Catching them red-handed: Real-time Aggression Detection on Social Media

Herodotos Herodotou
Cyprus University of Technology
Limassol, Cyprus
herodotos.herodotou@cut.ac.cy

Despoina Chatzakou
Centre for Research and Technology Hellas
Thessaloniki, Greece
dchatzakou@iti.gr

Nicolas Kourtellis
Telefonica Research
Barcelona, Spain
nicolas.kourtellis@telefonica.com

***Abstract*—Aggression on social media has evolved into a major point of concern. However, recently proposed machine learning (ML) approaches to detect various types of aggressive behavior fall short, due to the fast and increasing pace of content generation as well as evolution of such behavior over time. This work introduces the first, practical, real-time framework for detecting aggression on Twitter via embracing the streaming ML paradigm. This method adapts its ML binary classifiers in an incremental fashion, while receiving new annotated examples, and achieves similar performance as batch-based ML models, with 82–93% accuracy, precision, and recall. Experimental analysis on real Twitter data reveals how this framework, implemented in Spark Streaming, easily scales to process millions of tweets in minutes.**

***Index Terms*—online aggression detection, streaming machine learning, social media**

I. Introduction

In recent years, aggression has spiked across the Web, with abusive behavior incidents being reported in different contexts[1]. Such behavior can manifest in different platforms, independently of each service's utility and target audience. In particular, aggressive, abusive, offensive, bullying, racist, or sexist behavior has been studied in different media platforms such as Twitter [1], [2], Instagram [3], YouTube [4], Yahoo Finance [5], Yahoo Answers [6], 4chan [7] and across demographics (gender and age) [8]. Due to the regular negative publicity this problem receives in the news, popular platforms have tried to curb it by deploying new methods and features. For example, Twitter[2] and Instagram[3] have recently proposed new features that aim to limit or block the ability of aggressors to bully by posting messages on their victims' pages, profiles, or walls. However, even with such efforts in place, the problem still persists due to its following aspects.

First, many of these platforms enable users to produce and engage with a fast-paced generated content across different topics, thus, facilitating an exponential growth of user-contributed posts: 2.5T Facebook posts to date and 4.2B Instagram "likes" per day[4], 778M tweets per day[5], etc. Therefore, platforms have a constantly increasing workload to process for detecting abusive behavior. Second, users (and especially millennials, Generation Z, and Alpha) adapt quickly to new platform community rules or algorithms rolled out to detect such behavior. In fact, they typically find "innovative" ways to circumvent rules to show aggression, using new words or special characters to signify aggression but avoid detection [9]. Third, this complex behavior is difficult to model, as it highly depends on context, cultural background, nature of interaction, platform features, scope, etc. Finally, detecting this behavior is a minority class problem, making it harder to address effectively with automated systems: any given random data set has few examples to train classifiers on.

Unfortunately, methods proposed by industry and academia to detect such behavior have not adequately addressed these issues. As most current ML approaches used are batch-based, a trained model can become deprecated after some time [10]. Also, many approaches are computationally expensive for both training and testing [11], [12]. Thus, there is urgent need for new, real-time methods that: 1) *adapt* their ML modeling in a similarly fast rate as content is generated, keeping models up-to-date with users' adaptability, and 2) *scale* to process the increasing workload from generated content in real time.

We address this gap by making the following **contributions**:

- We propose the *first, practical, and real-time framework for aggression detection on Twitter*. Departing from past works, we embrace the streaming modeling paradigm, and adapt our ML binary classifiers incrementally, as new annotated examples are received through time.
- The framework is designed to be extensible and accurate while providing online alerts to moderators, using state-of-the-art streaming methods: Hoeffding Trees, Adaptive Random Forests, and Streaming Logistic Regression.
- The framework is *deployable on state-of-the-art streaming engines* such as Apache Spark (Section III).
- We *demonstrate its effectiveness* to detect aggressive behavior at real time on a large set of $86k$ tweets, annotated for this context (Section IV). Results from the streaming methods are similar to typical batch-based ML methods, with 82–93% performance in multiple metrics (accuracy, precision, recall; Section V).
- This Spark-implemented framework *easily scales out* to process millions of tweets in minutes, compared to MOA, a popular centralized streaming ML engine.

