X REAL-TIME BAD BEHAVIOR DETECTOR

A Project

Presented to the faculty of the Department of Computer Science

California State University, Sacramento

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

by

Raul Arambula

SPRING
2024

X REAL-TIME BAD BEHAVIOR DETECTOR

A Project

by

Raul Arambula

Approved by:

_____, Committee Chair
Dr. Anna Baynes

_____, Second Reader
Dr. Parham Phoulady

_____
Date

Student: Raul Arambula

I certify that this student has met the requirements for format contained in the University format manual, and this project is suitable for electronic submission to the library and credit is to be awarded for the project.

_____, Graduate Coordinator _____
Dr. Ying Jin Date

Department of Computer Science

Abstract

of

X REAL-TIME BAD BEHAVIOR DETECTOR

by

Raul Arambula

Social media and the internet have connected the world like never before. This connectivity, while hugely beneficial, has some stark negatives. One such negative is the advent of cyberbullying. Cyberbullying takes place on many social media platforms, such as X, formerly known as Twitter. X and many other social media platforms do their best to moderate their platforms but there is room for improvement. This project proposes a novel form of real-time bad "behavior" detector that ideally could be utilized to help moderate and create a friendlier environment on X. As opposed to most conventional batch-based models utilized in content moderation this model is streaming-based, which should allow it to better adapt to changes in cyberbullying techniques. This model is an ensemble-based voting classifier that utilizes an X post's characteristics as well as natural language encoding of the raw text of the post. The project utilizes X posts collected and labeled from prior research.

The proposed ensemble-based voting classifier was able to perform competitively with traditional batched-based classification models and other streaming-based classification models. The

proposed model's accuracy, f1, recall, and precision scores were in line with or outperformed the scores of other approaches. These results present some validity to the application of the model in real-world social media platforms.

_____, Committee Chair
Dr. Anna Baynes

_____
Date

ACKNOWLEDGEMENTS

I would like to thank the Department of Computer Science faculty and staff for their shared knowledge, support, and guidance. In particular, I would like to show my sincerest gratitude to my primary advisor, Dr. Anna Baynes, for their mentorship, expertise, and encouragement. Additionally, I want to thank Dr. Parham Phoulady for their time, feedback, and insights as my second reader. Both of their guidance was instrumental in the completion of this work.

Most importantly I would like to thank my family. My academic journey would not have been possible without their love, sacrifices, time, and support. I am beyond grateful for all of them.

TABLE OF CONTENTS

Page

LIST OF TABLES

Tables Page

LIST OF FIGURES

Figures Page

1. INTRODUCTION

Before the widespread use of the internet and social media bullies or other aggressors had to attack their victims in person [2]. It is far easier for bullies and aggressors to harass their targets constantly and possibly anonymously with the advent of social media [2]. One such possible place for these transgressions to take place is X, formerly known as Twitter. While social media applications attempt to discourage or prevent these occurrences, there is evident room for improvement [2]. I propose a novel form of real-time bad "behavior" detector to facilitate the moderation of X. The proposed model is a streaming ensemble voting classifier that utilizes an X post's characteristics along with the natural language encoding of the raw text.

The goal of this project is to create a real-time "bad behavior" detector for X that focuses on posts/tweets that are abusive, hateful, and normal. It is a real-time model because it is an online streaming machine learning model as opposed to most current models that are offline batch machine learning models [3]. This model expands on the work done by Herodotou's team by expanding the scope of the model from just utilizing a post's characteristics to also utilizing a post's raw text and using different classification models [4]. The language model encodings of a post's raw text were applied to facilitate better classification performance. A real-time streaming model was chosen because it should be more sensitive to fast-evolving trends than an offline batch model.

2. BACKGROUND

2.1 X/Twitter

X or Twitter has been the focus of much research and many experiments. There has been a wide range of areas explored. For instance, a lot of work has been conducted on reducing spam on X [5], [6], [7], [8]. It is very common to scroll through the homepage of X or scroll through the comments of a post and see many fake or bot accounts posting spam. This problem is not unique to X either, many of the other social media platforms still struggle with this. Some of the recent approaches focus on streaming learning models due to the ever-evolving and changing tactics of spammers [5], [6], [7], [8]. Streaming learners will be more sensitive to the shifts in concepts of these data streams. Another topic that has received a lot of attention is X posts sentiment analysis. The goal of sentiment analysis is to classify text as either being positive, negative, or neutral. This can be useful in multiple situations such as, marketing teams trying to gauge how well received a product was, film studios wanting to know the reception of their movies, and political campaign managers needing to know the general reactions to their candidate's speech. Similarly to spam some of the approaches are streaming models for many of the same reasons as spam, sentiment changes [9], [10], [11]. Sentiment analysis was one of the attributes attained from feature extraction and used as a feature for classification in this research.

Previous work has been conducted in the same area as the research presented here, namely experimentation and exploration in the classification and characterization of posts on X that exhibit bad behavior. The dataset utilized in this work is one such instance. The researchers conducted a large-scale crowdsourcing to label X posts. They collected 10 days of posts from X amounting to around 32 million posts [1]. After conducting metadata extraction, filtering and boosted sampling they settled on around 100k posts [1]. They initially started the experimental rounds with the labels offensive, abusive, hateful, aggressive, cyberbullying, spam, and normal

[1]. After a few rounds and some label concatenation, the final results were the labels abusive, hateful, spam, and normal [1]. Around 11% are abusive, 7.5% are hateful, 22.5% are spam, and 59% are normal [1]. Other work has gone beyond just classifying individual posts but users as being bad actors. This work categorized users as being bullies, aggressors, spammers, or normal [2]. They utilized characteristics and features beyond just the content a user posted, they also analyzed user account information and social networks [2]. For instance, they looked at what lists a user subscribed to, how old an account is, how many followers a user has, how popular a user is, and what a user's community looks like [2]. Some of the types of machine learning algorithms they used were tree-based, ensemble, or probabilistic [2]. All of these were batch-based, and none utilized streaming. Out of all the models tested Random Forest was their best. Their experiments also ran with two different cases. The first case considered all four labels. The second case did not include spammers. They utilized some different metrics for performance evaluation such as ROC curve, kappa, and RMSE [2]. Their Random Forest classifier achieved average precision and recall scores of around .72 for the four-class case and of around .90 for the three-class case [2]. Prior research has been conducted utilizing streaming models. One approach only considered the labels aggressive and normal for classifying posts [3]. They also utilized profile and network features along with text features to predict labels [3]. The models they tested were Hoeffding Tree, Adaptive Random Forest, and Streaming Logistic Regression [3]. The evaluation metrics used were comparable to the approaches utilized in this research, accuracy, precision, recall, and f1-score. In most metrics, their models performed similarly to or slightly less than the batch-based models [3]. Only did the recall score on some models better the batch-based approach [3]. Another experiment was conducted but the classes they were focused on were abusive, hateful, and normal [4]. They also did consider the same case as before by combining abusive and hateful into aggressive. Their approach used many of the same features and models for classification.

However, they did implement an adaptive bag-of-words method for swear words, but it was only used in the 2-label case of aggressive and normal [4]. Their best streaming model, Hoeffding Tree, was able to achieve f1-scores close to or slightly better than the batch-based approach [4].

2.2 Other Streaming Models

A lot of research has been conducted to find different approaches to improve streaming learning algorithms. One subset of approaches focuses on improving or extending upon Random Forest (RF) or Adaptive Random Forest (ARF). Another set of approaches concentrates on Hoeffding Trees. There are multitudes of approaches not covered here, however, these are the categories that are most relevant or related to this research's approach.

