It should be noted that, although Table 6 is only showing the metrics from one of the "**3 and 4-gram**" classification runs, the other folds exhibited the same curious disparity within the classifier's performance.

7.2 Naïve Bayes Classification

Being one of the simplest algorithms for the classification purposes, Naïve Bayes did not perform well on such a diverse and unbalanced data set. As it was noted in Section 7.1, the classifier finds it hard to differentiate between *No Bullying* and *Personal Attacks* classes, while perfectly labeling *Sexual Harassment* and *Cultural Harassment*. This might be because the *Personal Attacks* section is too broad and contain many different embeddings which overlap with *No Bullying*, while other sections are more refined and have specific n-grams which trigger the classifier. The confusion matrix for **3-gram** representation is shown in Table 7.

Class Label	0	1	2	3
0	740	120	305	275
1	0	104	0	0
2	0	0	105	0
3	90	20	53	55

Table 7: Naïve Bayes Confusion Matrix for "3 -gram" representation, averaged over 5-fold cross-validation. The class labels were replaced by their numeric counterparts due to space considerations. Label (0) is "No Bullying, " label (1) is "Sexual Harassment," label (2) is "Cultural Harassment," and label (3) is "Personal Attacks."

Combination	Low	High	Actual
ANN 3-gram	0.861	0.875	0.868
ANN 4-gram	0.870	0.883	0.876
ANN 3 and 4-gram	0.876	0.889	0.882
Naïve Bayes 3-gram	0.688	0.707	0.697
Naïve Bayes 4-gram	0.681	0.700	0.691
Naïve Bayes 3 and 4-gram	0.688	0.707	0.697
SVM RBF OVO(3-gram)	0.761	0.779	0.77
SVM Linear OVO(3-gram)	0.878	0.890	0.884
SVM Linear OVR(3-gram)	0.875	0.888	0.882
SVM RBF OVO(4-gram)	0.761	0.779	0.77
SVM Linear OVO(4-gram)	0.880	0.892	0.886
SVM Linear OVR(4-gram)	0.875	0.889	0.882
SVM RBF OVO(3 and 4-gram)	0.761	0.779	0.77
SVM Linear OVO(3 and 4-gram)	0.878	0.890	0.884
SVM Linear OVR(3 and 4-gram)	0.875	0.889	0.882

Table 8: 95% confidence interval on the Recall metric for all classifier and feature representation combinations implemented in this system. Among the SVM classifiers, the designation "OVO" stands for "One vs One," whereas "OVR" stands for "One vs the Rest."

7.3 SVM Classification

The SVM models depict a trend known when using SVM, where the decision boundary is non-linear or linear. Looking at the results it is clear that the decision boundary is leaning on the linear side as the RBF kernel model only achieved a precision of 59 percent. The same model had a recall was 0.77, which is below the goal of 0.87 achieved by Agrawal [?]. Of the two linear models the main difference in performance was not by much, yet the one vs one approach had the higher precision, and recall values. This is due to the one vs one model having more support vectors to compare against, at the cost of computation. While the one vs rest was not far behind this model was certainly the most streamlines in terms of time since it has far less support vectors to compare. Both linear models slightly surpassed the expected results again achieved by Argrawal [?]. Due to two of the three SVM models were using one vs one the calculation time was immense as it took over a week to calculate.

7.4 Statistical Significance Test

The results of the statistical significance testing done on the combinations of classifier and feature representation are shown in Table 8. A high/low chart depiction of the numerical data is shown in Figure 2.