[1]https://cyberbullying.org/facts and http://bit.ly/2QP1yi9
[2]https://www.bbc.com/news/business-51044086
[3]https://about.instagram.com/blog/announcements/stand-up-against-bullying-with-restrict
[4]https://www.socialpilot.co/blog/social-media-statistics
[5]https://www.internetlivestats.com/one-second/

II. Related Work

Under the umbrella of "abuse", various abnormal behaviors may be involved, such as offensive language and hate speech, bullying and aggression, and sexual exploitation. For an efficient detection of abusive behavior, ML-based approaches are among the most commonly considered. Many earlier works used traditional ML classifiers, such as logistic regression [11], [13], support vector machines [14], or ensemble classifiers [15]. E.g., authors in [11] used a ML-based method to perform hate speech detection on Yahoo Finance and News data, while authors in [15] used a combination of probabilistic, rule- and spatial-based classifiers with a vote ensemble meta-classifier to study the spread of online hate speech on Twitter.

More recently, deep learning architectures have also been used in an effort to improve the detection efficiency. For instance, in [16] various deep learning architectures were assessed, including Convolutional Neural Networks (CNNs), Long Short-Term Memory Networks (LSTMs), and FastText [17]. In the same direction, in [12] a CNN with Gated Recurrent Unit network was used, combined with word embeddings, to detect hate speech on Twitter. Finally, for the detection of abusive and aggressive behaviors, authors in [2] experimented with both traditional machine learning methods (i.e., probabilistic, tree-based, and ensemble classifiers), as well as deep neural networks (Recurrent Neural Networks).

Most of the previous work has focused on detecting abusive behaviors on a "batch mode". Namely, the goal was: given a set of data, to train an ML classifier to identify as effectively as possible different types of abusive behaviors. To increase the accuracy of the detection process, some of the proposed approaches are cost-ineffective in terms of the time needed for both training and testing purposes. But as we live in an ever-evolving world, it is of the utmost importance to develop methods that can support the constant monitoring of content produced in online sources, so that the detection of abusive behaviors can be carried out in its infancy. To this end, authors in [10] proposed a multi-stage cyberbullying detection solution for Vine comments to reduce the classification time and the time to raise alerts; an "incremental computation" approach was followed at both the feature extraction and classification stages to lower the computational complexity. Moreover, in [3] a two-stage online framework is proposed for cyberbullying detection on Instagram, where comments are examined as they become available. To ensure the scalability of the detection process, instead of constantly using the same set of features, an online feature selection approach was followed.

Contributions. To our knowledge, this work is the first to apply the streaming ML modeling paradigm for detecting aggressive behavior on social media. Our framework is able to adapt the ML models quickly and efficiently as content becomes available, while also being able to scale and process an increasing workload in real time. Finally, contrary to existing methods, our framework is able to detect aggression on individual tweets (and not at a session-level).

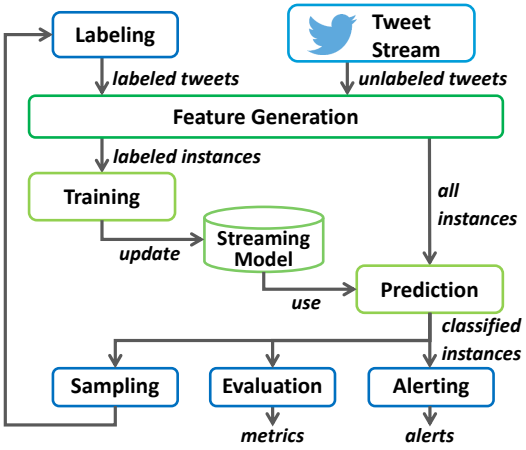


Fig. 1: Processing pipeline for detecting aggression on Twitter.

III. System Overview

The proposed framework for detecting aggression is characterized by two key features: data streaming and parallelization. The former enables the framework to dynamically update the classification model as soon as a new labeled tweet arrives and to detect aggressive tweets in real time. The latter is essential for scaling the proposed approach in a distributed cluster in order to handle the ever-increasing volume of tweets without affecting the classification performance.