2.2.1 Random Forest or Adaptive Random Forest

ARF is the application of the popular RF algorithm for offline batch learning but for online streaming data. Therefore, it is a type of streaming ensemble learning algorithm. This means that multiple base learners are used for classification. One way classification labels are assigned by an ensemble algorithm is through voting from all the base trees. There are many ways voting can be implemented such as majority, weighted majority, relational, classifier selection, and rank [12]. As an example, in rank voting all the base learners can output a list of class labels ranking their probability and then the label with the highest overall summed rank is the one predicted by the ensemble algorithm as a whole [12]. The three main characteristics of ARF are that it includes diversity through resampling or bagging, subsets of features for node splits are selected randomly, and each base tree has its own drift detector [13]. When a drift warning is given ARF begins training a tree in the background and if the warning escalates to a drift the drifting tree is replaced with the new tree that was training in the background [13]. A recent approach Adaptive Random Forest with Resampling aims to improve upon traditional ARF with a new resampling approach that improves classification performance for the minority class, especially for imbalanced data

streams [14]. It also improved execution time compared to ARF [14]. There were instances in experimentation where the majority class performance deteriorated slightly [14]. Another new method proposed, CS-ARF, is intended to reduce the high resource cost of ARF [15]. This was achieved through Compression Sensing to compress data while still maintaining the most important attributes [15]. The compression of data would lessen the resource cost of ARF. CS-ARF did not outperform ARF in most datasets but it was competitive with it and did outperform some other streaming classification algorithms [15]. An alternative to ARF that was proposed as a streaming form of RF was Streaming Random Forests. Streaming Random Forests creates multiple decision trees that learn incrementally [16]. It uses node windows and tree windows to decide when to prune, transform frontier nodes to internal nodes or leaf nodes, and create new trees [16]. The Streaming Random Forests algorithm performs similarly in classification abilities to traditional RF [16].

2.2.2 Hoeffding Tree

A Hoeffding Tree is an incremental decision tree. It uses the Hoeffding bound to decide when to split a node on the best attribute [17]. It is great in a streaming setting because the Hoeffding bound states that a certain amount of samples is sufficient enough to choose the optimal attribute, and that attribute would be the same one chosen if an infinite number of samples were seen [17]. An extension of Hoeffding Tree that was proposed was the Hoeffding Tree with nmin adaptation. Nmin is the number of instances needed to be seen to make a node split. The goal of nmin adaptation is that it would reduce the number of instances needed to be seen to split the node [17]. This would reduce computation costs while ideally having little impact on classification performance. Nmin adaptation works by applying a unique nmin value to each tree node instead of having one unified nmin value and the nmin value chosen for each tree node is based on incoming data to ensure that a split occurs within a certain confidence [17]. An alternative to

Hoeffding Adaptive Tree that was proposed recently was the Extremely Fast Hoeffding Adaptive Tree. This approach combines the Hoeffding Adaptive Tree with the Extremely Fast Decision Tree [18]. This is accomplished by implementing the eager splitting strategy of the Extremely Fast Decision Tree and the drift detection and response approach of the Hoeffding Adaptive Tree [18]. The result is an algorithm that reacts to concept drift faster and more effectively than the two models it is a combination of [18]. It also handles streams of data with little variation or change [18]. One new approach to Hoeffding Trees sought to increase its throughput of data streams. The Vertical Hoeffding Tree is a distributed learning algorithm that uses vertical parallelism to increase scalability and throughput [19]. The motivation for this improvement was the notion that data is typically already distributed so therefore there is no need for it to be constrained by one machine. Vertical parallelism involves slicing the data matrix by column, making attributes independent of each other, and counters for a single attribute are grouped on one node in the tree [19]. This makes memory requirements comparable to a standard decision tree whereas with horizontal parallelism memory would increase linearly with parallelism [19]. Also, by considering the attributes independent of each other the split criterion can be calculated in parallel on multiple nodes at once [19]. The approach was able to display a higher throughput of data with only minor degradation in accuracy.

2.3 Other Language Models

Large language models have been a popular topic as of late. Many huge innovations and discoveries have taken place lately. One recent development was the creation of GPT-4. It is a multimodal large language model created by OpenAI [20]. It is a transformer-based approach and can take in text or image inputs and produce text outputs. GPT-4 was pre-trained to predict the next token in a text using data collected from scraping the internet and data that was provided to OpenAI from third parties [20]. It was later fine-tuned with reinforcement learning feedback from

humans to comply with OpenAI's and other policies on artificial intelligence [20]. Due to the proprietary nature of the model, many of the specific details of its architecture, hardware, training, and other aspects have been kept secret. It is an improvement from previous versions but still retains the limitations of its predecessor such as hallucination, biases, and an inability to coherently explain its logic [21]. Another big latest development was the release of Llama 2. Llama 2 was developed by GenAI and Meta. Llama 2 was released with different model sizes ranging from 7 billion to 70 billion parameters [22]. It was trained on 2 trillion tokens and has a context length of 4k [22]. It was also later fine-tuned with human reinforcement learning feedback for safety and compliance [22]. More of the specifics of the architecture, hardware, and training were provided by GenAI and Meta. The basis of the language model used in this research is a bit older but still viable model. BERT is a transformer-based language model that can only encode developed by Google [23]. Therefore, unlike the previous models, it cannot be queried by text and output text. BERT consists of three modules embedding, encoders, and un-embedding [23]. The embedding layer turns arrays of one-hot encoded tokens into an array of vectors that represent the tokens [23]. The encoders transform the representational vectors and then the un-embedding component turns the representational vectors back into one-hot encoded tokens [23]. A version of BERT was implemented in this work.

3. SYSTEM ARCHITECTURE

The primary overall goal of the model is to classify X posts as either abusive, hateful, or normal, the three-class case. The model was also tested with abusive and hateful combined for a new label aggressive. The goal of that secondary objective was to test its classification performance for the labels aggressive and normal, the two-class case. Figure 1 demonstrates all the components and steps of the overall system architecture [24].

3.1 Preprocessing/Feature Extraction/Normalization

The model begins with reading a stream of X posts. When the model receives a post it begins preprocessing. During preprocessing the raw text of the post is cleaned of unnecessary punctuation, special characters, URLs, hashtags, user mentions, abbreviations (e.g., RT), and any other non-alphabetical characters. After posts have been cleaned of unnecessary information feature extraction begins. More detail is given in Chapter 4.

The final step before the data is fed into the models is normalization. Non-categorical numerical features were normalized using a z-score. Z-score is $z = (x - \mu)/\sigma$. X is the numerical value of the feature, μ is the mean, and σ is the standard deviation. The value of z is how many standard deviations that feature is away from the mean. Scores above the mean are positive while scores below the mean are negative. Categorical features were one-hot encoded. The class label column was label encoded with abusive being 0, hateful being 1, and normal being 2 in the three-class case. In the two-class case aggressive was 0 and normal 1.

3.2 Language Model/Training

The pre-trained language model (LM) used was msmarco-distilbert-dot-v5 [25]. The LM was created for semantic search. It has a 768-dimensional dense vector space and was originally trained on 500k query, answer pairs from the MS MARCO dataset [26]. This LM was chosen due to its performance on this dataset compared to other LMs in the SentenceTranformers Python

framework. Further training was done to help boost performance and fine-tune it for this dataset. Two versions of the LM were created and evaluated, one was trained on five days of data and the other on one day of data. After the language model is trained the raw text of the posts is encoded creating a text stream. This results in two data streams. One is the text stream containing the language model encodings of the raw text and the other is the characteristics stream containing the features extracted and normalized from the posts. The two data streams are sent to their respective models. The streaming models make predictions and are trained on each post once as it is read in and then that post is discarded. This form of training is what allows streaming models to remain sensitive to changes, thus allowing them to be always up-to-date. The models also will not need to be routinely manually retrained like batch models.

3.3 Prediction/Evaluation

The text stream model will give its probability distribution for each label of the given post and the characteristic model will do the same. These two probability distributions are then equally weighted and combined to find the overall predicted label for the post. The assigned label is then evaluated. After a prediction is made that prediction is used to partially train the models.

The evaluations performed on these models are f1-score, accuracy, precision, recall, and confusion matrix. These metrics aid in understanding and evaluating the performance of the models. Some interesting statistics and graphs can be generated from these metrics to discover what labels were hard to classify as well as how these models performed over time.

