Scalable and Timely Detection of Cyberbullying in Online Social Networks

Rahat Ibn Rafiq
Department of Computer Science
University of Colorado Boulder
rahat.rafiq@colorado.edu

Homa Hosseinmardi
Information Sciences Institute
University of Southern California
homahoss@isi.edu

Richard Han
Department of Computer Science
University of Colorado Boulder
richard.han@colorado.edu

Qin Lv
Department of Computer Science
University of Colorado Boulder
qin.lv@colorado.edu

Shivakant Mishra
Department of Computer Science
University of Colorado Boulder
mishras@cs.colorado.edu

ABSTRACT

Cyberbullying in Online Social Networks (OSNs) has grown to be a serious problem among teenagers. While a considerable amount of research has been conducted focusing on designing highly accurate classifiers to automatically detect cyberbullying instances in OSNs, two key practical issues remain to be worked upon, namely scalability of a cyberbullying detection system and timeliness of raising alerts whenever cyberbullying occurs. These two issues form the motivation of our work. We propose a multi-stage cyberbullying detection solution that drastically reduces the classification time and the time to raise alerts. The proposed system is highly scalable without sacrificing accuracy and highly responsive in raising alerts. The design is comprised of two novel components, a dynamic priority scheduler and an incremental classification mechanism. We have implemented this solution, and using data obtained from Vine, we conducted a thorough performance evaluation to demonstrate the utility and scalability of each of these components. We show that our complete solution is significantly more scalable and responsive than the current state of the art.

CCS CONCEPTS

• **Networks** → **Online social networks**; • **Human-centered computing** → **Social networks**; • **Applied computing** → Sociology;

KEYWORDS

Scalable Systems, Cyberbullying , Social Networks

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SAC 2018, April 9–13, 2018, Pau, France
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5191-1/18/04…$15.00
https://doi.org/10.1145/3167132.3167317

ACM Reference Format:
Rahat Ibn Rafiq, Homa Hosseinmardi, Richard Han, Qin Lv, and Shivakant Mishra. 2018. Scalable and Timely Detection of Cyberbullying in Online Social Networks. In *SAC 2018: SAC 2018: Symposium on Applied Computing , April 9–13, 2018, Pau, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3167132.3167317

1 INTRODUCTION

Unprecedented growth in the popularity of OSNs, especially among teenagers, has unfortunately resulted in significant increase in cyberbullying. The main differences between bullying and cyberbullying are the facts that the Internet can help the perpetrators of cyberbullying hide their identities, cyberbullying can be incessant because of the availability and access of the Internet, and the possibility of cyberbullying being viral and exposing the victims to an entire virtual world [29]. Numerous instances [5] and devastating consequences of cyberbullying [1, 8, 27] have led researchers to explore detection of cyberbullying incidents in OSNs like Ask.fm, Instagram, Vine, Twitter etc. [3, 12, 13, 19, 23, 28]. These works have mostly followed the methodology of collecting and labeling data from OSNs and designing cyberbullying classifiers. Past work has also investigated the issue of identifying imbalance of power between perpetrators and victims, which is a key feature of bullying, and distinguishing between cyber-aggression and cyberbullying [16], thus paving the way for highly accurate classifiers. Figure 1 shows an example of cyberbullying in Vine.

While progress has been made on the accuracy of classifiers for cyberbullying detection, there are two key practical issues that have largely been ignored to date. The first issue concerns the scalability of the cyberbullying detection solutions. OSNs, of course, involve an enormous amount of data, on the order of several hundred gigabytes per day. For example, it has been reported that for Vine around 39 million videos have been shared since it was introduced [26] while for Instagram, the amount of shared media is 40 billion[18].

The second issue concerns the timeliness of raising alerts whenever cyberbullying incidents are suspected. Cyberbullying is different from traditional, face-to-face bullying, because it can occur 24/7, and perpetrators can be anonymous and

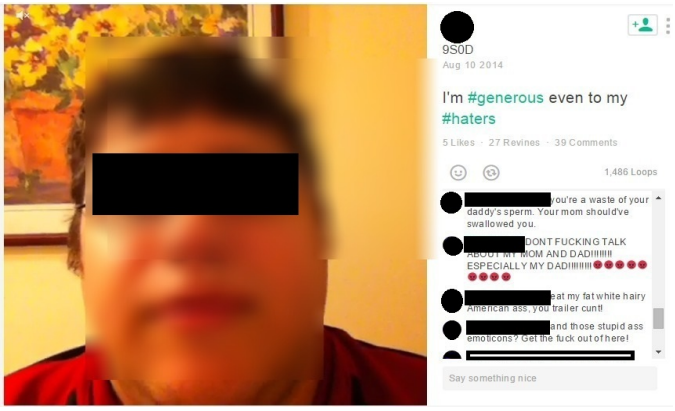


Figure 1: An example of cyberbullying on Vine.

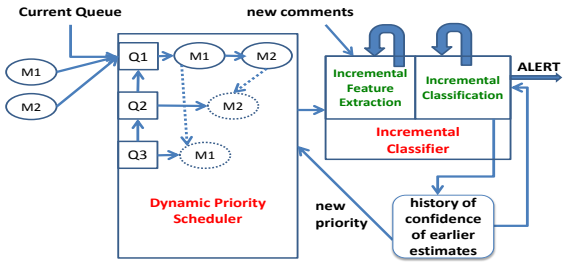


Figure 2: Scalable and responsive cyberbullying detection architecture.

have easy access to sophisticated tools to launch cyberbullying attacks. Furthermore, the consequences can be disastrous and it is extremely important to provide the necessary support to the victims as early as possible. So, a timely detection of cyberbullying is of paramount importance, so that an alert can be raised as soon as possible.