A. Processing Pipeline for Aggression Detection

The overall processing pipeline for detecting aggressive behavior on Twitter is shown in Figure 1 and includes the following steps: (1) feature generation, (2) training, (3) prediction, (4) alerting, (5) evaluation, (6) sampling, and (7) labeling. The data input comprises two streams with labeled and unlabeled tweets, while the output consists of multiple streams with alerts of aggressive tweets, evaluation metrics, and sample tweets for labeling.

Input streams. The Twitter Streaming API gives access to a stream of tweets in JSON format, each containing information about the tweet (e.g., content, timestamp) and the user who posted the tweet (e.g., name, number of followers). The stream of labeled tweets originates from labeling (discussed below) and uses the same JSON format plus an attribute with the class label (normal or aggressive; see Section IV-A).

Feature generation. The first step begins with the typical processing and cleaning, such as removing punctuation marks, special symbols, numbers, URLs, hashtags, and user mentions. Next, a set of user profile, text, and social network features are extracted (described in Section IV-B) and normalized to fall within a predefined range, such as between 0 and 1.

Training. This step consumes the stream of labeled feature instances and incrementally updates the streaming classification model. Each instance is processed only once and then it is discarded. Hence, training in a streaming setting is much more efficient than in batch processing, which requires collecting a large training set first and then processes it multiple times in an iterative fashion. Another benefit of stream-based learning

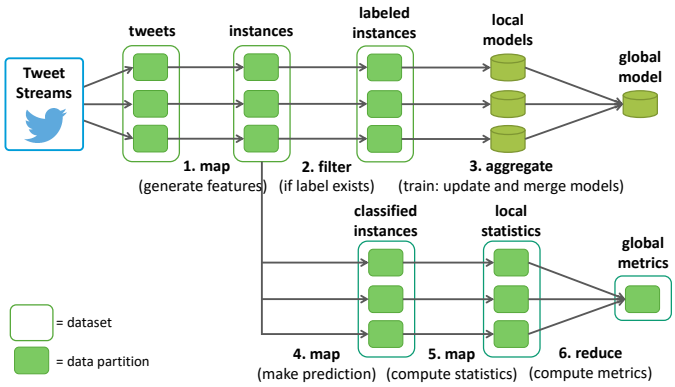


Fig. 2: Parallel data processing for feature generation, training, prediction, and evaluation in Apache Spark Streaming.

is that in naturally adapts to data distribution changes (i.e., concept drift), keeping the streaming model always up-to-date.

Prediction. The streaming model is used in this step to predict the class label of each tweet instance. Prediction is performed on unlabeled instances in order to detect aggressive behavior and raise appropriate alerts, while it is performed on labeled instances for evaluating the classification performance of the model by comparing the predicted labels with the true ones.

Alerting. This step raises an alert in real time whenever an aggressive tweet is detected. Multiple options exist for how to handle these alerts: (a) forward the tweet to a human moderator for review, (b) post an automatic warning on the tweet, or (c) remove the tweet automatically.

Evaluation. Classified labeled instances are used for computing important metrics (e.g., confusion matrix, accuracy, precision, F1-score) and evaluating the performance of the streaming classification model. On the other hand, classified unlabeled instances are used for computing various interesting statistics such as the distribution of predicted class labels and how they change over time.

Sampling. The primary goal of sampling is to select a small subset of tweets for manual labeling. Given that aggressive tweets are a minority, random sampling is not advisable as it would result in an imbalanced dataset. A better alternative is to implement a boosted random sampling technique that uses the predicted label to boost a random sample with tweets that are likely to be aggressive (without biasing the sample) [18].

Labeling. This step can generate new labeled tweets over time for improving the performance of the streaming model. Labeling can be performed by expert moderators or via a crowdsourcing platform like CrowdFlower [18], and is beyond the scope of this paper.

B. Parallel Data Processing for Streaming ML

For our framework to be practical, it must be able to scale out to handle the massive volumes of tweets generated per day. Hence, we have implemented our framework on top of Spark Streaming[6], a popular distributed stream processing engine. In Spark Streaming, the input data streams are divided in micro-batches and each micro-batch is processed through a directed acyclic graph of operations. The vertices of the graph represent the datasets at various phases of the processing and the edges correspond to high-level transformations such as map, filter, and aggregate. Datasets are partitioned and transformations are executed in parallel across a cluster, leading to fast and scalable stream data processing.