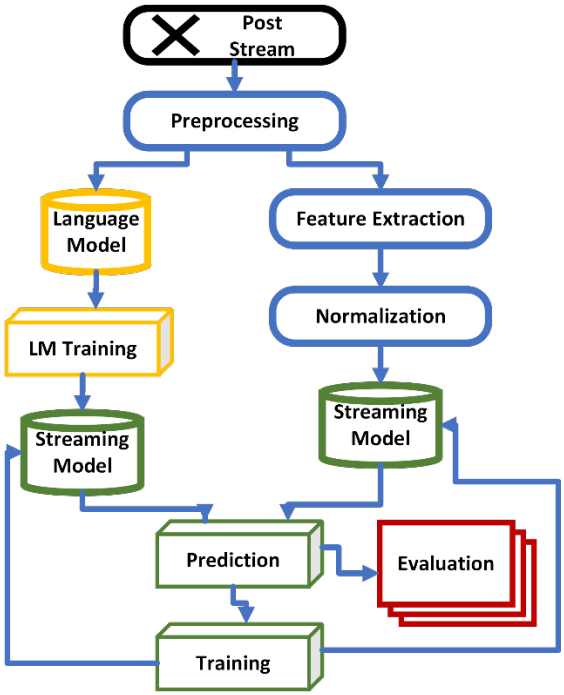

Figure 1: Complete System Architecture for Classifying X Posts

3.4 Types of Streaming Models

Many streaming algorithms exist, all with their strengths and weaknesses. Some are designed to better handle concept drift, others value computer resource efficiency, and others throughput of data. The streaming models utilized here were implemented from scikit-multiflow [27]. Scikit-multiflow is a Python machine-learning package for streaming data. The streaming model used for the text stream was naïve Bayes, and the three models used for the characteristic stream were Extremely Fast Decision Tree (EFDT), Hoeffding Adaptive Tree (HAT), and Streaming Random Patches (SRP).

3.4.1 Naïve Bayes

Naïve Bayes is a very simple classifier that assumes that attributes are conditionally independent of each other and the target label [28]. Bayes's Theorem is then used to compute the probability

of each target label. Even though naïve Bayes is a fairly straightforward classifier it is highly effective. Its application in this approach, also better highlights the impact of the language models on performance.

3.4.2 Extremely Fast Decision Tree

The Extremely Fast Decision Tree is a form of Hoeffding Tree. The main difference from the EFDT compared to a conventional Hoeffding Tree is that when creating the tree it will split at a node as soon as it is deemed useful [29]. It will not try to find the "best" split. However, if that decision is later evident to have been the wrong one it will revisit the split and replace it. EFDT learns faster than Hoeffding Tree but has a higher computational cost [29].

3.4.3 Hoeffding Adaptive Tree

The Hoeffding Adaptive Tree is also a form of Hoeffding Tree. HAT expands on the Hoeffding Tree by utilizing ADWIN. ADWIN is a change detector and error estimator that uses a self-adjusting window to help detect concept drift [30]. Concept drift is when the characteristics of the target variable change in unexpected ways. This will cause models to become less accurate. Drift detection and adaption are crucial for data that is ever-changing and evolving, such as social media trends. ADWIN will track the performance of branches and will replace them with new branches if the new ones outperform the old ones [31].

3.4.4 Streaming Random Patches

Streaming Random Patches is a streaming ensemble learner. This means that it combines a few base learners' predictions to increase overall predictive performance. The base learner for SRP is the Hoeffding Tree [32]. What sets SRP apart from other ensemble learners is the fact that it applies online bagging and random subspaces [32]. Bagging involves bootstrap sampling to diversify samples, then the bootstraps are trained independently in parallel and finally combined by average or voting. Random subspaces involves training the base learners with randomly

sampled features with replacement, this helps learners not over-emphasize certain features that appear highly predictive. SRP also uses ADWIN as an active drift detection strategy [32].

4. DATASETS AND FEATURE EXTRACTION

4.1 Dataset

The dataset utilized classifies X posts as either abusive, hateful, spam, or normal. The authors conducted multiple rounds of crowdsourcing annotation, with many different labels such as offensive, abusive, hateful, aggressive, cyberbullying, spam, and normal [1]. As the rounds went on highly correlated labels were merged and others were dropped [1]. At the end of their work, they concluded on 100k posts with the four labels mentioned earlier. Around 14k posts with the spam label were dropped due to it not being the main focus of this research and there being more specialized cases to solve that problem [5], [6], [7], [8].

4.2 Feature Extraction

There are multitudes of attributes that could be used to describe an individual's online presence. A few such attributes are the number of friends a user has, how many posts a user has made, how old a user's account is, etc. X's new policies prevent the open dissemination of a post's text with their unique post ID. This is meant to prevent the ability of being able to identify posts to their user. Therefore, the only way to retrieve further user attributes now requires registration as a developer and payment to access their API. This was deemed out of the scope of the project. Due to these circumstances, the focus of this research was solely on the post's text features. No user attributes such as the number of followers, number of likes, and number of posts, were utilized.

4.2.1 Basic Content

A basic content analysis was conducted on the text first. This includes text features such as the number of at signs, hashtags, URLs, and whether a post was a repost or not. The distribution of these characteristics was fairly similar. The only real discrepancies came with whether a post was a repost or not. Only around 3% of normal posts were reposts, about 50% of abusive posts were reposts, and around 36% of hateful posts were reposts. These numbers suggest that when a user is

attacking another user they are more likely to repost the user they are attacking. This likely means the aggressor is attacking the other user based on their post because the at sign could also be used to direct an attack at a user however the distribution of at signs seems insignificant. After analysis, these features were removed from the text to clean it for later cleaning and processing. Other features removed after feature extraction and analysis were punctuation, numbers, and any remaining non-English letter characters. Any posts with no text content after data cleaning and processing were dropped.

4.2.2 Syntax

The syntax of the text was evaluated next. The NLTK part of speech tagger was used to count the number of adjectives, verbs, and adverbs that were seen among the labels [33]. The distribution for adverb and adjective counts was roughly similar among all three labels. As Figure 2 shows the verb distribution for the hateful label did seem to exhibit more of a bell curve shape than the other labels. The other labels seemed to skew a bit left. Normal and hateful had a similar median value but both the normal and abusive labels had interquartile ranges that skewed left. This indicates that hateful tweets more consistently contain verbs and therefore are using those verbs as part of their attacks on other users. Most likely a negative action is being directed at users being attacked.

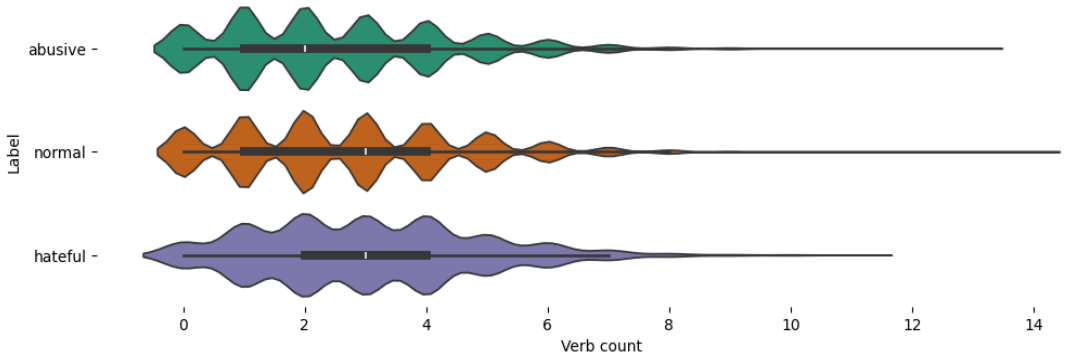

Figure 2: Verb Count Violin Plot

4.2.3 Style

The style of the text was the following feature analyzed. The characteristics monitored were the number of uppercase words used, the number of sentences used, the average sentence length, and the average word length. For the stylistic characteristics the number of uppercase words used, the average sentence length, and average word length distributions showed nothing of real interest. The number of sentences used did show an interesting distinction between the labels. In Figure 3 abusive and hateful posts showed a much higher preference for using fewer sentences. This data seems to indicate that those users attacking others much rather get to their point quicker. There is no need to write extensively when attacking another user. An effective attack does not require many sentences.