In this paper, we propose a multi-stage cyberbullying detection solution designed to improve the scalability and responsiveness of cyberbullying detection. *To the best of our knowledge, we are the first to propose a scalable and responsive solution to cyberbullying detection in OSNs.* A key property of the solution is that it achieves sufficient classification accuracy while accomplishing these two goals. The solution consists of two key components, namely, a dynamic, multilevel priority scheduler for improved responsiveness, and an incremental feature extraction and classification stage for scaling. Using online social networking data from Vine, we demonstrate the utility of both of these components, and show that our complete cyberbullying detection solution is significantly more scalable and responsive than the current state-of-the-art. We make the following important contributions:

- We propose an incremental computational design for feature extraction and classification that reuses previous classification results to reduce overhead with minimal impact on accuracy.
- We propose a dynamic, multi-level priority scheduler that assigns high preference to potential cyberbullying media-sessions, thereby improving responsiveness of the solution.
- Using real world data from Vine, we demonstrate that our integrated system substantially improves the scalability of cyberbullying, making cyberbullying detection feasible for Vine-scale social networks.
- We further demonstrate how our system scales to monitor much larger, Instagram-scale networks.

2 RELATED WORK

The majority of research on cyberbullying detection has focused on improving the accuracy of cyberbullying detection classifiers[6, 15, 22, 30]. Several cyberbullying detection applications have been developed in recent years [7]. Most of these applications (e.g., Mobicip) introduce parental control including category blocking, time limits, Internet activity reports, blocked phrases, and YouTube filtering, whereas others, e.g. iAnon and GoGoStat search for specific profanity words. Specific keyword based research on event detection have also been performed [20, 24] where crime and disaster events were extracted after hitting the Twitter API for a list of specific keywords(tornado, earthquake) to get the tweets related to an event. Cyberbullying, in contrast, is much more subtle. As shown in [13], profanity can not solely be an indicator of cyberbullying. So it is important to make use of the definition of cyberbullying and take into account the repetitive nature of perpetrated aggressiveness and imbalance of power while building an accurate cyberbullying detection solution.

As far as we know, none of the prior research has addressed the issue of computational scalability and/or responsiveness in the context of cyberbullying detection. Scalability has been, however, an important factor for other research areas such as misbehavior detection in online video chat services [4, 32] and cyber-attack detection in communities [11].

3 DESIGN OVERVIEW

Our goal of this research is to propose a cyberbullying detection system with two key characteristics, namely, scalable to handle large OSNs without sacrificing accuracy and timeliness of raising an alert when a cyberbullying instance takes place. To this aim, we face two key challenges. First, how to scale up the system while retaining a reasonable accuracy. Second, how to design a system so as to make sure an alert is raised as soon as a cyberbullying instance takes place while monitoring large number of media sessions (media and its associated comments in Instagram or Vine). In the following subsections, we describe the two components of our system that address these challenges.

X_n : saved feature vector values for n comments from before for all features;
δn : new comments to be processed;
X_n^i : feature vector value of i-th feature for n comments;
$X_{\delta n}^i$: feature vector value of i-th feature for δn comments;
$|X_n|$: number of total features;
forall i *in* $1, 2, ..., |X_n|$ **do**
  $X_{n+\delta n}^i : X_n^i$+Compute($X_{\delta n}^i$);
end

Algorithm 1: IncrementalFeatureExtraction()

3.1 Incremental Classifier

Our first challenge is to build a cyberbullying detection classifier that is scalable when it comes to time and computing resources while also retaining sufficient accuracy performance. While sophisticated deep learning classifiers have been recently introduced to solve complex problems with high accuracy [14, 17], they come up with considerable computational baggage. For example, in [14], the authors used deep learning in real time to process one 1080p video frame in 644ms using Samsung S7 with leveraging high performance GPUs (12 GPUs) and 4GB memory. While it is tempting to use deep learning for our system, we want our classifier to be able to leverage lightweight computational resources (Amazon AWS free tier 1GB memory, for example). In addition to being computationally lightweight, we also want our classifier to be faster than the slower current state-of-the-art AdaBoost[23] (as shown later in Table 1) to classify when new comments for a media session comes in, without sacrificing accuracy. We see that both these cases (deep learning and AdaBoost), while being highly accurate, does not meet two key challenges, being computational resource-wise lightweight and efficient, respectively.

Our approach towards lowering computational resource scalability and improving efficiency while retaining sufficient accuracy is to incorporate *incremental computation* [9, 10] into the design of the potential classifier. Incremental computation reuses data from previous stages within the current stage, thus resulting in less computational complexity. Traditional classifiers need to execute a full run as each new datum arrives, e.g. new comment for a media session. Instead, our approach reuses previous stages' results and combines them with the new comments, thereby reducing computational cost, rendering the solution scalable. We seek ways to apply this incremental approach to both feature extraction and classification stages of the potential classifier. To this aim, we seek to employ a classifier that, during feature extraction stage, uses features which, by nature can be incrementally linear in the sense that once the values corresponding to these features have been computed for the first n comments, then when δn new comments arrive, we only have to compute the individual feature vector values for the new δn comments while reusing the values for the previous n comments to compute the overall feature vector for the $n + \delta n$ comments. This dramatically reduces resource and computation cost because this approach is driven by δn at each run instead of $n+\delta n$. Algorithm 1 provides a pseudo-code of the incremental feature extraction algorithm for our incremental detector. Similarly, the candidate classifier should also be able to leverage this incremental approach during classification stage once the feature vectors are extracted from new data.

We found that Logistic Regression (LR) was the most promising classifier that met all these aforementioned criteria. LR works as follows: if we have n features $a_i, i = 0, 1, 2, 3..., n-1$, after training, LR assigns a weight $w_i, i = 0, 1, 2..., n-1$ to each of those features, and then computes the combined features value $c = \sum_0^{n-1} a_i w_i$. This value is fed to a sigmoid function with output ranging from 0 to 1[31]. The way we can leverage incremental computation in LR is as follows: LR takes as input a set of features X, and during the training process, the classifier generates a set of weights θ corresponding to those features. When a new media session comes in, the feature extraction step computes a matrix X for that particular media session and computes $C = X\theta$, which is then used to make the corresponding prediction. For the incremental feature extraction sub-component, we save the X_{old} value for the previous n comments, compute $X_{\delta n}$ for the new set of δn arrived comments and compute the new X by combining X_{old} and $X_{\delta n}$ instead of computing X all over again for all $n + \delta n$ comments. For the incremental classification part, we only use those components of X that have been changed to compute $C = X\theta$ instead of doing the full $X\theta$ computation. For this purpose, we save $X_i \times \theta_i, \forall i$, where X_i is the i-th feature at time t. Then we only change the corresponding feature vector value X_i at time $t + \delta t$ if it has been changed by comparing it to the previous saved X_i at time t. If it has been changed, only then we take it into the account to compute $_{\forall i} X_i \times \theta_i$ by simple addition and subtraction instead of full scale matrix multiplication.

