VAM: An End-to-End Simulator for Times Series Regression and Temporal Link Prediction in Social Media Networks

Fred Mubang
Department of Computer Science & Engineering
University of South Florida
Tampa, FL, USA
fmubang@mail.usf.edu

Lawrence O. Hall
Department of Computer Science & Engineering
University of South Florida
Tampa, FL, USA
lohall@mail.usf.edu

***Abstract*—We present a machine learning driven end-to-end simulator, called the *Volume-Audience-Match* simulator, or *VAM*. *VAM*'s purpose is to simulate future phenomena related to various topics of discussion in social media networks. We focus our attention on the social media platform, Twitter, due to its abundant use in today's world.**

***VAM* was applied to do time-series forecasting to predict the future: (1) number of total activities, (2) number of active old users, and (3) number of newly active users over the span of 24 hours from the start time of prediction. *VAM* then used these macroscopic volume predictions to perform user link predictions. A user-user edge was assigned to each of the activities in the 24 future timesteps.**

We report that VAM outperformed multiple baseline models in the time series task, which were the ARIMA, ARMA, AR, MA, and the Persistence Baseline models. Furthermore, we show that VAM outperformed the Persistence Baseline model used for the user-assignment tasks. It is also shown that using Reddit activity data improves prediction accuracy.

***Index Terms*—Time Series Prediction, Extreme Gradient Boosting, Social Media, Link Prediction**

I. Introduction

Social media's vast societal influence is apparent. Recent research has shown its effect in many aspects of society, such as election campaigns [1], the spread of COVID-19 misinformation [2], and the promotion of pump and dump cryptocurrency schemes [3].

Clearly, it would be ideal to predict the future phenomena related to any topic on any social media platform. To that end, we created an end-to-end simulator, called the *Volume-Audience-Match Algorithm*, or *VAM*. *VAM*'s goal is to predict what will happen for a given topic on a social media platform. Experimental results are shown on 18 topics across Twitter.

VAM is a model that consists of two components, or *modules* that work in the following way. Firstly, for each topic in a given social media platform, at some time step of interest, T, the *Volume Prediction Module* of *VAM* takes as input a set of past time series features both related to that topic, as well as external exogenous features that may influence that topic's behavior in the future. It then uses these features in order to perform time series forecasting. For any given *topic-timestep* pair, VAM predicts three time series of length S which are: (1) the topic's future event volume time series, (2) the topic's newly active user time series, and (3) the topic's active old user time series.

Secondly, the *User-Assignment Module* of *VAM* uses these 3 time series predictions, as well as previous user interaction history, to tackle the more fine-grained task of predicting, for a given topic, within the timespan of $T+1$ up to $T+S$: (1) which user performs which action and (2) with whom each user interacts. We frame this problem as a link prediction problem. An edge is comprised of a child user u and a parent user, v. An edge exists between u and v if u reacts to a post written by v. In Twitter, this reaction takes the form of a retweet, or tweet in the case of an initial tweet (i.e. self-loop).

Note that we use the term "module" to differentiate from the term "model" for clarity throughout this work. *VAM* is the name of the overall model, while the *Volume-Prediction* Module is the component of *VAM* that predicts the volume predictions, and the *User-Assignment Module* is the component of *VAM* that performs the user-to-user predictions.

We tested *VAM*'s predictive power on the Twitter dataset related to the Veneuzuelan political crisis [4] [5] . A time period spanning from Decemeber 28, 2018 up until March 7, 2019 was used.

This paper makes the following contributions.

1) We introduce *VAM*, a simulation pipeline that performs both time series regression and temporal link prediction tasks in an end-to-end manner.
2) We show that VAM strongly outperforms multiple baselines across a myriad of metrics for the time series prediction task.
3) We provide an analysis of the use of social media platform features to determine what helps *VAM* achieve the best time series prediction performance in Twitter. We show that using features from activity on Reddit was shown to improve predictions of Twitter activity.
4) Lastly, we show that VAM greatly outperforms the baseline model in multiple user-assignment tasks. We illustrate this by showing that VAM outperforms the baseline in the old user prediction task, the indegree prediction task, and Page Rank prediction tasks.

II. Background and Related Work

A. General Popularity Prediction in Social Media

The term "general popularity prediction" refers to the prediction of the overall future volume of activities in social media networks. In these works, the user-level activity prediction

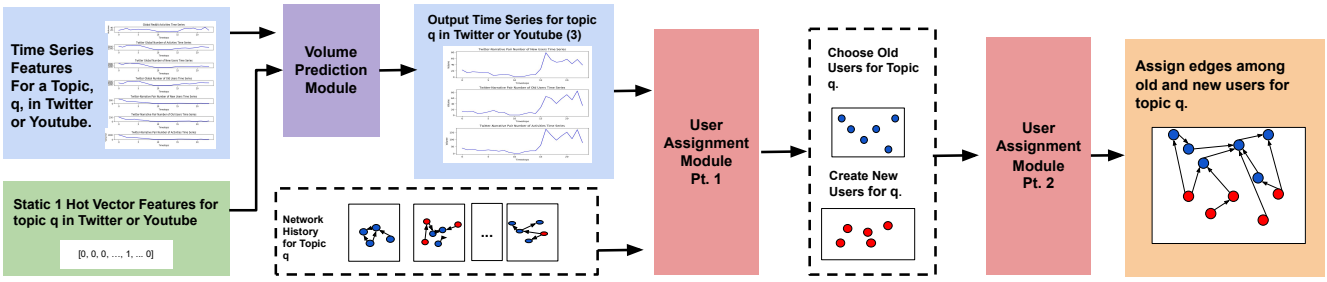


Figure 1: Framework for the Volume-Audience Match Algorithm (VAM).

is not considered. The work [6] utilizes neural networks to do this. The authors of [7] performed time series regression in the social media networks Facebook, Twitter, and Linkedin to predict the future volume of user activities in these platforms. They use various curve fitting models such as Polynomial, Logarithmic, and Exponential Regressions. The authors of [8] use a Hawkes-Process model to predict the times events in various media platforms. In [9], the authors used LSTMs to predict bursts of Github activity by utilizing exgogenous features from Reddit and Twitter.

B. Decompositional User-Level Prediction in Social Media

The term "decompositional user-level prediction" refers to works that aim to predict future user-level activity in social media networks, however, the methods break the task into 2 or more subtasks. There are 2 types of decompositional approaches we observed in the literature. Firstly, there are what we call "Volume-to-User" approaches, that first predict the overall number of events in a social media network with an initial model, and then assign users to these actions using a 2nd model. The framework in this work, *VAM* falls into this category. The previous works that also fall into this category are the SocialCube model [10], an ARIMA-driven method, as well as the proposed models of [11] and [12], which are LSTM-driven methods. *VAM* mainly differs from these methods in that it is used to predict user-to-user interactions in Twitter and YouTube, while the other methods predict user-to-repository interactions in Github.

The other type of decompositional methods are what we call the "clustering-based methods". These methods first use an initial model to cluster users in a social media network, and then a 2nd model to predict the future user-level actions using the cluster information. In [13] the authors use K-Means Clustering to cluster Github users based on their activity rate and in [14], the authors cluster repositories based on their topics such as programming languages and profile keywords. In each work, a 2nd set of models is then used to predict, using the cluster-information, the most likely user-repository pairs to occur in the future.