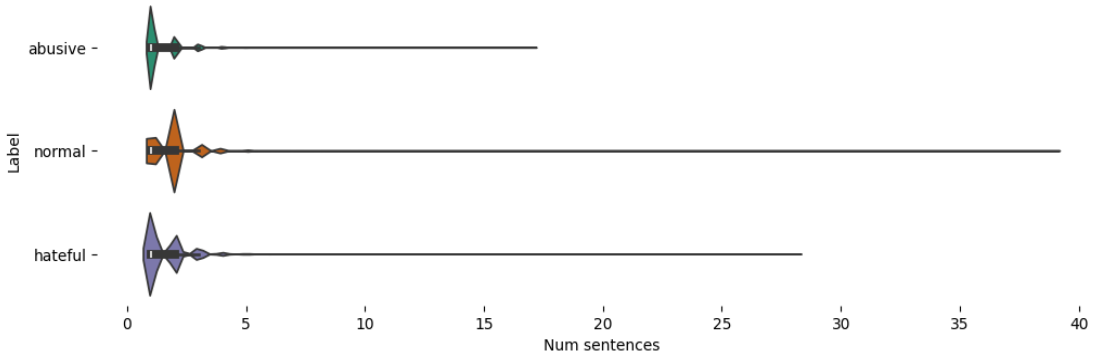

Figure 3: Number of Sentences Violin Plot

4.2.4 Sentiment/Swear Words

The final two characteristics gathered were sentiment and profanity count. Sentiment was calculated by using the NLTK pre-trained sentiment intensity analyzer VADER (Valence Aware Dictionary and sEntiment reasoner) [33]. The analyzer returns four values. The values are negative, neutral, positive, and compound. Negative, neutral, and positive sum up to 1 and cannot be negative. Compound is a value between -1 and 1 with -1 being extremely negative and 1 being

highly positive. This score is achieved by summing up the other three values, applying some internal weighting rules, and normalizing. This compound score was the value used to define a post's sentiment. The Python 'profanity-check' library was used to classify words as being profane or not [34]. It is a linear SVM model that was trained on 200k labeled samples, a 0 is assigned to non-curse words, and a 1 to curse words. This library was used to calculate the total number of profane words in a post. A minor disadvantage of this library is that it does not include swear words that are formed using special characters. Figure 4 demonstrates that sentiment appeared to highly delineate the three labels. Most normal posts were labeled neutral or positive leaning. Abusive posts were labeled neutral or negative leaning. Hateful posts were mostly labeled negative leaning. Hateful and abusive had similar medians and interquartile ranges but abusive had a higher frequency of neutral scores. The sentiments gathered seemed to align with the content of the posts mostly logically. It is reasonable rationale that hateful and abusive posts should lean negative while normal are neutral or positive. However, it is curious as to what content in certain abusive posts allowed some of them to pass as neutral in some cases. Figure 5 indicated that the profanity counts also proved very fruitful in separating the labels. Normal tweets had very little to no curse words while abusive and hateful posts had a few. Abusive and hateful have similar medians and interquartile ranges but hateful appeared to overall have higher frequencies across the range. Again, the deductions gathered from this data align with conventional thinking that it is sensible to see more curse words in posts from aggressors. It is not common to see extensive use of profanity in normal posts.

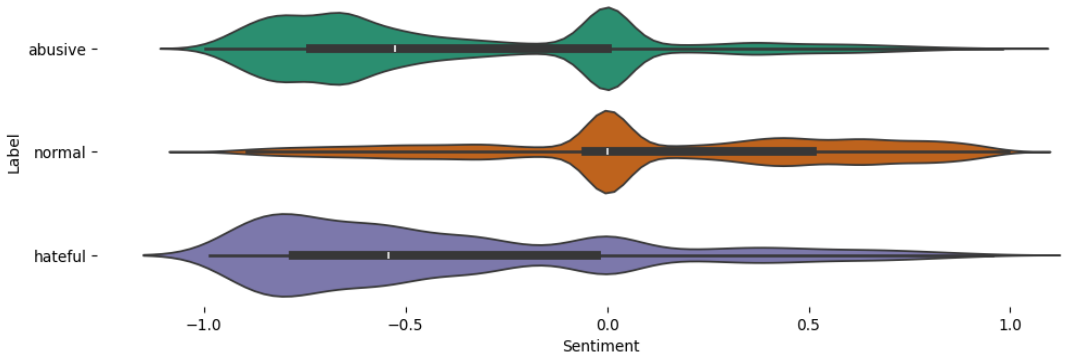

Figure 4: Post's Sentiment Violin Plot

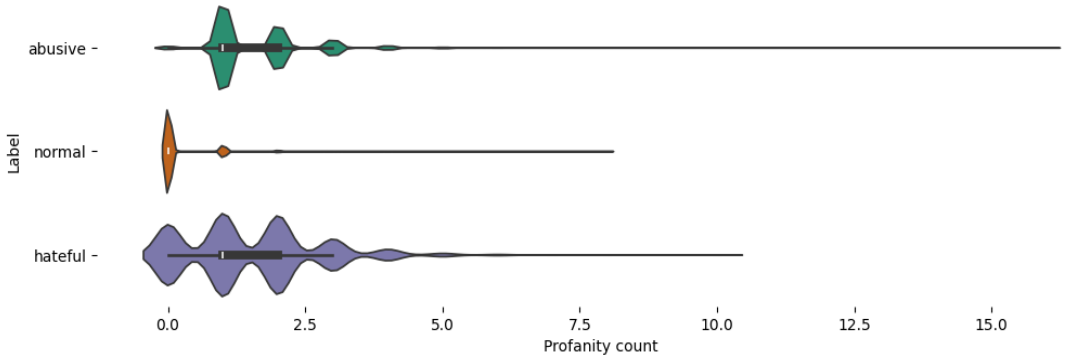

Figure 5: Number of Profanity Words Violin Plot

4.2.5 Gini Feature Evaluation

Gini importance was calculated on all the features. Gini importance is the importance of each feature based on its overall contribution to classifying the labels. As Figure 6 shows the features were arranged in descending order. The most important features by Gini importance were the number of profane words, sentiment, whether a post was a repost or not, the average sentence length, and the average word length. These top features proved the most helpful in separating and identifying the class labels of the posts. Some of these features also appear to align with the conclusions gathered from the violin plots.