To be able to use incremental computation in feature extraction stage, LR has to be able to make of features who are, by nature, amenable to feature extraction. In addition, our incremental logistic regression also has to show sufficient efficiency and scalability over the current state-of-the-art. All these goals have to be met without sacrificing crucial classification qualities like precision and recall. In this section, we provided the design of incremental techniques our LR classifier uses to scale up. From now onward, we will refer to this incremental LR classifier as classifier. In section 4.1, we first justify our choice of LR by comparing its execution time, precision and recall with current state-of-the-art[23] while making use of features that accommodate incremental computation. We then justify using the incremental computation approach in the LR by comparing its scalability performances with standard LR that does not use incremental approach.

3.2 Dynamic Priority Scheduler

While leveraging incremental computation helped our classifier to gain reasonable scalability, we found that it still lacked the other key issue that a potential cyberbullying detection system has to address: responsiveness. As the first step towards tackling this issue, we make use of two key observations. First, *not all media sessions need to be monitored equally*. The theme is to apply limited resources to where they needed the most. As most media sessions are not bullying in nature [23], we should be able to apply our resources on media sessions that are most likely to result in cyberbullying. This observation makes it natural to build a scheduler that just keeps monitoring media sessions with high priority and discarding all the low priority ones. We call this scheduler Static Priority Scheduler (SPS). Our investigation of the performance of SPS on Vine labeled data [23] found its precision and recall to be 70 and 58 percent respectively. The low recall value shows that by totally ignoring the media sessions that were given low priority some stage of their lifetimes to achieve responsiveness, we miss a significant portion of potential cyberbullying media sessions. This trade-off between performance and responsiveness led us to make our second observation: *a media session, with its incoming stream of comments, can slowly evolve into a cyberbullying instance even if it started as a normal one at its early stages of lifetime and vice versa*. As new comments arrive for a media session, it may become more or less indicative of cyberbullying, depending on the nature of the newly arrived comments. This means, it is important to examine all sessions including the ones with low priority, as some of them may evolve into cyberbullying sessions during later stages.

Investigation results after running SPS and the second observation formed the motivation of the design of our Dynamic Priority Scheduler (DPS). We define two levels of priority, namely high and low, for all media sessions and assign a high priority to all newly created media sessions. Now our challenge is to accommodate learning based in new comments so as to dynamically vary each media session's priority. After each invocation of our incremental classifier component (Section 3.1), a confidence value[2] of how likely a media session contains cyberbullying is generated. We make use of the history of these confidence values to dynamically change a media session's priority. The reason for using history as opposed to just the most recent confidence value has to do with the definition of cyberbullying. Cyberbullying is defined as an aggressive online behavior that is *carried out repeatedly* against a person who *cannot easily defend himself or herself*, creating a power imbalance [16]. To identify repeated aggressive behavior or whether a victim can defend himself or herself, we need to consider a much longer history than just the most recent confidence value. We calculate the average of all past confidence values for past classifications and current classification of a particular media session and compare that with a threshold value. If the average confidence value is more than a certain threshold value, we assign a high priority to the session and if the average value is lower than the threshold value, we assign a low priority. Algorithm 2 illustrates our priority setting algorithm using an average confidence threshold (0.2 in this example). Upon prioritizing the media sessions, we run our classifier component on high priority media sessions more frequently and postpone the classification and processing of low priority media sessions until a later phase, thus achieving responsiveness without sacrificing recall performances.

In this section, we outlined the motivation and design of introducing our dynamic priority scheduler into the proposed cyberbullying detection system. In Section 4.2, we first determine what threshold is appropriate for our DPS by comparing it with our baseline round-robin scheduler, where all media sessions always have the same priority and, so, monitored equally. Then we demonstrate through thorough experiments that DPS introduces significant responsiveness gain over round robin scheduler, thus justifying the utility of this component.

forall *media session* m **do**
  $Conf_i^m$: confidence value of the i-th comment session prediction for this media session;
  n: number of total comment session prediction in the confidence history;
  $Avg^m_{confidence} = \frac{\sum_{i=1}^{n} Conf_i^m}{n}$;
  if $Avg^m_{confidence} \geq 0.2$ *and current priority is* LOW **then**
    set current Priority to $HIGH$;
    continue;
  end
  if $Avg^m_{confidence} < 0.2$ *and current priority is* $HIGH$ **then**
    set current Priority to LOW;
    continue;
  end
end

Algorithm 2: SettingPriority()

3.3 An Example

Consider the example shown in Figure 2, where M1 and M2 represent newly created media sessions from Vine in real time. Three separate queues, Q1, Q2 and Q3 are maintained. The scheduler schedules media sessions in queue Q1 (pointed to by current head) for processing one by one in the queue order. After a media session has been processed by the cyberbullying detector component and if no alert is raised, it is placed at the end of either queue Q2 or queue Q3 depending on the new priority assigned to it (discussed in the next subsection). If the session's new priority is high, it is placed in queue Q2, and if the session's new priority is low, it is placed in queue Q3. When all media sessions in queue Q1 have been processed, queue Q2 becomes queue Q1, queue Q3 becomes queue Q2, and queue Q3 becomes empty. In the example shown in Figure 2, initially M1 and M2 are assigned high

priority and placed in queue Q1. M1 is scheduled first and is processed by the cyberbullying detector component. After this processing, it is assigned a low priority, and so is added at the end of queue Q3. M2 is scheduled next and is processed next. After this processing, it is assigned a high priority, and so is added at the end of queue Q2. At this time queue Q1 is empty, and queue Q2 becomes queue Q1 and queue Q3 becomes queue Q2. This process then continues. Since lower priority processes are eventually elevated into higher level queues, our solution ensures that no media session will starve. From this example, the low priority media session M1 is processed when the queue Q3 becomes Q1, thus making sure the low priority media sessions are also processed, albeit less frequently than its high priority counterpart.