C. Direct User-Level Prediction in Social Media

The direct user-level prediction methods predict future user activity in social media networks, but do so directly, unlike the decompositional approaches. The works of [15] and [16] utilize embedding neural networks to do this. The works of [17] and [18] use neural networks on sequences of adjacency matrices to predict user activity over time in Twitter. In [19] the authors introduce the Multiplexity-Based-Model, which captures social network evolution based on preferential attachment, attention and recency cognitive bias.

In [20], the authors used 3 sampling models, 1 Bayesian model, and 1 link prediction model to predict future user-to-repo interactions in Github. The authors found that the sampling models performed the best. Lastly, there's the work of [21], in which the authors created various machine learning models that predicted user-to-repository links in Github, as well as user comment threads in Twitter and Reddit.

D. General Temporal Link Prediction

There have been several previous works on temporal link prediction algorithms. Some utilize neural networks that embed each node in a given network into a low dimensional space, such as dyngraph2vec [22] and tNodeEmbed [23]. These embeddings can then be used for temporal link prediction or node classification. In our work, we do not employ node embedding as it can be computationally expensive in terms of training time and space.

There are also matrix factorization approaches to temporal link prediction, which are discussed in [24], [25], and [26]. However, these approaches also struggle with scalability due to high computational cost.

Lastly, there are temporal link prediction approaches which employ probabilistic methods, such as [27] and [28]. These methods have been shown to be effective but suffer from computational complexity in terms of space [28], and time in the case of [27] and [28]. Also note that unlike *VAM*, these temporal link prediction methods do not have the ability to predict the appearance of new users.

III. Importance of Predicting New Users

In this work we define a "new" user at timestep T as someone who has not previously been involved with a topic in the period spanning $t = 1$ up to $t = T - 1$. Our data analysis showed that for certain topic-platform pairs, there are a considerable number of new users that appear each day. For 5 out of 18 topics, at least 40% of the active users within

a given day are new on average. For 9 out of 18 topics, at least 25% of the users in a given day are new on average. For this reason, it is important to predict their appearances and activities in addition to that of the old users. We provide more detailed analysis of this phenomenon in the supplemental materials [29].

IV. Problem Statements

VAM addresses 2 problems, namely the (1) volume prediction of users and activities, as well as (2) the assignment these volume predictions to the appropriate users within the context of a user-to-user link prediction. In this section we will discuss these two problems in more detail.

A. Volume Prediction Problem

Definition IV.1 (***The Volume Prediction Task for a topic–timestep Pair***)**.** Let us say, that for some given platform, one is given static and temporal features for some topic, $q \in Q$, at some timestep T. Intuitively, T can be thought of as the current time step of interest. One must then predict the following 3 future time series relating to this *topic-timestep* pair, (q, T): (1) the activity volume time series, (2) the active old user volume time series, and (3) the new active user volume time series. Each time series must span from time $T+1$ up to $T+S$, with S being an integer that represents the length of the predicted time series. Furthermore, let $\hat{\boldsymbol{Y}} \in \mathbb{R}^{3\times S}$ be a time series matrix that represents the aforementioned predicted time series. In other words, $\hat{\boldsymbol{Y}}$ represents a prediction matrix such that each row represents one of the 3 output time series, and each column represents a time step in any of the time series.

$\hat{\boldsymbol{Y}}$ approximates the ground truth matrix, $\boldsymbol{Y}$. The time frame that $\boldsymbol{Y}$ encompasses ($T+1$ to $T+S$) is called *forecast period of interest*, or F_T. F_T can be thought of as a tuple of the form ($T+1$, $T+S$). $T+1$ is the first time step in the *forecast period of interest* and $T+S$ is the last time step.

In order to address the volume prediction problem, we created XGBoost regression models. The inputs are time series features and static features related to a given topic. The granularity of time step information given the models is hourly, and they predicted 24-hour time series (1 day). XGBoost models are known for being relatively quick to train, while retaining high predictive accuracy [30]. Because of this, grid searches over many hyper parameters can be done in a reasonable amount of time. In our initial experiments we tried LSTMs, but abandoned them due to their lengthy training times and many hyper-parameters. As we will show in Section VI, we trained many models in order to ascertain the usefulness of different combinations of exogenous social media features. Therefore, having relatively short training times was paramount. Furthermore, from a practical standpoint, daily models that can be quickly trained are useful when one wishes to analyze and apply their results to the real world.

B. User-Assignment Link Prediction Problem

In this subsection the problem statement for the User-Assignment problem is introduced. Let $\{G\}_{t=1}^{t=T}$ be a sequence of static graphs such that $G = \{G_1, G_2, ...G_T\}$. G represents the user-interaction history of some topic, q, on some social media platform.

Each graph at time step t, G_t can be viewed as a tuple of sets of the form (V_t, E_t, w). V_t is the set of all users (nodes) $u \in V_t$ present in graph G_t. E_t is the set of all edges that exist in graph G_t. The edges in E_t are of form $(u, v, w(u, v, t))$. An edge exists in E_t if user u responded to a post made by user v at time step t. The term, w represents a weight function such that $w(u, v, t)$ represents how many times user u responded to v at time step t. Using this information, we can now define VAM's user-prediction task as follows:

Definition IV.2 (***The User-Assignment Prediction Task for a topic-timestep Pair***)**.** Let us say, for some *topic-timestep* pair, (q, T), one is given a matrix, $\hat{\boldsymbol{Y}} \in \mathbb{R}^{3\times S}$. This matrix contains, for (q, T), the future volume prediction time series for the (1) number of events, (2) number of old users, and (3) number of new users. Furthermore, let us say one is given a set, $\{G\}_{t=1}^{t=T}$, that represents the user-interaction history of topic q, for each of the T time steps. Given these 2 input items, predict a sequence, $\{\hat{G}^{future}\}_{t=1}^{t=S}$. Intuitively, one can think of $\hat{G}^{future}$ as a set containing the future user interactions over the next S timesteps with regards to *topic-timestep* pair, (q, T). So, a sequence $\hat{G}^{future}$ would have the following form: $\hat{G}^{future} = \{\hat{G}_1^{future}, \hat{G}_2^{future}, ..., \hat{G}_{24}^{future}\}$. Graph $\hat{G}_1^{future}$ represents the future user interactions for topic q at time step $T+1$. Graph $\hat{G}_2^{future}$ represents the future user interactions for topic q at time step $T+2$, and so on. Note that $\hat{G}^{future}$ is an approximation of the ground truth graph set, G^{future}.

Intuitively, one can view the User-Assignment problem as a temporal link prediction problem, but with the added "assistance" of the future volume counts from some predictive model. A pictorial overview of VAM is shown in Figure 1.

V. Data Collection

A. Twitter and YouTube Data Collection

The raw data on the 2019 Venezuelan political crisis used in these experiments were originally collected by the data collectors at the Leidos company. For the Twitter data, Subject Matter Experts (SMEs) compiled a list of keywords and Twitter handles that would allow for the collection of the most relevant Venezuela tweets. These subject matter experts were individuals hired by Leidos who were fluent in Spanish and very familiar with the political situation in Venezuela. The keywords were evaluated by the SMEs for both their precision and recall with regard to tweets about the Venezuelan political crisis. Frequently co-occurring groups of keywords were then used to create 18 "topics". For example, the topic *international/aid_rejected* is a topic comprised of two separate keywords "international" and "aid_rejected". This topic refers to the disputed President of Venezuela, Maduro, rejecting humanitarian aid from other countries to the people of Venezuela [31].