Figure 2 shows the key datasets and transformations for performing feature generation, training, prediction, and evaluation in Spark Streaming. The input tweets for one micro-batch are randomly divided into the data partitions that form the "tweets" dataset. Next, each tweet is mapped (op #1) into an instance using the feature generation step described above. Instances with labels are then filtered (op #2), followed by the training step using an aggregate transformation (op #3), which is performed in two phases. In the first phase, the local models are incrementally updated in parallel, while in the second one, the local models are merged to update the global model. The map, filter, and the first aggregate phase are grouped together and executed as a set of parallel tasks. The updated global model (with a size of less than 1MB) is then distributed across the cluster and is available for use in the next micro-batch. Moreover, another set of parallel tasks perform the predictions (op #4) on all instances and compute local statistics (op # 5), such as the confusion matrix. Finally, the local statistics are reduced to compute (op # 6) global evaluation metrics such as accuracy and F1-score. The alerting and sampling steps consume the "classified instances" dataset and are also performed in parallel (not shown in Figure 2).

We integrated our framework with *streamDM*[7], a streaming ML library for Spark Streaming, and utilized three classification algorithms: (a) **Hoeffding Tree (HT)**, an incremental decision tree learner designed for large data streams [19]; (b) **Adaptive Random Forest (ARF)**, an adaptation of the classical Random Forest algorithm using HTs [20]; and (c) **Streaming Logistic Regression (SLR)**, an adaptation of the long-established logistic regression in a streaming setting.

IV. Datasets & Feature Extraction

A. Twitter Data

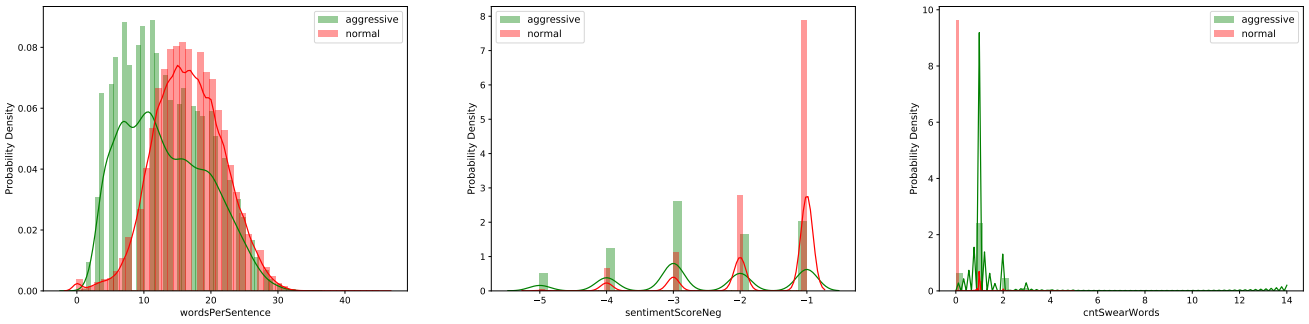
For our study, an abusive dataset provided by [18] is used, where tweets are characterized as *abusive*, *hateful*, *spam*, or *normal*. Overall, the dataset consists of $100k$ tweets. As our focus is on detecting aggressive behaviors in a streaming way, we remove the $14k$ tweets labeled as spam, as they can be dealt with more specialized techniques [21], while we consider abusive and hateful tweets as one class (aggressive).

B. Feature Extraction

A wide range of features can be considered to represent users' online presence and thus to detect the existence of abusive behavior. Such features can come from either the users' profile, posted content, or their social network. Authors

[6]https://spark.apache.org/streaming/

[7]http://huawei-noah.github.io/streamDM/



(a) Mean words per sentence. (b) Sentiment (negative). (c) Swear words.

Fig. 3: PDF (Probability Density Function) of (a) Mean words per sentence, (b) Sentiment (negative), and (c) Swear words.

in [2] studied a wide range of features falling in the aforementioned categories. Inspired by their analysis and after our own investigation, we selected the most contributing features, with the aim of reducing execution time without sacrificing performance. Indicatively, Figure 3 plots the PDF (Probability Density Function) of some of the features presented next.