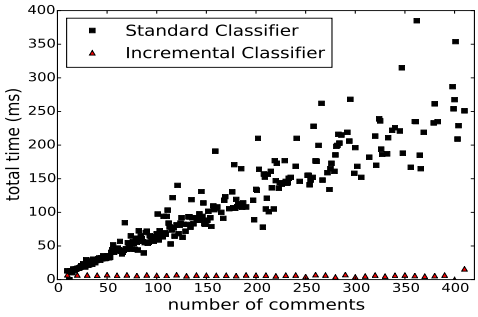


Figure 3: Total time taken by standard and incremental classifiers as new comments come in per media session.

4 PERFORMANCE EVALUATION

4.1 Incremental Classifier Evaluation

In [23], the AdaBoost classifier was reported to have the best performance based on accuracy, precision and recall values. Table 1 compares AdaBoost with logistic regression in terms of precision, recall,F-1 score and running time. The features AdaBoost classifier used were number of followers and followings, likes and views for media sessions, media caption polarity and subjectivity [21], total number of negative comments, summation of negative comment polarity and subjectivity, total individual comment polarity, total individual comment subjectivity, total negative words, total number of negative comments and unigrams. For Logistic Regression, the features we used were number of followers, followings, media caption polarity and subjectivity,total individual comment polarity, total individual comment subjectivity, total negative words, and total number of negative comments. We made sure that the features used by logistic regression were incrementally linear by nature, as noted in Section 3.1. The performance values showed in Table 1 were obtained using 10-fold cross validation on the labeled Vine data available in [23]. We notice that although the Adaboost classifier achieves a slightly higher precision, logistic regression achieves higher recall and F-1 score. Furthermore, the running time of logistic regression classifier is significantly less, more than five times faster than that of the Adaboost classifier. The reason for this is twofold. First, Adaboost needed unigram features to achieve a high precision and recall, but unigram feature extraction is computationally expensive. In comparison, logistic regression is able to achieve effectively the same precision and better recall while using features that are much more lightweight to compute, thus yielding much lower running time. Second, the Adaboost classifier is a meta-estimator that begins by fitting a classifier on the original data-set and then fits additional copies of the classifier on the same data-set but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases [25], thus making it computationally expensive compared to much simpler logistic regression. Based on these analyses, we have chosen the logistic regression classifier for detecting cyberbullying in our solution. **It is worth mentioning that we have employed other classifiers based on different combinations of features too (Decision Tree, Random Forest, Naive Bayes, Perceptron etc)** and only present the classifiers and feature combinations that yielded the best results.

Next, we show that leveraging incremental approach into our logistic regression classifier significantly improves the scalability of the execution stage. To this aim, we defined a baseline solution as consisting of non-incremental feature extraction and a non-incremental logistic regression classifier. As new comments arrive for the baseline solution, it would need to recompute all feature vectors from scratch, and recompute the entire logistic regression from scratch. We compared the total running time of the baseline solution with an incremental solution that implemented both incremental feature extraction and incremental logistic regression, as described in Section 3.1. Our measurements showed that the fraction of time taken by the logistic regression compared with the feature extraction time was negligible, so that the total running time was dominated by feature extraction.

Figure 3 shows the average time taken for the standard and the incremental classifiers as the number of comments increases in media sessions. To simplify the plot, we group the comments in sets of 10. The time taken by the standard classification solution goes up almost linearly with the number of comments in the media session, since the standard solution must recompute all features and regression weights. On the other hand, the time taken for the incremental classifier is basically constant every time a set of 10 additional comments come in because it only has to compute the feature values for the additional 10 new comments. The justification for using 10 comments-set is given in Section 4.2.

4.2 Dynamic Priority Scheduler Evaluation

In this section, we compare performances of our DPS with round robin scheduler, a scheduler with no assignment of priority. The aim of performing this comparison is twofold. First, we want to show the gain of responsiveness our scheduler achieve over round robin scheduler. Second, we also want to investigate several design choices crucial to building our

Table 1: Comparison of different classifiers using the 983 Labeled Media Sessions

Classifier	Precision	Recall	F-1 Score	Time (s)	Features
AdaBoost	0.7138	0.54	0.61	228	number of followers,followings, likes,views, media-caption polarity,subjectivity[21], total number of negative comments, summation of negative comment polarity and subjectivity, total individual comment polarity, total individual comment subjectivity, total negative words, total number of negative comments and unigrams
Logistic Regression	0.71	0.66	0.68	44.42	number of followers, followings, media caption polarity and subjectivity,total individual comment polarity, total individual comment subjectivity, total negative words, and total number of negative comments.

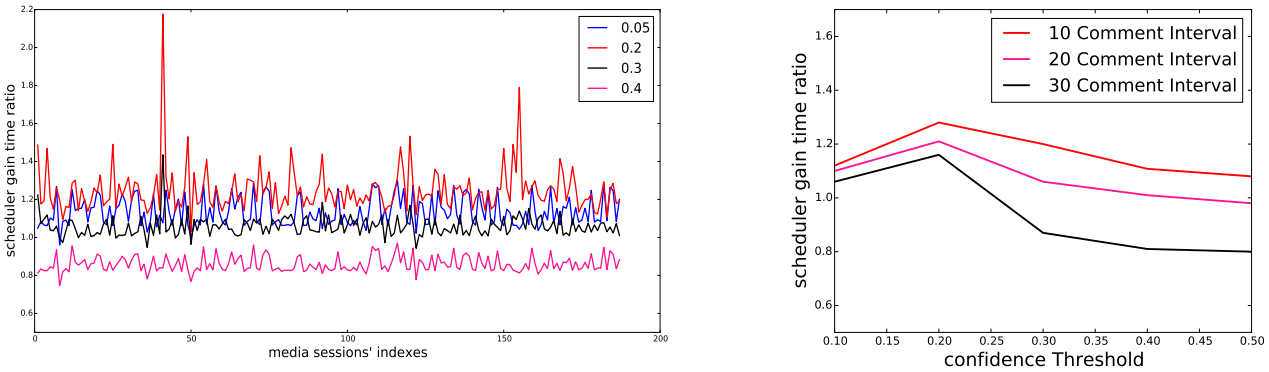


Figure 4: Left: Scheduler gain time ratio for different confidence thresholds for labeled cyberbullying media sessions. Right: Scheduler gain time ratio for different confidence thresholds by using different comment increment sizes for different number of media sessions using Vine labeled data [23].

scheduler to decide upon the best choice based on the metrics of performance gain each design choice achieves over round robin scheduler. We use labeled Vine dataset [23] to perform the experiments.