YouTube videos were selected by using the Google YouTube Query API with an SME-generated keyword list. Similar

to Twitter, the keywords used for this data collection were checked for relevancy to the Venezuelan political crisis. All individuals were anonymized (ID strings instead of names).

Twitter Graph Information				
Category	**Min**	**Max**	**Mean**	**Median**
Nodes	62,603	484,405	231,400	190,432
Edges	110,097	2,458,703	881,929	619,622
Edge Weight Total (Events)	122,581	4,580,984	1,397,973	826,865

Table I: Basic statistics of the 18 Twitter topic networks.

Table I contains basic statistics of the networks used in this work. Since there were 18 topics in our dataset, we had 18 Twitter networks. Each node in a Twitter graph represents a user and each edge represents an interaction between users, such as a retweet, or tweet (which can occur in the case of a self-loop). Note, that we did not include quotes or replies in our Twitter data, because they comprised such a small portion of overall Twitter activity (3.6%).

The smallest number of nodes for a given network was 62,603 nodes and the smallest number of edges observed was 110,097. The largest number of nodes in a given network was 484,405 and the largest number of edges observed was 2,485,703. Note that we only show basic statistics in Table I due to space constraints. For the full information of each network, see the supplemental materials in [29].

The YouTube networks are not shown because we only used the YouTube data for time series features for the *Volume-Prediction* task, and not for the *User-Assignment* task (which performs network predictions).

B. GDELT Data Collection

GDELT is a publicly available geopolitical event dataset [32]. It contains over 250 million event records in over 300 categories covering the entire world from 1979 to the present. It also contains a large network diagram connecting people, organizations, themes, and locations. We used a subset of the GDELT data related to the Venezuelan political crisis spanning from December 28, 2018 to March 7, 2019. Specifically, we used the time series of 3 feature categories in GDELT called *AvgTone*, *NumMentions*, and *GoldsteinScale*.

AvgTone represents the political sentiment of an event and *GoldsteinScale* represents the potential political impact of an event. The *NumMentions* value is a count of the total number of documents relating to an event across all source documents.

C. Reddit Collection

Reddit is a social media platform in which users read and comment on various message boards, known as *subreddits*. Reddit posts and comments spanning from Decemeber 28, 2018 to March 7, 2019 related to the Venezuelan political crisis were collected.

VI. Volume Prediction Methodology

A. Using Different Platforms and Volume Lookback Factors

We were interested in knowing if the next 24 hours of social media activity could be predicted from some initial timestep, T, so to that end, we set $S = 24$ in our experiments. We believe 24 hours is long enough to be useful in a practical application, but still short enough that it is a reasonable period for a model to predict within.

Furthermore, we wanted to know whether it was sufficient to use Twitter data alone in order to perform successful future activity predictions on Twitter, or if exogenous features from the other platforms were helpful. In order to determine this, we trained and tested every Twitter model with every possible subset of additional platform features. For example, we trained Twitter models on (1) only Twitter data, (2) Twitter and Reddit data only, (3) Twitter and YouTube data only, etc. All in all, there were 8 total Twitter/exogenous-platform combinations.

It is important to know how many previous hourly time steps should be used when creating samples for our datasets. To that end, we also tried different *volume lookback factors*, specifically 24, 48, 72, and 96. We refer to the volume lookback factor parameter as L^{vol}. The *User-Assignment* module of VAM also uses a lookback parameter, called L^{user}.

Training and testing was done with 32 different *Volume-Prediction* modules, because there were 8 platform combinations and 4 *volume lookback factor* combinations. For the sake of brevity, we just report the top 10 VAM models along with the 5 baseline models. Full results are in the supplementary material.

B. Sample Tuples

As previously mentioned, each sample in each dataset represents a *topic-timestep* pair, (q, T). The variable, q represents the topic of interest, and T represents the current time step of interest. Each *topic-timestep* sample is comprised of input features and output values. The inputs and outputs are described as follows.

Firstly, there is the *static input feature set*. This is a 1-hot vector that represents the topic of interest, q. Secondly, there are the *temporal input features*. These are the time series input features for our given sample. They differ depending on the model. The types of temporal features used are listed in Table II. Lastly, there are *the output targets*. The output for a given *topic-timestep* pair, (q, T), is the matrix $\boldsymbol{Y} \in \mathbb{R}^{3\times S}$. This matrix is comprised of the 3 output time series for the volume prediction task: the event volume time series, the new user volume time series, and the old user volume time series.

We shall illustrate this point with an example. Let us define the Twitter prediction matrix to be $\boldsymbol{Y}$, for topic $q = \textit{arrests}$, and current timestep of interest T= 100. Furthermore, we define a *volume lookback factor* of $L^{vol} = 48$ and we define the output time series size, S to be equal to 24.

Given these definitions, our model would use temporal information from time steps 53 up to 100 (48 time steps including 100) to predict the output values related to the Twitter phenomena for the *arrests* topic from time steps 101 to 124.

Our test period contained 21 *forecast periods of interest* (F_T). These periods were the 21 days spanning February 15th, 2019 to March 7th, 2019. Recall that there were $n^{topics} = 18$

Time Series Index Label	Time Series Description
1	New user volume time series for a given topic in Twitter.
2	Old user volume time series for a given topic in Twitter.
3	Activity volume time series for a given topic in Twitter.
4	New user volume time series for a given topic in YouTube.
5	Old user time series for a given topic in YouTube.
6	Activity volume time series for a given topic in YouTube.
7	Activity volume time series across all topics in Twitter.
8	New user volume time series across all topics in Twitter.
9	Old user volume time series across all topics in Twitter.
10	Activity volume across all topics in YouTube.
11	New user volume time series across all topics in YouTube.
12	Old user volume time series across all topics in YouTube.
13	The GDELT AvgTone time series.
14	The GDELT GoldsteinScale time series.
15	The GDELT NumMentions time series.
16	Activity volume time series in Reddit.

Table II: The table of all possible time series feature categories.

topics. So, for Twitter and YouTube each, there were $18*21 = 378$ test samples.

For the training period, the prediction days of interest spanned from December 28th 2018 to February 7th, 2019 (42 days). For the validation period, the prediction days of interest spanned from February 8th to February 14th (7 days).

For the training and validation sets, we wanted to generate as much data as possible, so, we calculated each daily sample both in terms of day and hour, and not just simply in terms of day like we did for the test set. That is, a sliding window was used and advanced 1 hour to create a new overlapping example. By using this method, we generated 17,730 samples for each training set, and 2,610 samples for each validation set.

C. XGBoost

Let $\mathcal{D}$ be a dataset such that: $\mathcal{D} = (\mathbf{x}_i, \boldsymbol{Y_i})$. Furthermore let the following be true:

$$|\mathcal{D}| = n^{samples} = n^{topics} * \tau; \mathbf{x}_i \in \mathbb{R}^m, \boldsymbol{Y_i} \in \mathbb{R}^{3\times S}.$$

The terms $n^{samples}$, n^{topics}, and τ represent the number of samples, topics, and prediction timesteps of interest, respectively. $\mathbf{x}_i \in \mathbb{R}^m$ represents an input feature vector of m features, and $\boldsymbol{Y_i} \in \mathbb{R}^{3\times S}$ represents the output matrix.