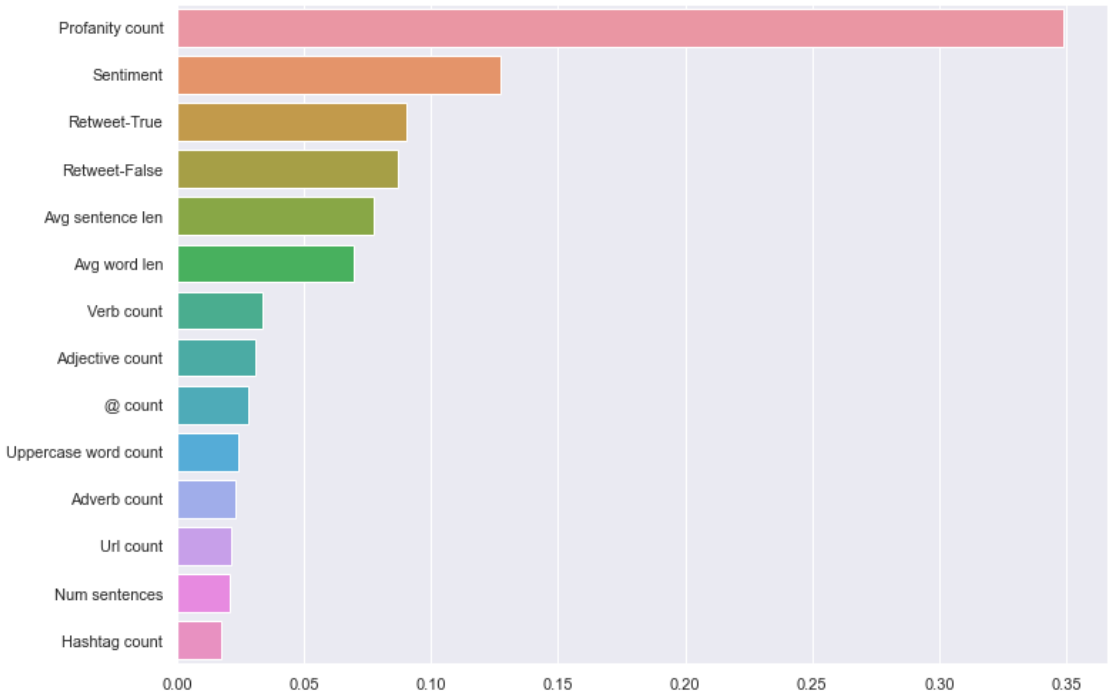

Figure 6: Gini Importance of Features

5. EXPERIMENTAL EVALUATION

In this chapter, we will investigate the performance of the models. A few different evaluations will be conducted. Firstly, all three batch-based models, decision tree (DT), random forest classifier (RFC), and logistic regression (LR), will be evaluated on their performance in classifying the labels abusive, hateful, and normal. A comparison will not only be conducted amongst the three models but also based on how frequently they were trained. The models trained on five days of data will be compared to those with only one day of training. Then the models will be analyzed in the second case, which is classifying the labels aggressive (hateful and abusive combined) and normal. Again, with a comparison of the number of days of training. After the batch models, the streaming models will be evaluated, Hoeffding Adaptive Tree (HAT), Extremely Fast Decision Tree (EFDT), and Streaming Random Patches (SRP). They also will be analyzed under the first case of identifying posts as abusive, hateful, and normal. Similarly to the batch models, the streaming models will be compared based on how many days of data the pre-trained language model used for finetuning. Then the streaming models will be compared on their performance with labelling aggressive and normal. Finally, the best batch-based model will be compared against the best streaming model for both classifying cases.

5.1 System and Software Specifications

The experiments were conducted on a 12-core, 1.70 GHz CPU with 32GB of RAM and 500GB of Gen4 SSD internal storage. The streaming approaches were implemented with scikit-multiflow v0.5.3 [27] and the batch-based models were implemented with scikit-learn v1.4.1 [35]. All figures were generated using either Microsoft Visio, seaborn, or Matplotlib [24], [36], [37]. The dataset used contained 86k X posts with the labels abusive, hateful, and normal. Experiments were also conducted by combining abusive and hateful into one new label aggressive.

5.2 Hyperparameter Tuning/Performance Metrics

Hyperparameter tuning was conducted on all the streaming characteristic models as well as the batch-based models. The naïve Bayes text model was not tuned because it has no hyperparameters. A grid search was utilized to find the ideal parameter settings. Table 1 shows the streaming models and table 2 shows the batch-based models with the parameters tuned, the range or options for those parameters, and the parameters that were chosen.

Table 1: Hyperparameter Tuning for Characteristic Streaming Models

Model	Parameter	Range or Options	Selected
HAT	Grace Period	100 - 500	500
	Split Criterion	InfoGain, Gini	InfoGain
	Split Confidence	.0000001 - .1	.1
	Tie Threshold	.01-.1	.075
	Leaf Prediction	NBA,MC,NB	NBA
EFDT	Leaf Prediction	NBA,MC,NB	MC
	Min Samples Reevaluate	10-30	20
	Rest of the parameters same as HT		
SRP	Subspace Mode	percentage, m, sqrtM1, MsqrtM1	MsqrtM1
	Training Method	RandomSubspaces, Resampling, RandomPatches	Resampling

Table 2: Hyperparameter Tuning for Batch-Based Models

Model	Parameter	Range or Options	Selected
DT	Criterion	Gini, Entropy, Log loss	Gini
	Splitter	Best, Random	Random
	Min Samples Split	2-4	4
	Min Samples Leaf	1-3	3
RFC	Criterion	Gini, Entropy, Log loss	Gini
	N estimators	50-150	100
	Min Samples Split	2-4	4
	Min Samples Leaf	1-3	1
	Verbose	0-2	1
LR	Max iter	50-150	150
	Solver	Lbfgs, Liblinear, Newton-cg, Newton-cholesky, Sag, Saga	Saga
	Verbose	0-2	0
	Penalty	L1, L2, Elastinet, None	None
	Multiclass	Multinomial, Ovr	Multinomial

Multiple performance metrics were assessed on all models. The metrics used were confusion matrix, accuracy, precision, recall, and f1-score. The metric primarily focused on for comparison and evaluation was f1-score due to its more holistic encapsulation of performance given that it integrates precision and recall into its calculation.

5.3 Batch Models

Figure 7 shows the trend in f1-score performance over 10 days of all the batch-based models on the three-label problem. All models were trained on the first day so none would display scores for day one. The models trained on five days of data will not have f1-scores for odd-numbered days. As is displayed by the figure the best-performing batch-based model was RFC. The version of RFC that seemed to perform the best on most days was trained on only one day of data. As table 3 further demonstrates DT struggled to perform at the same levels as RFC and LR. The metrics also show that for this small case, the number of days a model is trained has a negligible impact on performance. This most likely would not hold for a very long extended period as the models are sure to suffer from changes in trends but in the short term, not much is gained from frequent training.

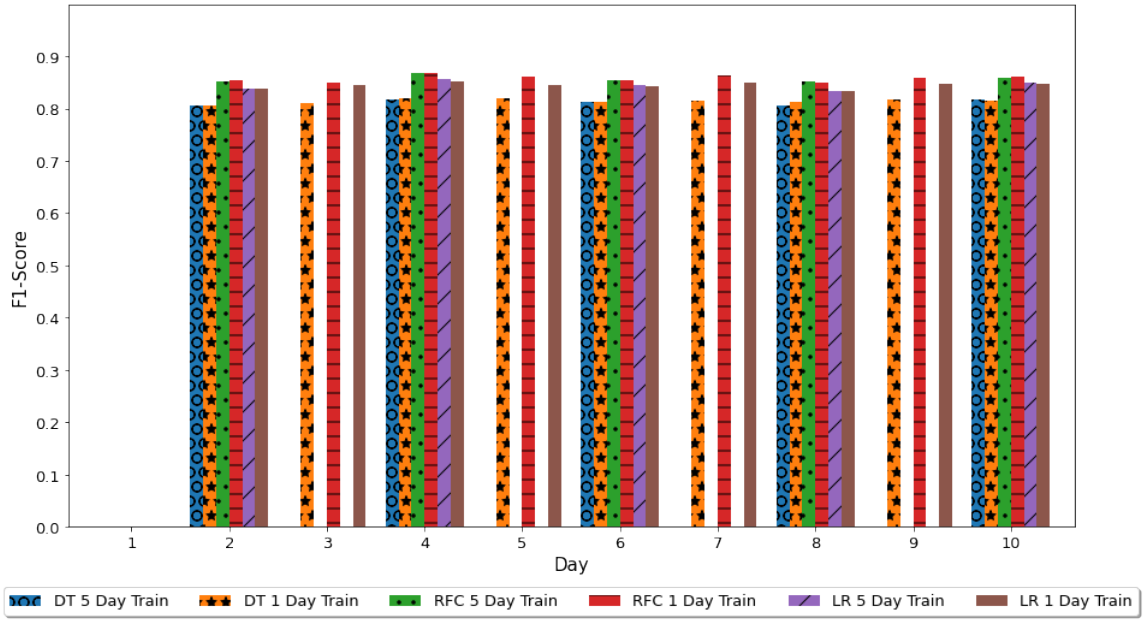

Figure 7: F1-score for All Batch Models on 3 Label Case

Table 3: Evaluation Metrics for All Batch Models on 3 Label Case

Metric	DT 5 Days Training	DT 1 Day Training	RFC 5 Days Training	RFC 1 Day Training	LR 5 Days Training	LR 1 Day Training
Accuracy	0.84	0.84	0.88	0.88	0.87	0.87
Precision	0.82	0.82	0.87	0.88	0.84	0.84
Recall	0.84	0.84	0.88	0.88	0.87	0.87
F1-score	0.83	0.83	0.86	0.86	0.85	0.85

Similarly, Figure 8 demonstrates the trend in f1-scores for all batch-based models but on the two-label case. Figure 8 shows that RFC is most often the best model, however, the discrepancy in score between RFC and LR is minimal. As table 4 presents, when the metrics are rounded the scores of RFC and LR are identical. Unsurprisingly all batch models are much better in the simpler case of just two labels. As before, training frequency had minimal to no impact on performance in this condensed period.