First, we need to determine what threshold is appropriate for our solution, as noted in 3.2. We also need to determine with what granularity our classifier ingests batches of new comments, because this affects the time to first alert. The scheduler will choose a high priority media session to pass to the classifier. In the time between classification attempts, a media session may receive N new comments. If all N comments are input to the classifier at once, and N is quite large, we may delay recognizing cyberbullying, i.e., a burst of negative comments may be swamped by the other positive comments. Therefore, we need to consider comments in small enough batches or intervals so that the classifier can catch cyberbullying with finer granularity and raise the alert early.

The left figure in Figure 4 assesses which combination of threshold and interval size produced the best improvement in response time using dynamic prioritization compared with a simple round-robin policy. The round-robin scheduler is defined as one where media sessions are not assigned any priority, and the scheduler simply rotates through all media sessions, with no particular attention being paid to likely cyberbullying sessions.

As can be seen from the figure, **by using a confidence threshold of** 0.2 **and comment increment size of** 10**, we were able to gain the maximum responsiveness over the round-robin scheduler**. We also like to note here that that, when the number of new comments for a media session is less than 10, we just take those 5 available new comments. We think this is because as the comment increment size goes up to 20 or 30, the burst of cyberbullying comments can get nullified by the other positive comments, which in turn influences the features(i.e. summation of individual comment sentiments) that are used by our incremental classifier. So 10 comment increment size tends to be the optimal size for having enough context of a comment thread to make a knowledgeable decision about cyberbullying while also not being too big to risk being nullified by other positive comments. The figure also confirms that the confidence threshold of 0.2 offers the best speedup for our dynamic priority scheduler. For example, if a media session m has been classified 3 times at t_1, t_2, t_3 with classification decisions not-bullying, not-bullying, not-bullying respectively

Table 2: When to Send Alert

Number of Predicted Cyberbullying Comment Sessions in the History	Precision	Recall	F-1 Score
≥ 1	0.68	0.71	0.69
≥ 2	**0.71**	**0.71**	**0.71**
≥ 3	0.71	0.71	**0.71**

with confidence values of $0.85, 0, 85, 0.55$, this means even though it has been classified as not cyberbullying, the confidence values of cyberbullying decision is also increasing $(0.15, 0.15, 0.45)$ which makes it a potential candidate for a future cyberbullying session. So we take the average of the previous classification confidence values of cyberbullying class (0.25 in this case) and see that the average confidence value is more than 0.2 and change the priority of this media session as high and insert it in the dynamic priority scheduler. We mention that we tried all possible combinations of comment chunk sizes and confidence thresholds and only present those combinations that yielded the best results. The right figure in Figure 4 demonstrates the gain time achieved by our scheduler for each media sessions from the Vine dataset labeled as cyberbullying. This figure further justifies the choice of confidence threshold of 0.2 in our scheduler. These experiments helped us to not only justify our choice of using DPS but also helped us to decide upon the crucial design choices of using confidence threshold of 0.2 and 10 comment increment size.

4.3 Alert Performance

Since each media session will be passed sporadically to the classifier by the scheduler, the classifier will generate a sequence of cyberbullying detection decisions over time for each media session. It is therefore worth considering to what extent we should utilize the history of detection decisions in generating the alert. The default is to generate an alert immediately after the classifier decides that the current batch of 10 comments, in combination with earlier content, constitutes cyberbullying. However, we wish to be sure and avoid false positives. One option is to decouple the alert from the classification, and delay the alert until N positive decisions have been recently seen. This design gives us some flexibility in trading off responsiveness and precision.

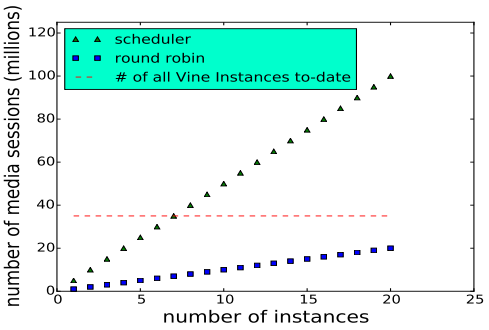
For each media session, we maintain an array storing the results of each classification result of that session along with the time of that classification. We use this array to decide when to raise an alert. In particular, we set a threshold value, which is the number of times a media session has been classified as cyberbullying since the last time an alert was raised for that session, or from the beginning if no alert has yet been raised. After experimenting with different number of threshold values(2), we find that by raising an alert only when we have at least 2 decisions for cyberbullying since the last time an alert was raised, we achieve the best precision,recall and F-1 score of of 0.71,0.71 and 0.71 respectively, thus reducing the number of false alarms. This performance is a marked improvement over the Static Priority Scheduler (SPS) described in Section 3.2 that had a recall of only 58%. Moreover, the recall is, in fact, an improvement over the standard classifier's 0.66 (See Table 1 for comparison). This marked improvement of recall over two baselines (SPS and standard classifier) demonstrates the justification of using incremental classifier component and dynamic priority scheduler along all the associated design choices: that these components are efficient and responsive and also retains sufficient classifier performance when compared to current state-of-the-art.