We then define a matrix of functions, $\Phi(\mathbf{x}_i) \in \mathbb{R}^{3\times S}$ such that:

$$\hat{\boldsymbol{Y_i}} = \Phi(\mathbf{x}_i) = \begin{bmatrix} \phi_{1,1}(\mathbf{x}_i) & \phi_{1,2}(\mathbf{x}_i) & \dots & \phi_{1,S}(\mathbf{x}_i) \\ \phi_{2,1}(\mathbf{x}_i) & \phi_{2,2}(\mathbf{x}_i) & \dots & \phi_{2,S}(\mathbf{x}_i) \\ \phi_{3,1}(\mathbf{x}_i) & \phi_{3,2}(\mathbf{x}_i) & \dots & \phi_{3,S}(\mathbf{x}_i) \end{bmatrix} = \\ = \begin{bmatrix} \hat{y}_i^{1,1} & \hat{y}_i^{1,2} & \dots & \hat{y}_i^{1,S} \\ \hat{y}_i^{2,1} & \hat{y}_i^{2,2} & \dots & \hat{y}_i^{2,S} \\ \hat{y}_i^{3,1} & \hat{y}_i^{3,2} & \dots & \hat{y}_i^{3,S} \end{bmatrix} \tag{1}$$

Each function $\phi_{a,b}(\mathbf{x}_i)$ in the matrix represents a separate XGBoost model, and each of these models maps to an output-type-and-timestep pair value, $\hat{y}_i^{a,b}$. An integer variable, a can be used to indicate any particular row in the matrix, such that $1 \leq a \leq 3$, and an integer variable, b can be used to indicate any particular column of the matrix such that $1 \leq b \leq S$. Recall that rows represent one of the 3 output types (actions, new users, or old users), while columns represent one of the S future time steps.

The function, $\Phi(.)$, represents the *Volume-Prediction Module*, which contains $3 * S$ XGBoost models, $\phi_{a,b}(.)$, and each XGBoost model is an ensemble of CART trees. Intuitively, one can think of each of the XGBoost models as "specializing" on a particular *(output-type, timestep)* pair.

There are 3 * S models used because XGBoost is comprised of regression trees. A regression tree can only predict 1 output. So, to predict a time series, one would need a regression tree for each timestep in the time series. The alternative to the multiple-model approach would be to predict an output, feed that output back into the XGBoost model as an input, predicting the 2nd output, and so on. The problem with this approach is that one would run into the issue of compounding errors over time. As a result, these errors could cause these model to predict time series that do not come close to approximating the ground truth at all.

D. XGBoost Parameter Selection

We used the *XGBoost* [30] and *sk-learn* [33] libraries to create and train our models. The parameters used for our XGBoost models are as follows. The subsample frequency, gamma, and L1 regularization were set to 1, 0, and 0 respectively. For the other parameters, we performed a grid search over a pool of candidate values. We used our validation set to evaluate for the best parameters to use. For the *column sample frequency*, the candidate values were 0.6, 0.8, and 1. For the *number of trees* parameter, the candidate values were 100 and 200. For the *learning rate*, the values were 0.1 and 0.2. For *L2 Regularization*, the values were 0.2 and 1. Lastly, for *maximum tree depth*, the values were 5 and 7.

For the loss function, Mean Squared Error was used. For normalization, log normalization was used.

E. Baselines

We compared VAM to 5 baseline models in this work, which were the *Persistence Baseline*, ARIMA, ARMA, AR, and MA models [34].

The *Persistence Baseline* model predicts the events during time frame $T+1$ to $T+S$ by simply outputting the events that occurred at time $T - S$ to T. The assumption of this model is that the future will exactly resemble the recent past. This assumption may sound naive, however we found this baseline to perform very well against the others.

The Auto-Regressive Integrated Moving Average model (ARIMA) and its variants (ARMA, AR, and MA) are widely used statistical models and, hence, used for comparison as well. The ARMA, AR, and MA models are variants of ARIMA depending on what the *p, d,* and *q* parameters are set to. The ARIMA model has $p > 0$, $d > 0$, and $q > 0$. The AR model has $p > 0$, $d = 0$, and $q = 0$. The ARMA model has $p > 0$,

$d = 0$, and $q > 0$. Lastly, the Moving Average (MA) model has $p = 0$, $d = 0$, and $q > 0$.

To train each of these ARIMA-based models, a grid search was performed with p and q's possible values being 0, 24, 48, 72, and 96, and d's possible values being 0, 1, and 2. A different model was trained per topic/output-type pair. So, for example, the *(Maduro, # of new users)* pair had its own ARIMA, ARMA, AR, and MA models. The validation set was used to select the best model parameters for the test period and the *RMSE* metric was used to select the best model parameters.

VII. Volume Prediction Results

A. Volume Prediction Metrics

In order to ensure that the time series predictions were correctly measured for accuracy, 6 different metrics were used over each of the 21 *forecast period of interest* instances of the test period spanning February 15th, 2019 to March 7th, 2019. Results were averaged across the 21 instances for each metric.

We used firstly, *RMSE* and secondly, *MAE* to measure how accurate each time series was in terms of "volume over exact time step". We thirdly used *Normalized Cumulative RMSE*, which converts the simulated and ground truth time series into cumulative sum time series, and then divides each by their respective maximum values. This metric allows us to know how well each predicted time step performed without considering the overall scale or "exact timing" of each value in the time series. This type of measurement is important because sometimes a time series could predict a burst within some range of timesteps, but not in the exact spot. However, knowing a burst of activities will occur within some range of timesteps is better than not knowing at all.

Fourthly, we used *Symmetric Absolute-Percentage-Error* (*SAPE*). This measures how accurate the total number of events was for each model, without regard to the temporal pattern. The formula is as follows. Let F be the forecast time series, and let A be the actual time series:

$$SAPE = \frac{|sum(F) - sum(A)|}{sum(F) + sum(A)} * 100\%$$

.

Fifthly, and sixthly, we used the *Volatility Error (VE)* and *Skewness-Error (SkE)* metrics. The *Volatility Error* is measured by calculating the standard deviation of both the ground truth and simulated time series, and then calculating their absolute difference. The *SkE* metric is measured by calculating the skewness of both the ground truth and simulated time series, and then calculating their absolute difference. The skewness statistic used utilizes the adjusted Fisher-Pearson standardized moment coefficient. It can be found at the top of page 7 in [35].

The *VE* and *SkE* metrics were used in order to measure how well the simulated time series captured the "burstiness" of the ground truth time series. We found that sometimes a particular model might seemingly have an impressive RMSE, MAE, or NRMSE relative to other models, but upon visually inspecting the time series plots, the model in question does not seem to capture the ground truth's "bursts" or "dips" that the other models seem to capture. We explain this phenomenon in more detail the supplemental materials [29].

B. Overall Results

Table III shows the the overall results for the best 10 Twitter models for the 6 aforementioned metrics. To see the full results of all 37 models, please refer to the supplemental materials [29].

Since there were many metrics, we calculated 1 "overall" metric that represents how well each model performed across all 6 metrics. We call this new metric the "Overall Normalized Metric Error (ONME)". It was calculated by creating 6 "metric groups", each comprised of the 37 model metric results for that particular metric. A similar "normalized error metric" was used in [19]. The model results within each of the 6 groups were normalized between 0 and 1 by dividing each model metric result by the sum of all model metric results within that particular group.