Profile Features. This feature category includes the account's age (number of days since its creation), number of a user's posts, number of lists subscribed to. Overall, we observed that, on average (days), accounts that post normal content tend to be older ($\approx 1,487$) compared to the aggressive ones ($\approx 1,305$).

Text Features. This feature category includes some basic text features, syntactic and stylistic ones, the expressed sentiment, as well as the existence of swear words within posts.

- *Basic:* Number of hashtags, uppercase words, and URLs used in a post. We found that in the aggressive posts, users tend to write with higher frequency in capital letters, which is quite an expected result as uppercase text can be indicative of intense emotional state or 'shouting'. Specifically, the average (STD) number of uppercase words for the normal and aggressive posts are 0.96 (2.10) and 1.80 (3.23), respectively.
- *Syntactic:* We analyze the writing style of the authors of a text. Specifically, we focus on Part-of-Speech (POS) features, which depict the relative frequency of adjectives, adverbs, and verbs in a text. We found that aggressive posts tend to contain fewer adjectives than normal. Adjectives are generally used to describe nouns and pronouns, and thus their lower usage in aggressive posts indicates that users may prefer to carry out more direct attacks without adding additional information.
- *Stylistic:* Mean number of words per sentence and mean word length. Figure 3a depicts the PDF of the mean number of words per sentence. Normal posts seem to contain relatively more words compared to the aggressive ones (16.66 and 13.16 words on average, respectively). As for the mean word length, a similar pattern was observed for the normal and aggressive posts.
- *Sentiment:* To assess how positive or negative is the sentiment expressed in the posted content (on a $[-5, 5]$ scale), we use the SentiStrength tool[8]. From Figure 3b, we observe that normal posts include significantly less negative sentiments compared to the aggressive ones.
- *Swear Words:* To estimate the number of existing curse words within posts, we use a list of swear words obtained from AllSlang[9]. Figure 3c indicates that aggressive posts (avg., 1.03) contain a significantly higher number of swear words compared to the normal posts (avg., 0.04).

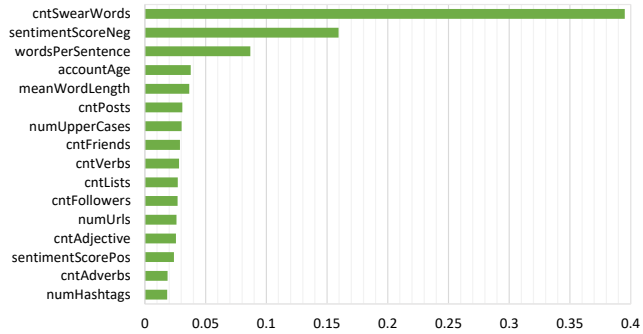


Fig. 4: Evaluation of features based on *Gini* importance.

Network Features. This feature category aims to measure a user's popularity based on two different criteria: the number of followers (*in-degree*) and friends (*out-degree*). Overall, these measures can quantify a user's opportunity to have a positive or negative impact in their ego-network in a direct way.

Features Evaluation. Figure 4 shows the feature relevance based on *Gini* importance. We see that the most important feature is the swear count, followed by negative sentiment, number of words per sentence, account age, mean word length, and count posts. Overall, we observe that the text based features are among the most contributing ones.

V. Experimental Evaluation

In this section, we evaluate the effectiveness and efficiency of our framework to correctly detect aggressive behavior in a large set of tweets. Our evaluation methodology is as follows:

1) We study the performance of three streaming ML methods (Section V-A).

[8]http://sentistrength.wlv.ac.uk/

[9]https://www.noswearing.com/dictionary

TABLE I: Hyperparameter tuning for streaming models

Model	Parameter	Range or Options	Selected
HT	Split Criterion	Gini, InfoGain	InfoGain
	Split Confidence	0.001 - 0.5	0.01
	Tie Threshold	0.01 - 0.1	0.05
	Grace Period	200 - 500	200
	Max Tree Depth	10 - 30	20
ARF	All HT parameters above		
	Ensemble Size	10 - 20	10
SLR	Lambda	0.01 - 0.1	0.1
	Regularizer	Zero, L1, L2	L2
	Regularization	0.001 - 0.1	0.01