4.4 Scalability Evaluation

In this section, we first demonstrate the way our proposed system scales when it has to deal with a substantial number of media sessions. For this purpose, we first deploy Amazon AWS free tier 1GB memory virtual machine instances to start monitoring media sessions, implementing both our proposed scheduler and round robin scheduler. An acceptable responsiveness of the system is our primary goal along with the scalability of the system. In these experiments, we decided that an average alert time under 2 hours is acceptable, which means an alert will be within 2 hours of a cyberbullying instance. We acknowledge that this decision is purely because of the lack of research in this particular area. In future, we will conduct an elaborate survey to explore the acceptability of a potential cyberbullying system's responsiveness. For the experiments presented in this section, we replicated the 100000 media sessions' traffic from the dataset in [23] up to the scale of 39 million. Those media sessions were gathered by performing snowball sampling after selecting a random seed. We believe the randomness of the seed selection, snowball sampling, and large number (100000) of media sessions in this dataset should enable the scaled up traffic to reasonably approximate the behavior of the overall network.

The left graph in Figure 5 shows the number of media sessions that can be processed as the number of AWS instances is increased, keeping the average alert time under 2 hours. This figure shows that the number of media sessions that can be processed increases linearly with the increase in the number of instances, and that our system scales *five times better* than the round robin scheduler. Given that the Vine social network generated about 39 million media sessions since 2012 [26], our system is capable of monitoring Vine-scale social networks with only 8 AWS instances, keeping the average alert time below 2 hours. In contrast, a round-robin scheduler would require upwards of 40 instances.

The right graph in Figure 5 shows the average alert time for round robin and dynamic priority scheduler versus the number of media sessions. It can be seen from the figure that we are able to process 5 million media sessions with our proposed system with an average alert time under 2 hours where as for round robin, it is 1 million. To show the cost of using our dynamic priority scheduler in terms of worst case scenario, we see in Figure 6 that around 10 percent of media



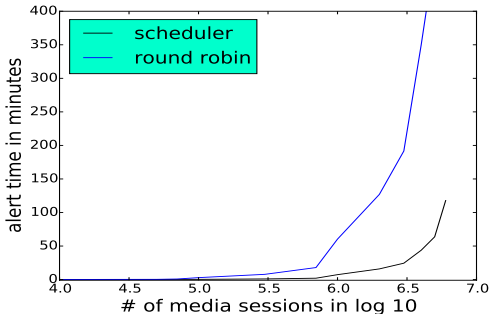


Figure 5: Left: Number of media sessions vs number of instances needed to monitor them, keeping average alert time under 2 hours. Right: Average alert time vs number of media sessions for round-robin and dynamic priority scheduler.

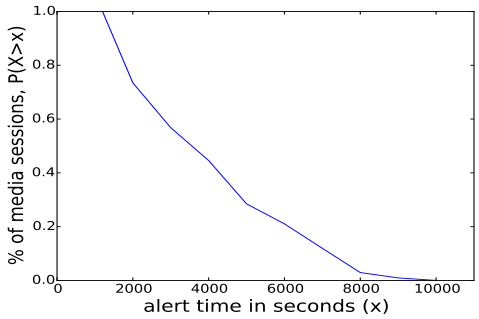


Figure 6: CCDF of alert time for 5 million media sessions in 1GB memory amazon AWS instance using dynamic priority scheduler.

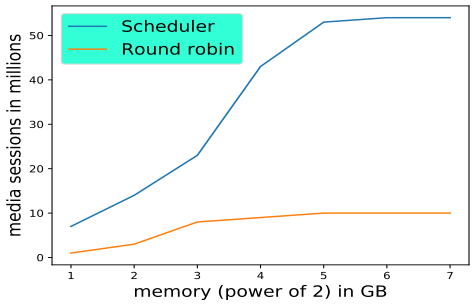


Figure 7: Memory vs number of media sessions in millions.

sessions get their alerts after 2 hours. This is the cost we pay for postponing the processing of low priority media sessions in our scheduler.

Next,we investigate the resources our system will need to monitor a comparatively more popular social network like Instagram. Since its establishment in 2010, Instagram has accumulated over 40 billion media sessions [18], which means almost 6 billion media sessions per year, a number much larger than Vine's 39 million [26]. To accommodate

Table 3: Total Time Comparison for Different Approaches and Different Number of Media Sessions (seconds)

Approach	10000	50000	100000
AdaBoost	5674	26784	-
Logistic Regression	1110	5320 (44X)	10438
Incremental Classifier	22	120 (223X)	206

such a high volume social network, based on Figure 5, we would need 1,200 AWS instances to keep the average alert time under 2 hours every year. To scale up for such a load, we have two choices. We can either spawn off 1,200 such instances of 1 GB memory or we can increase the memory of our instances to process more media sessions. To further investigate the memory performance, we implemented our system with different memory-sized instances belonging to Amazon AWS services. Figure 7 shows the number of media sessions processed by each instance of a particular memory size. The number of media sessions in the Y axis illustrates the highest number of media sessions that can be processed by that instance without having an average alert time of over two hours. The figure demonstrates that, while increasing memory does help increase the media session monitoring capacity, at a certain point around 32 GB/instance, additional memory no longer enables additional monitoring of media sessions. That is, the graph plateaus around 50 million media sessions, so that there is no additional benefit to using 64 GB or 128 GB instances compared to 32 GB instances. We hypothesize that this behavior is due to computation becoming the main bottleneck rather than memory. Therefore, to monitor Instagram-scale social networks, we would need approximately 120 32 GB instances. Note that without our dynamically scheduled incremental classification system, approximately 600 32 GB instances would be required, almost five times as many, as it can be seen from Figure 7.

For evaluating the incremental classifier's scaling performance, we compare three types of approaches, namely the best reported Vine cyberbullying classifier [23] (Standard AdaBoost), Logistic regression without incremental feature

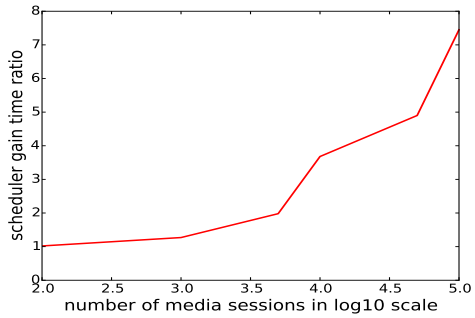


Figure 8: Gain time ratio (round-robin scheduler/dynamic priority scheduler)for different number of media sessions.