The models in each table are sorted and ranked from lowest to highest *ONME*.

In order to illustrate how well each *VAM* performed against the best performing baseline we used a metric that we call the "Percent Improvement From Best Baseline" ($PIMFBB$). These values represent, as a percent, how much the *ONME* improved from the best baseline, which in this case was the *Persistence Baseline*. The formula for this value is as follows:

$$PIMFBB = 100\% * \frac{BestBaselineError - ModelError}{BestBaselineError}$$

The upper bound of $PIMFBB$ is 100%, which occurs if a model's *ONME* is 0. This is clearly the best possible result. The lower bound for *ONME* is negative infinity because any given model could potentially perform infinitely worse than the best baseline.

C. In Depth Volume Prediction Result Analysis

The best Volume-Prediction (VP) Module for Twitter belonged to the *VAM-TR-96* model. This was the model trained on Twitter and Reddit data, with a lookback factor of 96. The *ONME* for this model was 0.02477, which was about a 17.53% improvement from the *Persistence Baseline* model. In second place was the *VAM-TY-96* model, which was trained with Twitter and YouTube data.

It is interesting to note that a *VAM* model trained on only Twitter data does not appear until 7th place (*VAM-T-96*). This suggests that YouTube and Reddit features can improve Twitter predictions.

What is also interesting to note is that none of the *VAM* models containing GDELT features appeared in the top 10 results. Perhaps the reason GDELT features were not as helpful as YouTube or Reddit could be the following. GDELT is an aggregator of news articles, which take time to write, vet, and edit. However, Twitter, YouTube, and Reddit posts do not require as long of a time period to create. As a result, the Twitter activity related to a real-world phenomena may have

already occurred by the time time GDELT adds a new article to its database.

D. VAM-TR-96 Metric Results by Topic

In the supplemental materials are bar plots and tables illustrating VAM's performance against the best baseline model per each topic and metric pair. Due to space limits, we briefly describe the topic-level metric results in this subsection.

For the RMSE metric, VAM won against the best baselines on 18 out of 18 topics. For MAE, VAM won 17 times; for Normalized Cumulative RMSE (NC-RMSE), VAM won 12 times; for Symmetric Absolute Percentage Error (S-APE), VAM won 12 times; for Skewness Error (SkE), VAM won 9 times; and for Volatility Error (VE), VAM won 13 times.

Overall, VAM outperformed the best baselines 81 out of 108, or 75% of the time. VAM performed particularly well at the "exact volume over time" metrics (RMSE and MAE). It performed decently for the "magnitude" or "scale" metric (S-APE). It also performed decently on the Volatility Error metric, which measures how well the volatility, or standard deviation of the time series matches that of the ground truth. It struggled the most with the Skewness Error metric, which measures the assymetry of the time series.

Figure 2 shows the performance of the *VAM-TR-96* model against the 5 baselines on various topics and days. As one can see, *VAM* was able to more closely approximate the ground truth than the baseline models. On the *other/chavez/anti* and *protests* topics, *VAM* more closely approximated some bursty ground truth behavior in comparison to the baseline models (with some error of course).

VIII. User Assignment Methodology

A. Overview

Recall the user-assignment task for VAM. Once the *VP-Module* predicts matrix $\hat{\boldsymbol{Y}}$ for *topic-timestep* pair, (q, T), the task for the *UA-Module* is to use $\hat{\boldsymbol{Y}}$ and the graph history set, $\{G\}_{t=1}^{t=T}$ to predict the future graph sequence, $\{\hat{G}^{future}\}_{t=1}^{t=S}$. As mentioned earlier, $\hat{G}^{future}$ is an approximation of the ground truth graph set, G^{future}.

The user-assignment is done in the following way. For S iterations, a graph G_s^{future} ($s \leq S$) is generated and added to the overall final G^{future} sequence. Eight main data structures are used to aid in the user-assignment task. They will be described in the following subsections.

B. The Recent History Table

Firstly, there's a recent history table, called H^{recent}. This is a table containing event tuples generated using information from G. Each tuple contains the following information: (1) the child (acting) user, (2) the parent (receiving) user, (3) the number of interactions between child and parent at some timestep t, (4) a flag indicating whether the child is new at timestep t, and (5) a flag indicating whether the parent is new at timestep t.

H^{recent} is known as a "recent" history table because it is made from only the most recent graph snapshots from G. The lookback factor parameter L^{user} is used to determine the number of snapshots to use. For example, if $L^{user} = 5$, then only the 5 most recent graphs in sequence G will be used to make H^{recent}. The assumption here is that recent history is all that is needed to make temporal network predictions.

C. Old and New Users

The next two data structures are the set of selected old users, $\hat{O}_s$ and set of generated new users, $\hat{N}_s$. Note that *VAM* "knows" the number of old and new users because that is what was predicted by the *Volume-Prediction* Module.

D. Old and New User Probability Tables

The 4th data structure is the *Old User Activity Probability Table* (W^{old}). It is a table containing each old user's probability of being active (e.g. tweeting/retweeting) at some timestep t.

The 5th data structure is the *New User Archetype Table* (W^{new_arch}). This table models how different "archetypes" of new users have behaved in the past. These archetypes are generated using recently active user information from H^{recent}. These attributes are the (1) probability of acting and (2) probability of being influential (e.g. being retweeted). This archetype table is then used to create the 6th data structure W^{new} which contains the activity and influence probabilities for the users in $\hat{N}_s$.

E. Old and New Parent Tables

The last two tables are the old and new user parent tables, D^{old_parent} and D^{new_parent}, respectively. These are hash tables in which each key is a user, and the value is a table containing (1) a list of that user's historical "parents" (a.k.a. users that the user of interest is most likely to retweet) and (2) the probability that the user of interest will retweet or comment that particular parent.

These 8 data structures are used to predict each G_s^{future} in the temporal sequence G^{future}. Algorithm 1, labelled *Assign_Users* contains the pseudocode for the User-Assignmnent algorithm. For an in-depth explanation of the algorithm, please refer to the supplemental materials [29].