The statistical significance testing yields several insights

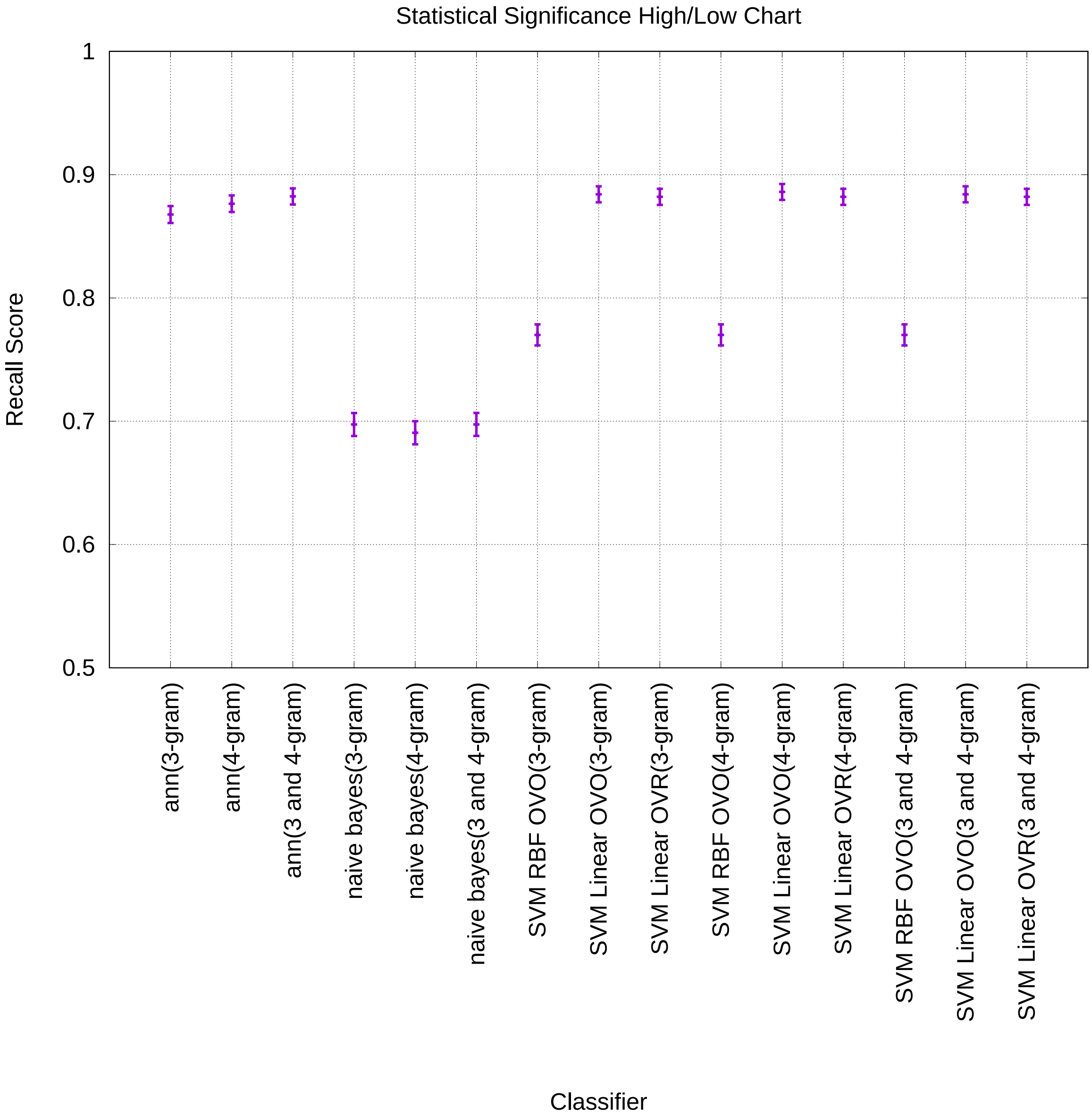

Figure 2: High/Low chart depicting the 95% confidence interval for each classifier and feature representation combination implemented in this system.

into the performance of the system. The clearest takeaway from this analysis is that all classification runs that used an ANN classifier are significantly better than all classification runs that used a Naïve Bayes classifier. The *Recall* score separation between the 95% interval lower bound of the worst ANN performer and the upper bound of the best Naïve Bayes performer is 0.154, whereas the separation between the upper bound of the worst ANN performer and the lower bound of the best ANN performer is merely 0.001.

Among the ANN classifiers, only the comparison between the "ANN 3-gram" and "ANN 3 and 4-gram" classification runs rejects the statistical null hypothesis $H_0 : \hat{r}_0 = \hat{r}_1$ since the upper bound of the 95% confidence interval of "ANN 3-gram" (0.875) is below the lower bound of the confidence interval of "ANN 3 and 4-gram" (0.876), thus accepting the statistical alternative hypothesis $H_a : \hat{r}_0 \neq \hat{r}_1$.