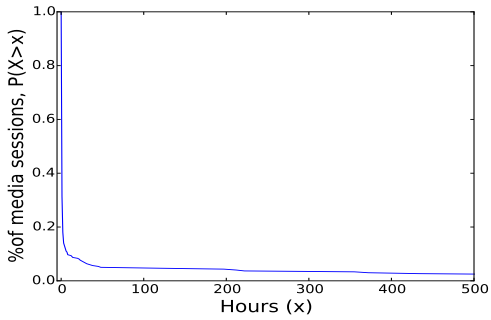


Figure 9: CCDF of Time Interval in hours until First Comment For Bullying Media Sessions.

extraction or classification (Standard Logistic regression), and Logistic regression with incremental feature extraction and classification (Incremental Classifier). Table 3 shows the time needed in seconds for these three approaches to process different numbers of media sessions. The table clearly demonstrates that the choice of using incremental classifier helped us to improve classification time by 223 times faster than AdaBoost for 50,000 media sessions and 44 times faster than the standard logistic regression. For this evaluation purpose, we used the Vine dataset provided in [23].

Next, we compare the responsiveness of our dynamic priority scheduler against the unprioritized round robin scheduler. The metric we use to compare these two approaches is responsiveness gain, meaning the ratio of time taken by the round-robin scheduler over the time taken by our dynamic priority scheduler to raise an alert. Figure 8 shows the responsiveness gain vs. number of media sessions. The gain tends to increase as the number of media sessions goes up, reaching almost 7 times faster responsiveness for $100,000$ media sessions. This improvement is due to the fact that our scheduler tends to process cyberbullying media sessions first whereas round robin processes all media sessions at each pass. For this reason, as the number of media sessions grows, so does the improvement of using our dynamic priority scheduler due to its priority processing of cyberbullying media sessions.

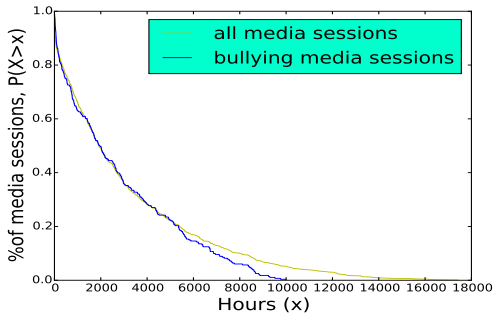


Figure 10: CCDF of activity for all and bullying media sessions.

For further insights into resource scalability, we present activity graphs of the media sessions from Vine. We investigate the distribution of how long a bullying media session takes to receive its first comment. Figure 9 shows that very few bullying media sessions receive their first comment after 500 hours since session creation. Hence, one way to improve scaling is to stop monitoring any session that takes longer than 500 hours for its first comment. Secondly, Figure 10 shows the CCDF of activity of media sessions in Vine. A fair percentage of media sessions receive comments even after 10000 hours after initial media posting. In comparison, bullying media sessions received all their comments within 9000 hours, i.e. within a year of their creation. So another way to improve scaling would be to purge out all media sessions that are one year old.

5 CONCLUSION AND FUTURE WORK

In this work, we have developed a cyberbullying detection system for media-based social networks, consisting of a dynamic priority scheduler, a novel incremental classifier, and an initial predictor. The evaluation results show that our system substantially improves the scalability of cyberbullying detection compared to an unprioritized system. Morever, we demonstrate that our system can fully monitor Vine-scale social networks for cyberbullying detection for a year using only eight 1 GB AWS VM instances. We discover the point (32 GB) at which adding memory no longer enables monitoring of more media sessions, and project that our system would need 120 32 GB instances to fully monitor Instagram-scale traffic for cyberbullying.

As part of future work, we propose to investigate the plateauing effect that limits the effectiveness of adding more memory, namely that there is likely a computational bottleneck that needs to be further addressed. We also plan to research on developing a diverse,scalable and responsive classifier with better precision and recall performances for Vine as well other diverse OSNs like Facebook.

REFERENCES

[1] Ryan Broderick. 2014. 9 Teenage Suicides In The Last Year Were Linked To Cyber-Bullying On Social Network Ask.fm. http://www.buzzfeed.com/ryanhatesthis/a-ninth-teenager-since-last-september-has-committed-suicide. (2014). [Online;accessed 14-January-2014].

[2] Probability Calibration. 2017. http://scikit-learn.org/stable/modules/calibration.html. (2017). [Online; accessed September, 2017].

[3] Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Emiliano De Cristofaro, Gianluca Stringhini, and Athena Vakali. 2017. Detecting Aggressors and Bullies on Twitter. In *Proceedings of the 26th International Conference on World Wide Web Companion (WWW '17 Companion)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 767–768. https://doi.org/10.1145/3041021.3054211

[4] Hanqiang Cheng, Yu-Li Liang, Xinyu Xing, Xue Liu, Richard Han, Qin Lv, and Shivakant Mishra. 2012. Efficient Misbehaving User Detection in Online Video Chat Services. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining (WSDM '12)*. ACM, New York, NY, USA, 23–32. https://doi.org/10.1145/2124295.2124301

[5] National Crime Prevention Council. 2007. Teens and Cyberbullying. (2007). Executive summary of a report on research conducted for National Crime Prevention Council.

[6] Karthik Dinakar, Roi Reichart, and Henry Lieberman. 2011. Modeling the detection of Textual Cyberbullying. (2011).

[7] Linda DiProperzio. 2015. Cyberbullying Applications. http://www.parents.com/kids/safety/internet/best-apps-prevent-cyberbullying/. (2015). [Online; accessed February 6, 2015.].