TABLE II: Key evaluation metrics for HT, ARF, SLR, DT

Metric	Streaming			Batch
	HT	ARF	SLR	DT
Accuracy	0.91	0.91	0.91	0.91
Precision	0.83	0.82	0.88	0.91
Recall	0.93	0.93	0.89	0.91
F1-score	0.88	0.87	0.88	0.91

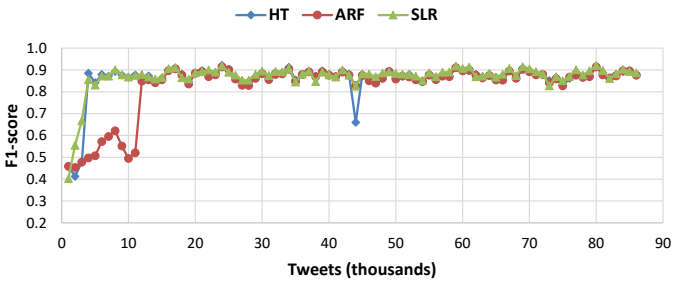


Fig. 5: F1-score for HT, ARF, and SLR.

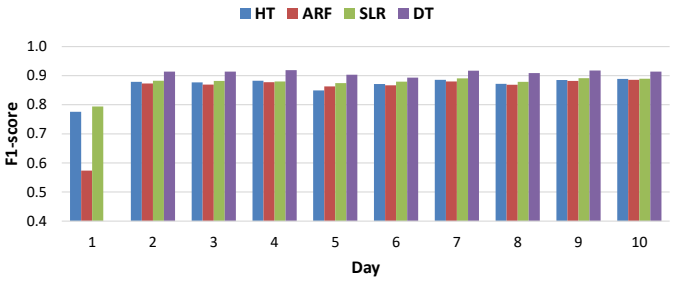


Fig. 6: F1-score for streaming vs. batch ML methods.

2) We compare the performance of streaming and batch-based ML algorithms (Section V-B).
3) We examine the scalability of our approach (Section V-C).

Experimental Setup. Our experiments were executed on a server with a 64-bit, 8-core, 3.2GHz CPU, 64GB RAM, and 2.1TB storage, running CentOS Linux 7.2. We implemented our approach over streamDM v0.2 and Spark Streaming v2.3.2. The dataset used consists of $86k$ tweets, from which $53,835$ are labeled as *normal* and $32,149$ as *aggressive*.

Hyperparameter Tuning. For each ML model, we used grid search to find optimal hyperparameter settings. Table I lists the value ranges or options considered for each parameter for the 3 streaming ML models used in our experimental evaluation, namely Hoeffding Tree (HT), Adaptive Random Forest (ARF), and Streaming Logistic Regression (SLR).

Performance Metrics. For all ML methods, we measure typical ML metrics such as accuracy, precision, recall, and F1-score. We focus on *F1-score* because it integrates precision and recall into a unified measure. For streaming, we used the popular *prequential evaluation* scheme, where instances are first used to test, and then to train the streaming ML model.

A. Performance of Streaming ML Methods

For this experiment, we executed the three streaming binary classifiers using prequential evaluation over the entire labeled Twitter dataset and show the overall evaluation metrics in Table II. In general, all three approaches achieve very similar performance with 91% accuracy and $\sim 88\%$ F1-score.

Figure 5 shows the F1-scores for the three methods every one thousand tweets. We notice that all methods eventually achieve an F1-score in the range of 83-92%. Through time, all methods exhibit similar trends, i.e., their variability in performance is similar. This means they are affected in a similar manner by transient changes in the distributions of the available features. The only exception is for the HT, whose performance dropped once at 66% due to a sudden decrease in recall. Moreover, HT and SLR take $\sim 4k$ instances to reach their full capacity with respect to detection performance, while ARF takes longer ($\sim 12k$ instances) to plateau its performance.

B. Streaming vs. Batch ML Methods

Compared to streaming ML methods that process each instance only once, batch methods typically observe each instance multiple times. Thus, it is important to compare the performance of streaming methods against similar batch methods. Specifically, we tested the Decision Tree $J48$, Random Forest, and Logistic Regression using $WEKA$ v3.7[10].