IX. User-Assignment Metrics

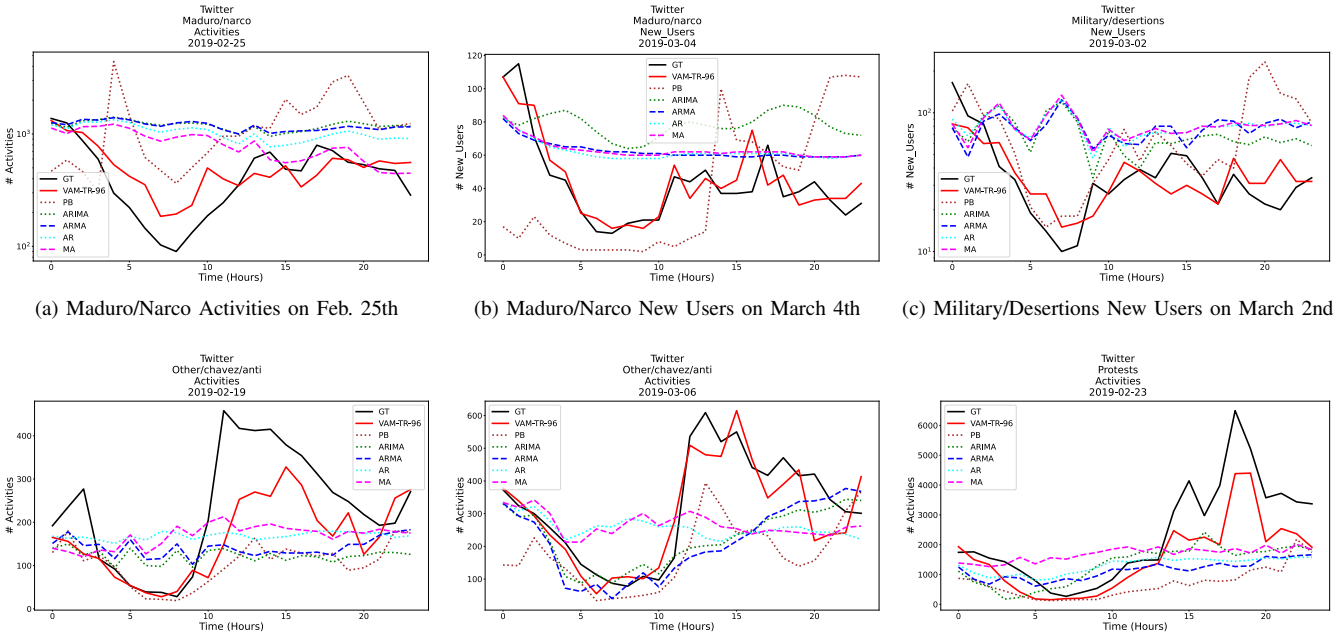
A. Jaccard Similarity for Old Users

To measure how well VAM predicted old users, we used the weighted Jaccard Similarity metric, also known as the Ruzicka Similarity [36]. It aims to measure how well VAM predicted the old users in each hour with consideration of how "influential" they were to the overall network (a.k.a. how often they were retweeted).

Let A represent the set of the actual old user set within a particular hour, and let P represent the predicted set of old users within a particular hour. Furthermore, let $\mathbf{a}$ and $\mathbf{p}$ represent vectors that contain the weights of each user in the A and P sets, respectively. For example, $\mathbf{a}_k$ represents the the weight of user A_k from the A set. With this in mind, the weighted Jaccard Similarity is defined as follows:

Overall Twitter Volume Prediction Results									
Rank	**Model**	**RMSE**	**MAE**	**VE**	**SkE**	**S-APE**	**NC-RMSE**	**Overall Normalized Metric Error**	**ONME PIFBB (%)**
1	**VAM-TR-96**	675.08053	482.97939	358.63956	0.99388	26.91419	0.11566	**0.02477**	**17.53362**
2	VAM-TY-96	687.28588	491.65766	366.44394	0.9095	27.62297	0.1172	0.02479	17.45628
3	VAM-TR-24	665.88435	467.68103	351.6489	0.95903	28.04565	0.12311	0.02479	17.45479
4	VAM-TR-48	666.11639	472.04979	356.92729	0.98988	27.50714	0.11939	0.02481	17.39281
5	VAM-TRY-96	683.30911	489.27646	365.33833	0.96186	27.19465	0.1184	0.02495	16.94269
6	VAM-TR-72	681.71863	483.31309	369.61879	0.97503	27.51952	0.11992	0.0251	16.43668
7	VAM-T-96	682.29561	488.55522	370.65342	0.99218	27.63607	0.11627	0.02512	16.3687
8	VAM-TRY-24	675.10884	477.63081	363.32588	0.96143	29.3425	0.12553	0.02536	15.57669
9	VAM-T-48	691.26144	490.17236	376.60796	0.95685	28.29505	0.12152	0.02539	15.47802
10	VAM-TY-48	687.22783	492.10927	375.16132	0.94802	29.09011	0.1208	0.02542	15.37886
11	Persistence_Baseline	888.9082	619.26606	454.85759	0.96809	29.42484	0.15699	0.03004	0.0
12	MA	922.13789	701.64627	444.88704	1.38811	37.35475	0.14152	0.0333	-10.86283
13	ARMA	1068.57479	823.89253	531.86923	1.24775	34.36105	0.1353	0.03489	-16.14924
14	AR	1174.3006	904.72248	605.11153	1.37021	34.54514	0.12422	0.0372	-23.83783
15	ARIMA	1321.44676	1034.54112	658.98357	1.21444	37.36525	0.14517	0.04026	-34.04431

Table III: Overall Twitter VAM volume prediction results compared to baselines.



(a) Maduro/Narco Activities on Feb. 25th (b) Maduro/Narco New Users on March 4th (c) Military/Desertions New Users on March 2nd

(d) Other/Chavez/Anti Activities on Feb. 19th (e) Other/Chavez/Anti Activities on March 6th (f) Protests Activities on Feb. 23rd

Figure 2: Some 24-hour time series plots of volume predictions. The solid black curves are ground truth time series and the solid red curves are the *VAM-TR-96* predicted time series. The dotted and dashed curves are the baseline time series.

$$J(\mathbf{a}, \mathbf{p}) = \frac{\sum_k min(\mathbf{a}_k, \mathbf{p}_k)}{\sum_k max(\mathbf{a}_k, \mathbf{p}_k)}$$

.

B. Defining Success for New User Prediction

Since our task also involves predicting the creation and activity of new users, in addition to old users, defining and measuring predictive success becomes a bit more difficult. Since we do not "know" the names of a new user before they appear in the ground truth, it is impossible to exactly match a new user that *VAM* generates, with a new user that exists in the ground truth. So, in order to work around this issue, we measure success using more macroscopic views of the network, specifically the Page Rank Distribution and the Complementary Cumulative Degree Histogram (CCDH).

C. Page Rank and Earth Mover's Distance

The Page Rank score [37] measures how influential a particular node is upon the entire network. In our experiments, we calculated Page Rank on the weighted indegree of our networks. If *VAM* properly simulated the activities of old and new users, that means that *VAM*'s simulated network Page Rank Distribution should closely approximate the ground

Algorithm 1 Assign_Users

Input: The full temporal graph G; the number of output timesteps to be predicted S; the user assignment lookback factor L^{user}; the volume prediction matrix $\hat{\boldsymbol{Y}} \in \mathbb{R}^{3\times S}$; Old user index old_idx; New user index new_idx; Activity index act_idx