[8] Russel Goldman. 2014. Teens Indicted After Allegedly Taunting Girl Who Hanged Herself. http://abcnews.go.com/Technology/TheLaw/teens-charged-bullying-mass-girl-kill/story?id=10231357. (2014). [Online; accessed 14-January-2014].

[9] Matthew A. Hammer, Joshua Dunfield, Kyle Headley, Nicholas Labich, Jeffrey S. Foster, Michael Hicks, and David Van Horn. 2015. Incremental Computation with Names. In *Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2015)*. ACM, New York, NY, USA, 748–766. https://doi.org/10.1145/2814270.2814305

[10] Matthew A. Hammer, Khoo Yit Phang, Michael Hicks, and Jeffrey S. Foster. 2014. Adapton: Composable, Demand-driven Incremental Computation. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '14)*. ACM, New York, NY, USA, 156–166. https://doi.org/10.1145/2594291.2594324

[11] Keith Harrison. 2012. *Scalable Detection of Community Cyber Incidents Utilizing Distributed and Anonymous Security Information Sharing*. Ph.D. Dissertation. UTSA. Advisor(s) White, Gregory. AAI3548643.

[12] Homa Hosseinmardi, Amir Ghasemianlangroodi, Richard Han, Qin Lv, and Shivakant Mishra. 2014. Towards Understanding Cyberbullying Behavior in a Semi-Anonymous Social Network. In *ASONAM'14*. IEEE, Beijing,China, 244–252.

[13] Homa Hosseinmardi, Sabrina Arredondo Mattson, Rahat Ibn Rafiq, Richard Han, Qin Lv, and Shivakant Mishra. 2015. *Analyzing Labeled Cyberbullying Incidents on the Instagram Social Network*. Springer, Cham, 49–66. https://doi.org/10.1007/978-3-319-27433-1_4

[14] Loc N. Huynh, Youngki Lee, and Rajesh Krishna Balan. 2017. DeepMon: Mobile GPU-based Deep Learning Framework for Continuous Vision Applications. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '17)*. ACM, New York, NY, USA, 82–95. https://doi.org/10.1145/3081333.3081360

[15] A. Kontostathis K. Reynolds and L. Edwards. 2011. Using Machine Learning to Detect Cyberbullying. *ICMLA'11* 2 (2011), 241–244. https://doi.org/10.1109/ICMLA.2011.152

[16] Robin M. Kowalski, Sue Limber, Susan P. Limber, and Patricia W. Agatston. 2012. *Cyberbullying: Bullying in the digital age*. John Wiley & Sons, Reading, MA.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* 60, 6 (May 2017), 84–90. https://doi.org/10.1145/3065386

[18] Evan Lepage. 2015. Instagram Statistics. http://blog.hootsuite.com/instagram-statistics-for-business/. (2015). [Online; accessed February 6, 2015.].

[19] H. H. S. Li, Z. Yang, Q. Lv, R. I. R. R. Han, and S. Mishra. 2014. A Comparison of Common Users across Instagram and Ask.fm to Better Understand Cyberbullying. In *IEEE BDCloud'14*. IEEE, Sydney, Australia, 355–362. https://doi.org/10.1109/BDCloud.2014.87

[20] Rui Li, Kin Hou Lei, Ravi Khadiwala, and Kevin Chen-Chuan Chang. 2012. TEDAS: A Twitter-based Event Detection and Analysis System. In *ICDE '12*. IEEE, Washington, DC, USA, 1273–1276. https://doi.org/10.1109/ICDE.2012.125

[21] Steven Loria. 2016. Python Sentiment Library. https://github.com/sloria/textblob. (2016). [Online; accessed May 30, 2016].

[22] M. Ptaszynski, P. Dybala, T. Matsuba, F. Masui, R. Rzepka, K. Araki, and Y. Momouchi. 2010. In the service of online order Tackling cyberbullying with machine learning and affect analysis. *International Journal of Computational Linguistics Research* 1, 3 (2010), 135–154.

[23] Rahat Ibn Rafiq, Homa Hosseinmardi, Richard Han, Qin Lv, Shivakant Mishra, and Sabrina Arredondo Mattson. 2015. Careful what you share in six seconds: detecting cyberbullying instances in vine. In *ASONAM*. ACM, Paris,France, 617–622.

[24] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. ACM, New York, NY, USA, 851–860. https://doi.org/10.1145/1772690.1772777

[25] Python scikit learn. 2016. AdaBoost Classifier. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html. (2016).

[26] Craig Smith. 2015. Vine Statistics. http://expandedramblings.com/index.php/vine-statistics/. (2015). [Online; accessed February 6, 2015.].

[27] Laura Smith-Spark. 2014. Hanna Smith suicide fuels calls for action on Ask.fm cyberbullying. http://www.cnn.com/2013/08/07/world/europe/uk-social-media-bullying/. (2014). [Online;accessed 14-January-2014].

[28] Francesca Spezzano. 2016. Bad Actors in Social Media. In *Proceedings of the First International Workshop on Computational Methods for CyberSafety (CyberSafety'16)*. ACM, New York, NY, USA, 1–1. https://doi.org/10.1145/3002137.3002138

[29] StopCyberBullying.org. 2017. 5 Differences between Cyber Bullying and Traditional Bullying. http://onlinesense.org/5-differences-cyber-bullying-traditional-bullying/. (2017). [Online; accessed May 22, 2017.].

[30] Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes, Bart Desmet, Guy De Pauw, Walter Daelemans, and Veronique Hoste. 2015. Automatic detection and prevention of cyberbullying. In *International Conference on Human and Social Analytics, Proceedings*, Pascal Lorenz and Christian Bourret (Eds.). IARIA, Saint Julians, Malta, 13–18.

[31] Wiki. 2016. logistic Regression. https://en.wikipedia.org/wiki/Logistic_regression. (2016). [Online; accessed May 30, 2016].

[32] Xinyu Xing, Yu-li Liang, Sui Huang, Hanqiang Cheng, Richard Han, Qin Lv, Xue Liu, Shivakant Mishra, and Yi Zhu. 2012. Scalable Misbehavior Detection in Onlline Video Chat Services. In *KDD*. ACM, NY, USA, 552–560.