In order to offer a fine-grained comparison between streaming and batch methods, we implemented a training scenario for our batch methods that periodically updates the batch model on newly acquired and labeled data. In particular, the batch method is trained on data of one day and tested on the next day. Then, it is trained on the data of the subsequent day and tested on the day after, etc. Note that our Twitter dataset was collected over a period of 10 consecutive days of $\sim$8-9k tweets each day. As the batch methods performed very similarly, we only show results for the decision tree (DT).

Figure 6 shows the daily F1-scores for the three streaming ML methods and the batch DT. There are two key observations arising from the results. First, the batch method (DT) performs 1%–3% better compared to all streaming methods for all days. Second, the performance of DT is fairly stable through time since the model gets updated with new data every day, and hence, it can keep up with transient changes in the

[10]https://www.cs.waikato.ac.nz/ml/weka/

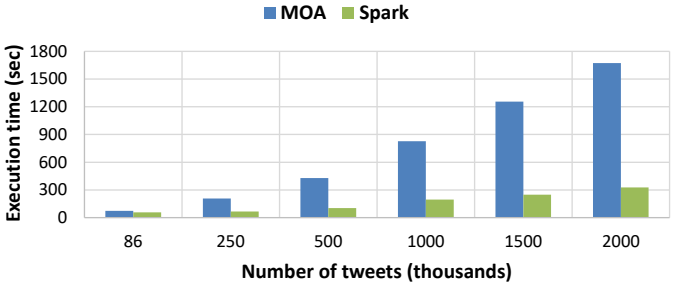


Fig. 7: Execution time of HT in MOA vs. Spark, given a number of incoming tweets to be processed.

features' distributions. The batch approach, however, is not convenient in practice as it requires storing all the labeled tweets of each day and performing a separate training process. Overall, even though a streaming ML method processes each instance only once, it is able to achieve similar performance to batch-based ML methods.

C. Scalability for Real-time Detection on Twitter

In this section, we study the scalability of our framework implemented over $Spark$ and compare it against MOA v19.05.0[11], a state-of-the-art framework for data stream mining. MOA is a single-threaded ML engine, while $Spark$ processing is naturally parallelizable. In each case, we process a fixed number of unlabeled tweets (ranged from $250k$ to $2m$ tweets) intermixed with the $86k$ labeled tweets.

Figure 7 shows the total execution time for processing the entire pipeline (recall Section III-A) with HT, while using MOA or $Spark$. MOA exhibits a clear linear trend, where doubling the number of tweets to be processed approximately doubles the execution time taken to process them. This is to be expected, since MOA processes tweets sequentially. $Spark$, on the other hand, takes advantage of task parallelism and eight available threads to process the same number of tweets much faster than MOA. For example, to process 2 million tweets, $Spark$ needs $5.1\times$ less time than MOA. These performance results clearly demonstrate the scalability of our streaming approach, in order to achieve real-time aggression detection.

VI. Conclusion and Future Work

In this work, we have developed an aggression detection framework for social media, where the entire pipeline – from preprocessing and feature extraction to training and testing – employs the data streaming paradigm. The streaming ML models built are incrementally updated and remain always up-to-date, despite any transient aggression behaviors. The evaluation results show that such models can perform similar to popular batch-based ML methods, achieving 82-93% accuracy, precision, and recall after processing just a few thousand tweets. At the same time, the framework is remarkably scalable and can process millions of tweets in minutes. In the future, we plan to examine finer granularity classification tasks beyond binary, as well as the ability of detecting a variety of behaviors (e.g., bullying, aggressive, abusive, offensive, racist, sexist) for diverse media platforms such as Twitter, Facebook, Instagram, YouTube, etc.

[11]https://moa.cms.waikato.ac.nz/

Acknowledgements

The research leading to these results has received funding from the EU's H2020 Programme, No 691025 (MSCA-RISE project ENCASE) and No 830927 (project CONCORDIA). The paper reflects only the authors' views and the Commission is not responsible for any use that may be made of the information it contains.

References

[1] D. Chatzakou, N. Kourtellis, J. Blackburn, E. De Cristofaro, G. Stringhini, and A. Vakali, "Mean Birds: Detecting Aggression and Bullying on Twitter," in *WebSci*. ACM, 2017, pp. 13–22.