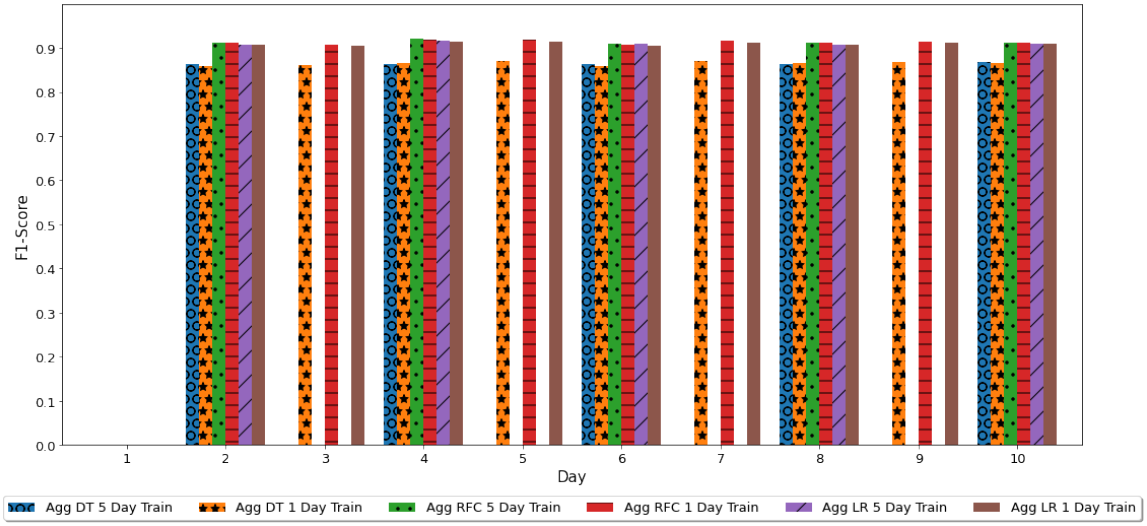

Figure 8: F1-score for All Batch Models on 2 Label Case

Table 4: Evaluation Metrics for All Batch Models on 2 Label Case

Metric	DT 5 Days Training	DT 1 Day Training	RFC 5 Days Training	RFC 1 Day Training	LR 5 Days Training	LR 1 Day Training
Accuracy	0.88	0.89	0.91	0.91	0.91	0.91
Precision	0.88	0.89	0.91	0.91	0.91	0.91
Recall	0.88	0.89	0.91	0.91	0.91	0.91
F1-score	0.88	0.89	0.91	0.91	0.91	0.91

5.4 Streaming Models

Similarly to the batch models, all the streaming models for the three-label case were evaluated as a group. As Figure 9 shows of EFDT, HAT, and SRP, SRP was consistently the worst performer regardless of the amount of fine-tuning training the LM had for the sentence encodings given to the naïve Bayes model. Figure 9 also demonstrates that when EDFT and HAT were paired with

naïve Bayes models that used encodings from the less fine-tuned LM, they also underperformed. Table 5 establishes that EDFT and NB with five days of training and HAT and NB with five days of training perform very similarly. Given that all models use the same naïve Bayes for the LM sentence encodings SRP must have been the factor that held the overall model back.

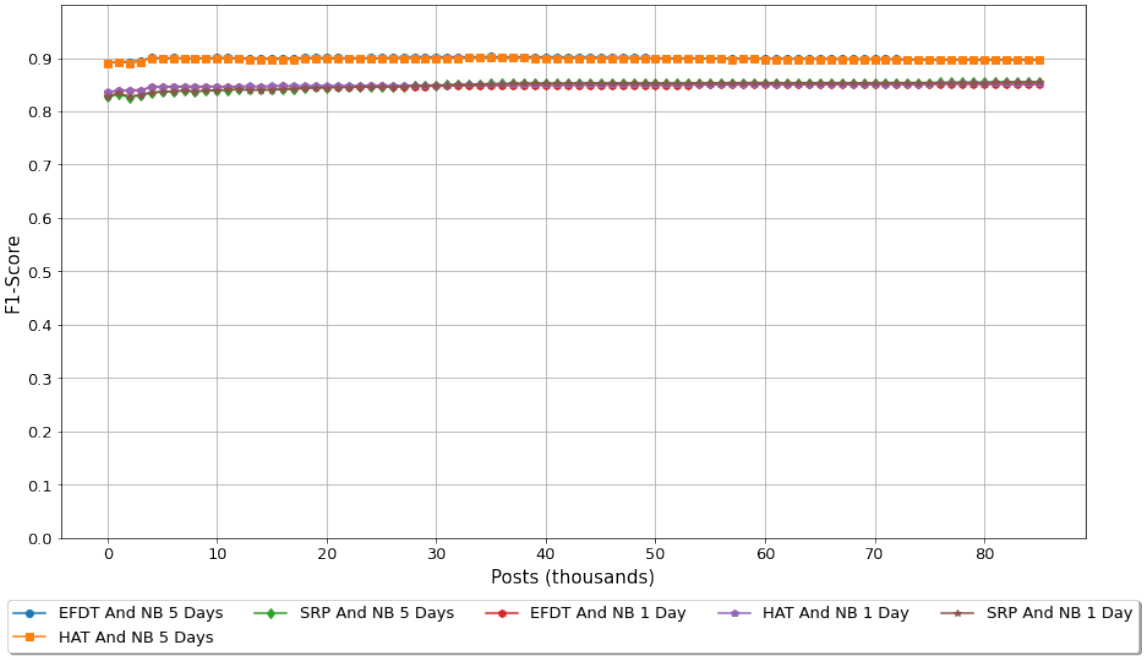

Figure 9: F1-score for All Streaming Models on 3 Label Case

Table 5: Evaluation Metrics for All Streaming Models on 3 Label Case

Metric	EFDT And NB 5 Days Training	EFDT And NB 1 Day Training	HAT And NB 5 Days Training	HAT And NB 1 Day Training	SRP And NB 5 Days Training	SRP And NB 1 Day Training
Accuracy	0.88	0.84	0.88	0.84	0.88	0.88
Precision	0.92	0.87	0.91	0.87	0.88	0.88
Recall	0.88	0.84	0.88	0.84	0.88	0.88
F1-score	0.90	0.85	0.90	0.85	0.86	0.85

When looking at Figure 10 for the two-label case the conclusions are much the same as the three-label case. SRP still underperforms even in the easier case and EFDT and HAT underperform again when paired with the naïve Bayes using the less fine-tuned LM. Table 6 also displays much of the same with EFDT and HAT using the more finetuned naïve Bayes having similar performances.