Comparing the ANN classifier with the variety of SVM classifier configurations yields more interesting results, however. Several of the SVM variations either matched or slightly out-performed the best ANN classifier. It is also interesting to see that SVM classifiers using a Radial Basis Function (RBF) kernel actually performed significantly worse when compared against their Linear kernel counterparts. Another interesting insight into the SVM classification runs is that feature representation did not have as profound an impact as it did with either ANN or Naïve Bayes. Based on the 95% confidence intervals, there is no significant difference between any of the SVM Linear kernel classifiers and the best non-SVM classifier (ANN using "3 and 4-gram" feature representation).

8. WORK DISTRIBUTION

The proof-of-concept implementation exercise is divided into seven (7) primary task areas. The order in which the tasks are presented in this paper does not necessarily represent the chronological order in which the tasks were be carried out.

8.1 Data Set Labeling

As stated in Section 6.1, the source data set for this proof-of-concept needed to be refined in order to accommodate the classification taxonomy as described in Section 4. Although all team members contributed to this task, **Mr. Marc Mailloux** serves as the task's point of contact.

8.2 Text Tokenization

Transforming the training and test sets from plain English into a form that ignores small derivations from the contextual meaning can improve the accuracy of the model. For this task, **Mr. Marc Mailloux** serves as both the primary developer and the point of contact.

8.3 Text Embedding

Transforming the text representation from plain English text to vector embeddings which will be fed into the system. For this task, **Mr. Maxim Shelopugin** provides the main methodology and **Mr. Rolando Nieves** increases the performance for the multi-threaded environment.

8.4 Cross-Validation

Separating the data set into training and testing, as well as providing minor optimizations like double-counting is done in a separate module, and the point of contact for it is **Mr. Maxim Shelopugin**.

8.5 Classifiers

The classifier task, given its wide breadth, is divided among all team members. A list of the classifiers that will be used, along with a short explanation justifying their use, as well as the classifier's implementation point of contact, follows:

- Artificial Neuron Network (ANN) - ANNs possibly represent the most flexible of classifiers, both in the form of input they accept as well as the output they produce. The point of contact for this classifier is **Mr. Rolando Nieves**.

- Support Vector Machines (SVM) - SVMs have the possibility of requiring the least amount of computing power during both training (as compared with ANNs) and classification. The point of contact for this classifier will be **Mr. Marc Mailloux**.

- Naïve Bayes - This was the classifier used in the *Anti-Bully* system this proof-of-concept will be based on. It will be interesting to assess the impact, if any, of implementing the same classifier on a multi-class environment. The point of contact for this classifier will be **Mr. Maxim Shelopugin**.

The statistical significance testing, as described in Section 6, will be implemented by **Mr. Rolando Nieves**. Mr. Nieves will also serve as the point of contact for the classifier task.

8.6 Presentation

Mr. Maxim Shelopugin will be responsible for producing any materials required to present the results of this proof-of-concept to interested parties, including a summary poster.

8.7 Final Report

Although all team members will contribute content to it, **Mr. Rolando Nieves** will be responsible for producing the final report that will document the results observed while exercising the proof-of-concept implementation documented in this paper.

9. CONCLUSION

The proof-of-concept system documented in this report has proven to be a solid enhancement of the efforts started by Michelle Li [**?**]. The system provides good evidence that well-established machine learning techniques can serve the greater good in helping address and possibly curb society's darker impulses.

In current society, however, fielding such a system would levy more than just technological challenges. Privacy concerns are a very important consideration when putting the technology demonstrated in this report to practice. In the opinion of the authors, some deployment choices could steer clear from such privacy concerns, such as giving service users the option to deploy and activate the system's features on their end, vs. forcing the deployment across the entire service's installation base.

There are still areas where a system like the one documented in this report could be improved upon. First among them is refining the algorithms such that anomalies as those documented in Section 7.1 are better mitigated. Agrawal et. al. [**?**] addresses one of the enhancement suggestions, which is the portability of a trained system across multiple communication platforms. Another opportunity for improvement lies in enhancing the system such that either unsupervised learning or Reinforcement Learning can replace the use of labeled data sets in what has been up to this point a supervised learning system. Unsupervised learning would require engineering the feature representation of text units such that patterns emerge "organically." Reinforcement learning would require an automated reward system that encourages the proper labeling of text units. Both of those endeavors have the clear potential of proving very rewarding to any interested researchers.