Output: The predicted temporal graph sequence $\hat{G}^{future}$

$G^{recent} = Get\_Recent\_Temporal\_Graph(G, L^{user})$
Initialize G^{future} as an empty array
for $s = 1$ up to S **do**
    $num\_old\_users \leftarrow \hat{\boldsymbol{Y}}[old\_idx][s]$ # predicted old users at s
    $num\_new\_users \leftarrow \hat{\boldsymbol{Y}}[new\_idx][s]$ # pred. new users at s
    $num\_acts \leftarrow \hat{\boldsymbol{Y}}[act\_idx][s]$ #pred. actions at s
    $H^{recent} \leftarrow Get\_Recent\_History\_Table(G^{recent})$ #get recent history
    $W^{old\_cand} \leftarrow Get\_Active\_Old\_User\_Candidates(H^{recent})$ #get "pool" of potentially active old users
    $W^{old}, \hat{O}_s \leftarrow Get\_Most\_Likely\_Active\_Old\_Users(W^{old\_cand})$ # select most likely active users from pool
    $\hat{N}_s \leftarrow Generate\_New\_Users(num\_new\_users)$ #create IDs to represent new users and store in a set
    $W^{new\_arch} \leftarrow Get\_New\_User\_Archetype\_Table(H^{recent})$ #create new user arch. table
    $W^{new} \leftarrow Assign\_Attributes\_to\_New\_Users(W^{new\_arch}, num\_new\_users)$ #assign attributes to new users
    $D^{old} \leftarrow Create\_Old\_User\_Parent\_Table(H^{recent}, \hat{O}_s)$ # get old users' most likely parents
    $D^{new} \leftarrow Create\_New\_User\_Parent\_Table(H^{recent}, \hat{N}_s, W^{new\_arch})$ # get new users' most likely parents
    $\hat{G}_s^{future} \leftarrow Create\_Links(\hat{O}_s, \hat{N}_s, num\_acts, W^{old}, W^{new}, D^{old}, D^{new})$ #perform link prediction with user sets
    Append $\hat{G}_s^{future}$ to $\hat{G}^{future}$ #append newly predicted graph to array
    $G^{recent} = Get\_Recent\_Temporal\_Graph(G^{recent}, \hat{G}_s^{future}, L^{user})$ #update G^{recent} with predicted $\hat{G}_s^{future}$ graph
end for
return $\hat{G}^{future}$

truth network's Page Rank Distribution. In order to measure the distance between the predicted and actual Page Rank distributions we used the Earth Mover's Distance Metric [38].

D. The CCDH and Relative Hausdorff Distance

The Complementary Cumulative Degree Histogram (CCDH) of a graph G is defined as $(N(k))_{k=1}^{\inf}$, in which $N(k)$ denotes the number of vertices of degree *at least k* [39]. It is closely related to the more well-known concept of *degree distribution*. In our experiments, we calculate the CCDH on the unweighted indegree distribution of the ground truth and simulated networks. Success is defined by how closely the predicted CCDH matches the ground truth CCDH.

In order to measure the distance between the predicted network CCDH and the ground truth network CCDH we use the Relative Hausdorff (RH) Distance. Previous work has shown the RH-Distance to be a suitable metric for measuring the distance between two CCDHs [40].

X. User-Assignment Results

A. Multiple Trials

Since VAM's User-Assignment algorithm is probabilistic, it was run 5 times with 5 different seed initializations. The 3 user-assignment metrics (Jaccard Similarity, EMD, and RHD) were then calculated across each of the 5 trials and averaged together. These averaged results are shown.

B. User Clustering

We sought to analyze the user-assignment results on 3 sets of users: (1) the full set of users, (2) the highly influential cluster of users, and (3) the less-influential, or "lowly" cluster of users. We did this because we wanted to see how good VAM was at predicting different types of users.

The 2 "highly influential" and "lowly influential" clusters were created by calculating the total weighted indegree of each user per each topic within the full period of the data (December 28th, 2018 to March 7th, 2019). The weighted indegree can also be thought of as the total number of times a user was retweeted, and this number can be thought of as a representation of how "influential" a user was in its respective network. Then, the median indegree values for a topic were calculated. For each topic, if a user's indegree value was equal to or lower than the median, the user was placed into the "lowly influential" cluster. If a user's indegree value was above the median, the user was placed into the "highly influential" cluster. Since there are 18 topics in our dataset, there were 18 "highly influential" user clusters, and 18 "lowly influential" user clusters.

C. Jaccard Similarity Results

Figures 3a, 3b, and 3c contain the weighted Jaccard Similarity results for the full user set, the highly influential cluster, and the lowly influential cluster, respectively. The orange bars represent the VAM scores, and the blue bars represent the Persistence Baseline (PB) scores. Higher bars mean better results. For the sake of easier visualization, each VAM/PB score on a particular topic was normalized in a pair-wise fashion so the results across all 18 topics could be more easily viewed. In other words, for a particular topic, the normalized VAM score was calculated as follows:

$$normalized\_VAM\_score = \frac{VAM\_score}{VAM\_score + PB\_score}$$

This same procedure was done for the PB scores as well. As a result each VAM and PB score per topic shown in the barplots will add up to 1.

Overall, VAM performed well on all 3 of these user groups. On the full user cluster, VAM outperformed the baseline 18 out of 18 times on the full user set. For the high and low influence clusters, VAM won 17 and 18 times, respectively. Overall, VAM outperformed the baseline on 53 out of 54 topic-metric pairs, or about 98% of the time. We also measured VAM's performance against the baseline for the Unweighted Jaccard Similarity. We found that VAM strongly outperformed the baseline on that metric as well, however due to space constraints, those results have been placed in the supplemental materials [29].

To see tables containing the unweighted and weighted Jaccard Similarity numerical results, please refer to the supplemental materials.

D. Earth Mover's Distance Results

Figure 4 contains the Earth Mover's Distance results for the full user set, highly influential users, and lowly influential users. In these plots, the lower bars are better. For the full user set, VAM outperformed the baseline 17 out of 18 times. For the highly influential cluster, VAM won 18 out of 18 times, and for the lowly cluster, it won 8 out of 18 times. Overall, VAM won over the baseline 43 out of 54, or 79% of the time. As one can see, VAM did well with predicting the node influence of the entire network as a whole and with the highly influential users, but struggled to predict the node influence of the less-influential users. This could be because the less-influential users are less frequently retweeted, therefore making them harder to predict.

E. Relative Hausdorff Distance Results

Similar to the Earth Mover's Distance plots, Figure 5 contains the Relative Hausdorff Distance results for the full user set, highly influential users, and lowly influential users. For the full user set and highly influential cluster, VAM won 15 out of 18 times. For the lowly influential cluster, VAM won 11 out of 18 times. Overall, VAM outperformed the baseline on 41 out of 54 topic-metric pairs, or 75.9% of the time. Also similar to the Earth Mover's Distance results, VAM also performed worse on the lowly influential cluster.

XI. Runtime Information

The Volume-Prediction and User-Assignment modules of VAM were run on computers with an Intel Xeon E5-260 v4 CPU. Each CPU was comprised of 2 sockets, 8 cores, and 16 threads. Each computer had 128 GB of memory.

The Volume Prediction Module was run on a single computer and took about 7 minutes to train and test. The User-Assignment Module was run in parallel over 5 computers (1 per trial). The average runtime of the User-Assignment Algorithm across the 5 trials was about 2 hours and 13 minutes, which is quite reasonable considering that there were 18 topics and millions of edges. Since there were 21 days in the test period, on average the User-Assignment algorithm took about 6.33 minutes to simulate the activities for 1 day (or 24 hours) across all 18 topics.

XII. Conclusions and Future Work

In this work we presented the *Volume Audience Match* simulator, *VAM*. It is the first end-to-end simulator of user activity in social media platforms that utilizes time series prediction and probabilistic link prediction to estimate future activity of both old and new users. In this work, VAM was used to predict both overall and user-level activity from the recent Venezuela political crisis on a per-topic basis.

On the Volume-Prediction task, VAM was shown to have good performance against multiple widely used statistical models (ARIMA, ARMA, AR, and MA), as well as the Persistence Baseline. As previously mentioned, it outperformed these baselines on 81 out of 108 topic-metric pairs, or 75% of the time. On the User-Assignment task, VAM strongly outperformed the Persistence Baseline on 137 out of 162 topic-metric pairs, or about 84.56% of the time. With refinement, VAM could be used as an alert system for potential future real world activity.