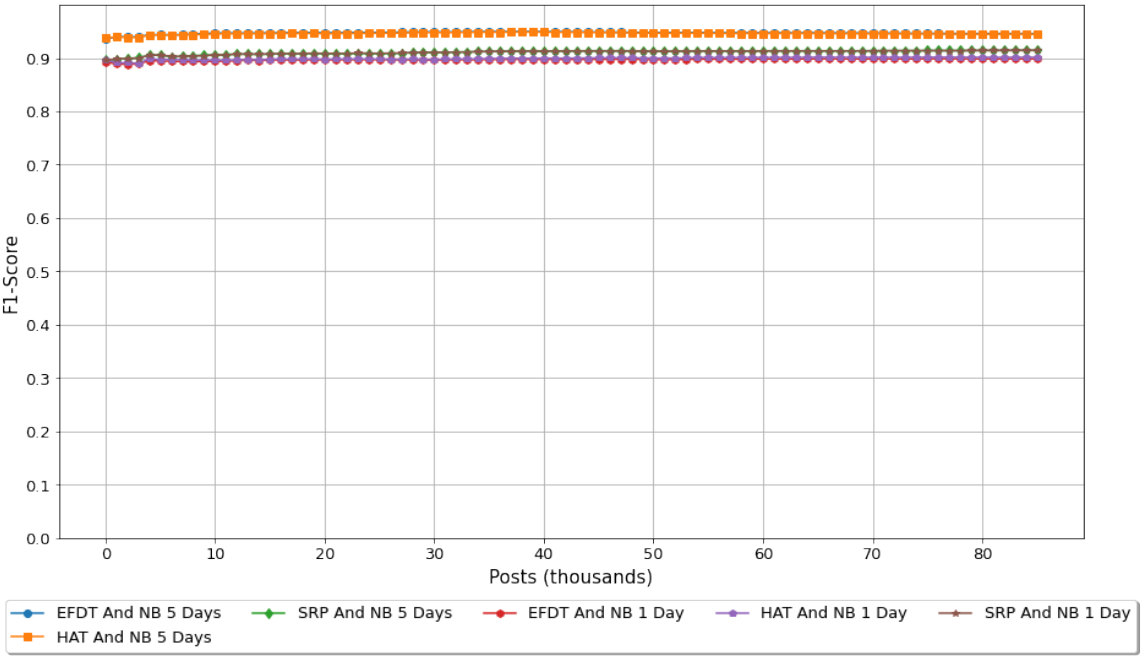

Figure 10: F1-score for All Streaming Models on 2 Label Case

Table 6: Evaluation Metrics for All Streaming Models on 2 Label Case

Metric	EFDT And NB 5 Days Training	EFDT And NB 1 Day Training	HAT And NB 5 Days Training	HAT And NB 1 Day Training	SRP And NB 5 Days Training	SRP And NB 1 Day Training
Accuracy	0.95	0.90	0.94	0.90	0.91	0.91
Precision	0.95	0.90	0.94	0.90	0.91	0.91
Recall	0.95	0.90	0.94	0.90	0.91	0.91
F1-score	0.95	0.90	0.94	0.90	0.91	0.91

5.5 Batch vs. Streaming Models

Table 7 shows the performance of the best streaming model compared to the best batch model and the best streaming model from Herodotou et al. for both cases [4]. Both streaming models are equal to or better than the batch model in most categories for classifying posts as either abusive, hateful, or normal. The two streaming models are very competitive however, the proposed model does manage to show a slight edge regarding precision and f1-score. For the two-label case of aggressive and normal, again both streaming models are at least slightly more performant than the batch models. Here there is more of an edge in performance towards the proposed streaming model in comparison to the other. Much of the difference in performance between the two streaming models most likely stems from the integration of LM encodings with naïve Bayes in the proposed model. While there are different streaming models used for the post's characteristics compared to previous work they most likely would not have as much impact on performance. The LM encodings most likely provide the naïve Bayes with more specific and substantial attributes to aid in the classification of posts. There is an argument to be made that the training that would be used for batch-based models has just been shifted to the LM model in this proposed approach and it is therefore not a true streaming real-time approach.

Table 7: Evaluation Metrics for Best Models

Metric		***3 Label Case***			***2 Label Case***	
	EFDT And NB 5 Days Training	**RFC 5 Days Training**	**Herodotou et al. HT**	**EFDT And NB *5* Days Training**	**RFC 5 Days Training**	**Herodotou et al. HT**
Accuracy	0.88	0.88	0.89	0.95	0.91	0.93
Precision	0.92	0.87	0.85	0.95	0.91	0.92
Recall	0.88	0.88	0.89	0.95	0.91	0.90
F1-score	0.90	0.86	0.87	0.95	0.91	0.91

The pre-trained LM model in this approach received fine-tuning to increase performance given that a smaller, less resource-heavy LM model was used in this approach. As will be discussed further in Chapter 6, in real practice this would not be necessary because the social media companies that would deploy this approach would have superior resources to ingrate higher quality and larger pre-trained LM models, thereby removing the need for fine-tuning. The fine-tuned learning employed in this approach merely served as a proof of concept for more substantial pre-trained LM models that require more resources.

While the proposed approach proved to be viable and an improvement, there are still areas of weakness in the approach. As Figure 11 shows all the streaming models still struggle with correctly identifying the hateful label. The hateful label most likely has a lot of overlap with the abusive label in terms of characteristics. The finer details that distinguish the two labels apart appear difficult to identify.

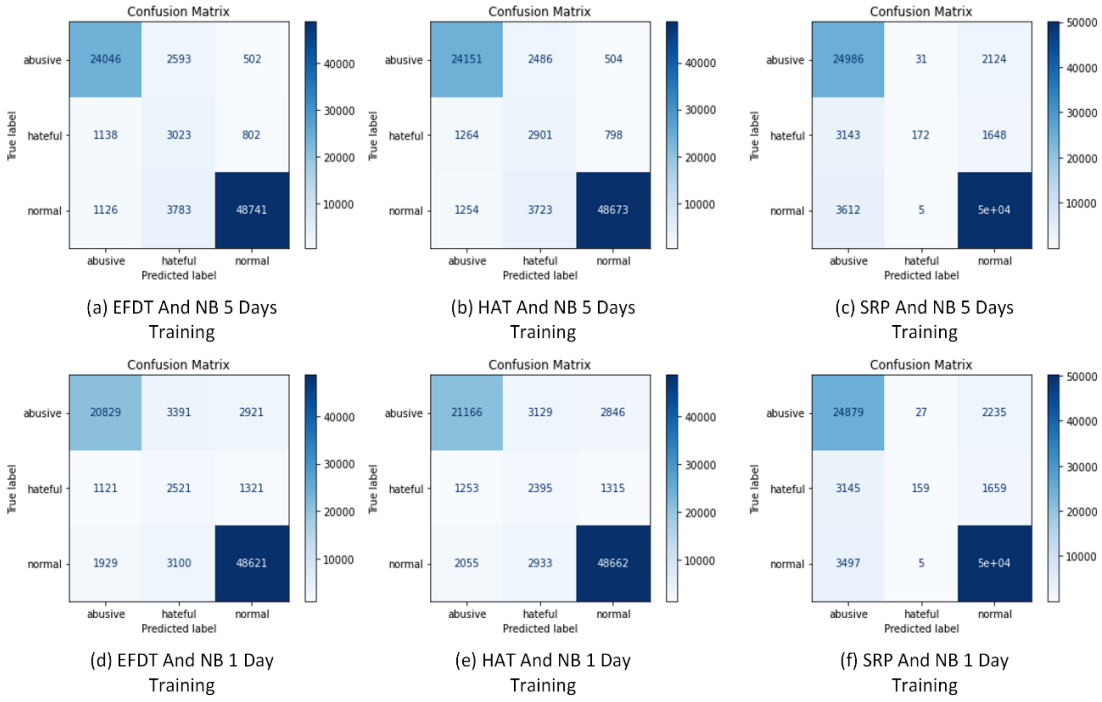

Figure 11: Confusion Matrix for (a) EFDT And NB 5 Days Training, (b) HAT And NB 5 Days Training, (c) SRP And NB 5 Days Training, (d) EFDT And NB 1 Day Training, (e) HAT And NB 1 Day Training, and (f) SRP And NB 1 Day Training

6. CONCLUSION AND FUTURE WORK

In this work, a new form of moderation model was proposed to facilitate the ability of X and other social media applications to better mitigate unwanted behavior such as bullying through abusive and hateful comments. The proposed approach expands on previous work in this area by integrating two models instead of the use of a singular model. Previous work utilized a singular streaming model that labeled posts on the characteristics gathered from feature extraction only. The new approach combines a post's characteristics streaming model with a language sentence encoding streaming model to label X posts as abusive, hateful, or normal. The reason for this new approach was to boost overall performance by including new features that should assist in the classification of posts. The evaluation metrics bolster the idea that this approach could prove more fruitful. The streaming models were able to outperform traditional batch-based learners in certain metrics and were highly competitive with previous streaming models.

For future work, there is much room for investigation and possible improvement. While the models' showed improvement and high capability, identifying hateful tweets still appeared challenging. Better accuracy could be achieved using larger, better pre-trained language models. The language model implemented in this approach was smaller and less performant due to limited resources. A more advanced language model such as Llama 2 and GPT-4 should boost performance [20], [22]. Another area for investigation is the scope of the model. The model focuses only on the labels abusive, hateful, and normal from X posts. The viability of the approach on other social media platforms such as Facebook, Instagram, and YouTube should be investigated. Given that certain features extracted in this model are unique to X and therefore not found on other platforms some modifications would need to be made to address those, for instance, Instagram and YouTube comments would not contain the feature "repost". Other forms of bullying were also not considered. For instance, the term hateful was used collectively for

forms of bullying such as racism, sexism, homophobia, or xenophobia. The approach should be analyzed in its abilities with these more granular labels and other forms of bullying that may not entail profanity such as trolling, doxxing, or dissing. This model may find forms of bullying that do not entail profanity or highly charged language initially challenging but given the online-learning aspect of the approach, the model should eventually adapt to the new labels. This approach did not consider a user's social network connections or profile such as the number of friends they have, how many people they follow, how many likes they receive, how many reposts they receive, how old is their account, or how often they post. Adding a user's social network connections and profile features may cause concerns over user privacy. However, given that the proposed approach purely analyzes the content of posts and utilizes no information about the user who posted it there should be little concern for privacy. Also, after the model reads in a post it is immediately discarded so no collection of posts that could be accessed later is ever maintained. All these and other characteristics may be useful in finding further gains in performance.