[2] D. Chatzakou, I. Leontiadis, J. Blackburn, E. D. Cristofaro, G. Stringhini, A. Vakali, and N. Kourtellis, "Detecting Cyberbullying and Cyberaggression in Social Media," *ACM Transactions on the Web (TWEB)*, vol. 13, no. 3, pp. 1–51, 2019.

[3] M. Yao, C. Chelmis, and D. S. Zois, "Cyberbullying Ends Here: Towards Robust Detection of Cyberbullying in Social Media," in *WWW*, 2019.

[4] Y. Chen, Y. Zhou, S. Zhu, and H. Xu, "Detecting Offensive Language in Social Media to Protect Adolescent Online Safety," in *PASSAT and SocialCom*, 2012.

[5] N. Djuric, J. Zhou, R. Morris, M. Grbovic *et al.*, "Hate Speech Detection with Comment Embeddings," in *WWW*. ACM, 2015.

[6] I. Kayes, N. Kourtellis, D. Quercia, and F. Iamnitchi, A. & Bonchi, "The Social World of Content Abusers in Community Question Answering," in *WWW*. ACM, 2015.

[7] G. E. Hine, J. Onaolapo, E. De Cristofaro, N. Kourtellis, I. Leontiadis, R. Samaras, G. Stringhini, and J. Blackburn, "Kek, Cucks, and God Emperor Trump: A Measurement Study of 4chan's Politically Incorrect Forum and Its Effects on the Web," in *ICWSM*. AAAI, 2017.

[8] P. Smith, J. Mahdavi, M. Carvalho, S. Fisher, S. Russell, and N. Tippett, "Cyberbullying: Its Nature and Impact in Secondary School Pupils," *Child Psychology and Psychiatry*, vol. 49, no. 4, pp. 376–385, 2008.

[9] K. R. Allison, "Social Norms in Online Communities: Formation, Evolution and Relation to Cyber-Aggression," in *Extended Abstracts of CHI*. ACM, 2018, p. 1–4.

[10] R. I. Rafiq, H. Hosseinmardi, R. Han, Q. Lv, and S. Mishra, "Scalable and Timely Detection of Cyberbullying in Online Social Networks," in *SCA*. ACM, 2018, pp. 1738–1747.

[11] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, "Abusive Language Detection in Online User Content," in *WWW*. ACM, 2016, pp. 145–153.

[12] Z. Zhang, D. Robinson, and J. Tepper, "Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network," in *European Semantic Web Conference*. Springer, 2018, pp. 745–760.

[13] T. Davidson, D. Warmsley *et al.*, "Automated Hate Speech Detection and the Problem of Offensive Language," in *ICWSM*. AAAI, 2017.

[14] W. Warner and J. Hirschberg, "Detecting Hate Speech on the World Wide Web," in *2nd Workshop on Language in Social Media*, 2012.

[15] P. Burnap and M. L. Williams, "Cyber hate speech on Twitter: An application of machine classification and statistical modeling for policy and decision making," *Policy&Internet*, vol. 7, no. 2, pp. 223–242, 2015.

[16] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep Learning for Hate Speech Detection in Tweets," in *WWW*. ACM, 2017, pp. 759–760.

[17] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of Tricks for Efficient Text Classification," *arXiv preprint arXiv:1607.01759*, 2016.

[18] A. M. Founta, C. Djouvas, D. Chatzakou, I. Leontiadis, J. Blackburn, G. Stringhini, A. Vakali *et al.*, "Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior," in *ICWSM*. AAAI, 2018.

[19] P. Domingos and G. Hulten, "Mining High-speed Data Streams," in *SIGKDD*. ACM, 2000, pp. 71–80.

[20] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck *et al.*, "Adaptive Random Forests for Evolving Data Stream Classification," *Machine Learning*, vol. 106, no. 9-10, pp. 1469–1495, 2017.

[21] M. Giatsoglou, D. Chatzakou, N. Shah, C. Faloutsos, and A. Vakali, "Retweeting Activity on Twitter: Signs of Deception," in *PAKDD*. Springer, 2015, pp. 122–134.