REFERENCES

[1] A. Founta *et al.*, "Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior," *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 12, no. 1, Jun. 2018. doi:10.1609/icwsm.v12i1.14991

[2] D. Chatzakou *et al.*, "Mean Birds: Detecting Aggression and Bullying on Twitter," *Proceedings of the 2017 ACM on Web Science Conference*, Jun. 2017. doi:10.1145/3091478.3091487

[3] H. Herodotou, D. Chatzakou, and N. Kourtellis, "Catching them red-handed: Real-time Aggression Detection on Social Media," *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, Apr. 2021. doi:10.1109/icde51399.2021.00211

[4] H. Herodotou, D. Chatzakou, and N. Kourtellis, "A Streaming Machine Learning Framework for Online Aggression Detection on Twitter," *2020 IEEE International Conference on Big Data (Big Data)*, Dec. 2020. doi:10.1109/bigdata50022.2020.9377980

[5] A. S. Nagdeve and Manisha. M. Ambekar, "Spam detection by designing machine learning approach in Twitter Stream," *2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC)*, Oct. 2020. doi:10.1109/icsidempc49020.2020.9299607

[6] H. Tajalizadeh and R. Boostani, "A novel stream clustering framework for SPAM detection in Twitter," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 3, pp. 525–534, Jun. 2019. doi:10.1109/tcss.2019.2910818

[7] E. Alothali, H. Alashwal, M. Salih, and K. Hayawi, "Real time detection of social bots on Twitter using machine learning and Apache Kafka," *2021 5th Cyber Security in Networking Conference (CSNet)*, Oct. 2021. doi:10.1109/csnet52717.2021.9614282

[8] F. Concone, G. L. Re, M. Morana, and S. K. Das, “SPADE: Multi-stage spam account detection for online social networks,” *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 4, pp. 3128–3143, Jul. 2023. doi:10.1109/tdsc.2022.3198830

[9] M. Abhineswari and R. Priyadarshini, “Analyzing large-scale Twitter real time streaming data with manifold machine learning algorithms in Apache Spark,” *2023 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI)*, Dec. 2023. doi:10.1109/icdsaai59313.2023.10452549

[10] B. Gokulakrishnan, P. Priyanthan, T. Ragavan, N. Prasath, and As. Perera, “Opinion mining and sentiment analysis on a Twitter Data Stream,” *International Conference on Advances in ICT for Emerging Regions (ICTer2012)*, Dec. 2012. doi:10.1109/icter.2012.6423033

[11] K. Mahor and A. K. Manjhvar, “Public sentiment assessment of coronavirus-specific tweets using a transformer-based Bert Classifier,” *2022 International Conference on Edge Computing and Applications (ICECAA)*, Oct. 2022. doi:10.1109/icecaa55415.2022.9936448

[12] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, “A survey on Ensemble Learning for Data Stream Classification,” *ACM Computing Surveys*, vol. 50, no. 2, pp. 1–36, Mar. 2017. doi:10.1145/3054925

[13] H. M. Gomes *et al.*, “Adaptive random forests for evolving data stream classification,” *Machine Learning*, vol. 106, no. 9–10, pp. 1469–1495, Jun. 2017. doi:10.1007/s10994-017-5642-8

[14] L. E. Boiko Ferreira, H. Murilo Gomes, A. Bifet, and L. S. Oliveira, “Adaptive random forests with resampling for Imbalanced Data Streams,” *2019 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2019. doi:10.1109/ijcnn.2019.8852027

[15] M. Bahri, H. M. Gomes, A. Bifet, and S. Maniu, “CS-ARF: Compressed adaptive random forests for evolving data stream classification,” *2020 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2020. doi:10.1109/ijcnn48605.2020.9207188

[16] H. Abdulsalam, D. B. Skillicorn, and P. Martin, “Streaming random forests,” *11th International Database Engineering and Applications Symposium (IDEAS 2007)*, Sep. 2007. doi:10.1109/ideas.2007.4318108

[17] E. Garcia-Martin, N. Lavesson, H. Grahn, E. Casalicchio, and V. Boeva, “Hoeffding trees with Nmin adaptation,” *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, Oct. 2018. doi:10.1109/dsaa.2018.00017

[18] C. Manapragada, M. Salehi, and G. I. Webb, “Extremely fast hoeffding adaptive tree,” *2022 IEEE International Conference on Data Mining (ICDM)*, Nov. 2022. doi:10.1109/icdm54844.2022.00042

[19] N. Kourtellis, G. De Francisci Morales, A. Bifet, and A. Murdopo, “VHT: Vertical Hoeffding Tree,” *2016 IEEE International Conference on Big Data (Big Data)*, Dec. 2016. doi:10.1109/bigdata.2016.7840687

[20] OpenAI *et al.*, ‘GPT-4 Technical Report’, *arXiv [cs.CL]*. 2024.

[21] S. Bubeck *et al.*, ‘Sparks of Artificial General Intelligence: Early experiments with GPT-4’, *arXiv [cs.CL]*. 2023.

[22] H. Touvron *et al.*, ‘Llama 2: Open Foundation and Fine-Tuned Chat Models’*, arXiv [cs.CL]*. 2023.

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, ‘BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding’, *arXiv [cs.CL]*. 2019.

[24] Microsoft Corporation, *Microsoft Visio*. 2018.

[25] N. Reimers and I. Gurevych, “Sentence-bert: Sentence embeddings using Siamese Bert-Networks,” *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019. doi:10.18653/v1/d19-1410

[26] P. Bajaj *et al.*, ‘MS MARCO: A Human Generated MAchine Reading COmprehension Dataset’, *arXiv preprint arXiv:1611. 09268*, 2016.

[27] J. Montiel, J. Read, A. Bifet, and T. Abdessalem, ‘Scikit-Multiflow: A Multi-output Streaming Framework’, *Journal of Machine Learning Research*, vol. 19, no. 72, pp. 1–5, 2018.

[28] M. Bahri, S. Maniu, and A. Bifet, “A sketch-based naive Bayes algorithms for evolving data streams,” *2018 IEEE International Conference on Big Data (Big Data)*, Dec. 2018. doi:10.1109/bigdata.2018.8622178

[29] C. Manapragada, G. I. Webb, and M. Salehi, “Extremely Fast Decision Tree*,”* *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul. 2018. doi:10.1145/3219819.3220005

[30] A. Bifet and R. Gavaldà, “Adaptive Learning from Evolving Data Streams,” *Advances in Intelligent Data Analysis VIII*, pp. 249–260, 2009. doi:10.1007/978-3-642-03915-7_22

[31] C. Manapragada, M. Salehi, and G. I. Webb, “Extremely Fast Hoeffding Adaptive Tree,” *2022 IEEE International Conference on Data Mining (ICDM)*, Nov. 2022. doi:10.1109/icdm54844.2022.00042

[32] H. M. Gomes, J. Read, and A. Bifet, “Streaming Random Patches for Evolving Data Stream Classification,” *2019 IEEE International Conference on Data Mining (ICDM)*, Nov. 2019. doi:10.1109/icdm.2019.00034

[33] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. Beijing: O'Reilly, 2009.

[34] 'profanity-check'. [Online]. Available: https://pypi.org/project/profanity-check/. [Accessed: 3-Nov-2023].

[35] F. Pedregosa *et al.*, 'Scikit-learn: Machine Learning in Python', *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[36] M. L. Waskom, 'seaborn: statistical data visualization', *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021.

[37] J. D. Hunter, 'Matplotlib: A 2D graphics environment', *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.