Future work includes a variety of tasks. Firstly, we would aim to use 2 machine learning models in the *User Assignment Module* to predict the most likely active users and the final link predictions. Perhaps these models could outperform the weighted random sampling approach that VAM's *User-Assignment* module currently employs. For the *Volume Prediction Module* we would try multiple machine learning models in addition to XGBoost, such as LSTMs or fully-connected neural networks and compare their performance.

Acknowledgments

The authors thank Leidos, and Pacific Northwest National Laboratory for providing the Twitter, YouTube, and GDELT data. This work is partially supported by DARPA and Air Force Research Laboratory via contract FA8650-18-C-7825.

References

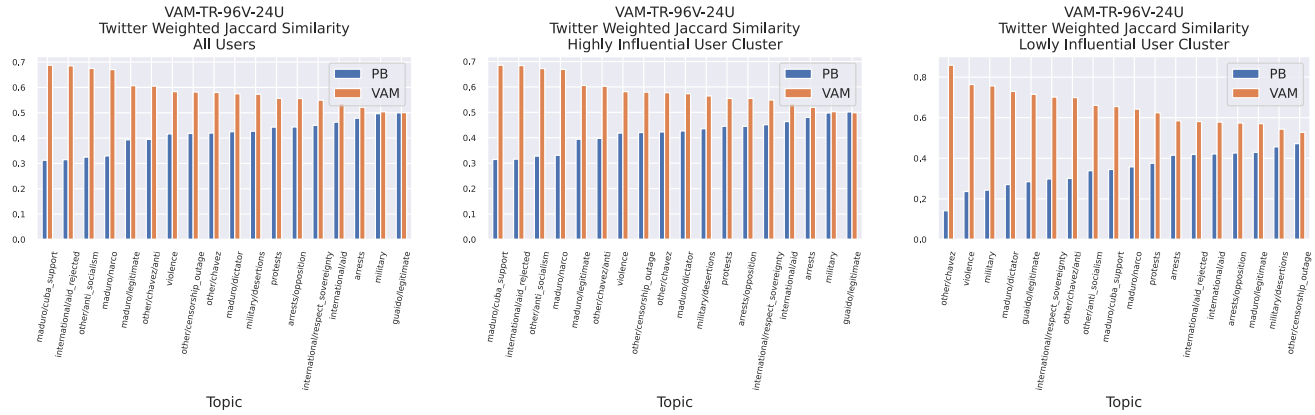
[1] J. Bright, S. Hale, B. Ganesh, A. Bulovsky, H. Margetts, and P. Howard, "Does campaigning on social media make a difference? evidence from candidate use of twitter during the 2015 and 2017 u.k. elections," *Communication Research*, vol. 47, 10 2017.

[2] S. Tasnim, M. Hossain, and H. Mazumder, "Impact of rumors or misinformation on coronavirus disease (covid-19) in social media," *Journal of Preventive Medicine and Public Health*, vol. 53, 04 2020.

[3] J. Kamps and B. Kleinberg, "To the moon: defining and detecting cryptocurrency pump-and-dumps," *Crime Science*, vol. 7, 11 2018.

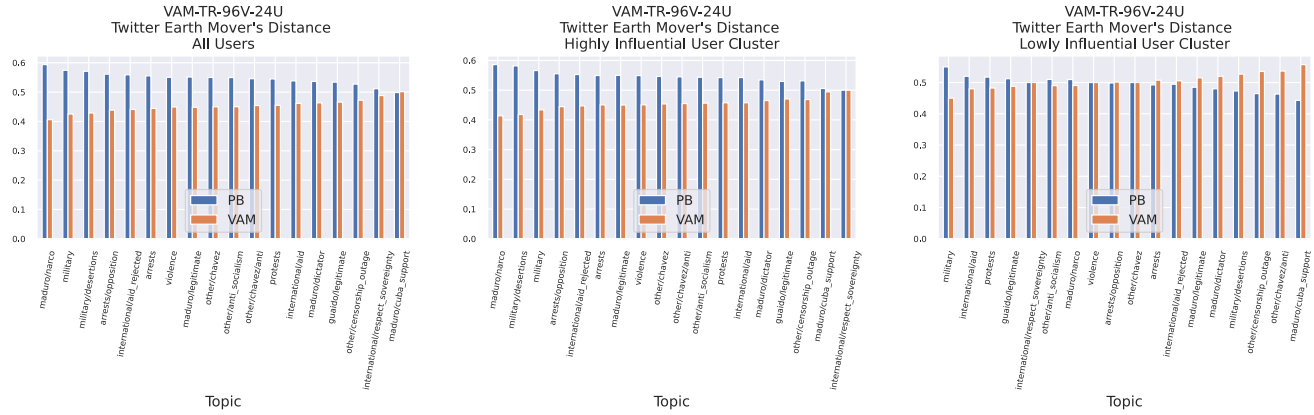
[4] B. Christofaro and S. Baker. A timeline of the political crisis in venezuela, which began with claims of election rigging and has now led to an attempted military coup. [Online]. Available: https://www.businessinsider.com/venezuela-juan-guaido-inside-stratospheric-rise-of-opposition-2019-3

[5] BBC. Venezuela crisis: How the political situation escalated. [Online]. Available: https://www.bbc.com/news/world-latin-america-36319877



(a) All Users - Weighted JS (Higher is Better) (b) Highly Influential Cluster - Weighted JS (Higher is Better) (c) Lowly Influential Cluster - Weighted JS (Higher is Better)

Figure 3: These barplots show the weighted Jaccard similarity results for the full user set, highly influential cluster, and lowly influential cluster.



(a) All Users - EMD (Lower is Better) (b) Highly Infl. Users - EMD (Lower is Better) (c) Lowly Infl. Users - EMD (Lower is Better)

Figure 4: These barplots show the Earth Mover's Distance for the full set of users, highly influential users, and lowly influential users.

[6] S. Madisetty and M. S. Desarkar, "Social media popularity prediction of planned events using deep learning," in *Advances in Information Retrieval*, D. Hiemstra, M.-F. Moens, J. Mothe, R. Perego, M. Potthast, and F. Sebastiani, Eds. Cham: Springer International Publishing, 2021, pp. 320–326.

[7] M. Jayaram, G. Jayatheertha, and R. Rajpurohit, "Time series predictive models for social networking media usage data: The pragmatics and projections," *Asian Journal of Research in Computer Science*, pp. 37–50, August 2020.

[8] Q. Kong, R. Ram, and M.-A. Rizoiu, "Evently: Modeling and analyzing reshare cascades with hawkes processes," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, ser. WSDM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1097–1100. [Online]. Available: https://doi.org/10.1145/3437963.3441708

[9] N. H. Bidoki, A. V. Mantzaris, and G. Sukthankar, "An lstm model for predicting cross-platform bursts of social media activity," *Information*, vol. 10(12), pp. 1–13, 2019.

[10] S. Yao, Y. Hao, D. Liu, S. Liu, H. Shao, J. Wu, M. Bamba, T. Abdelzaher, J. Flamino, and B. Szymanski, "A predictive self-configuring simulator for online media," in *2018 Winter Simulation Conference (WSC)*, 2018, pp. 1262–1273.

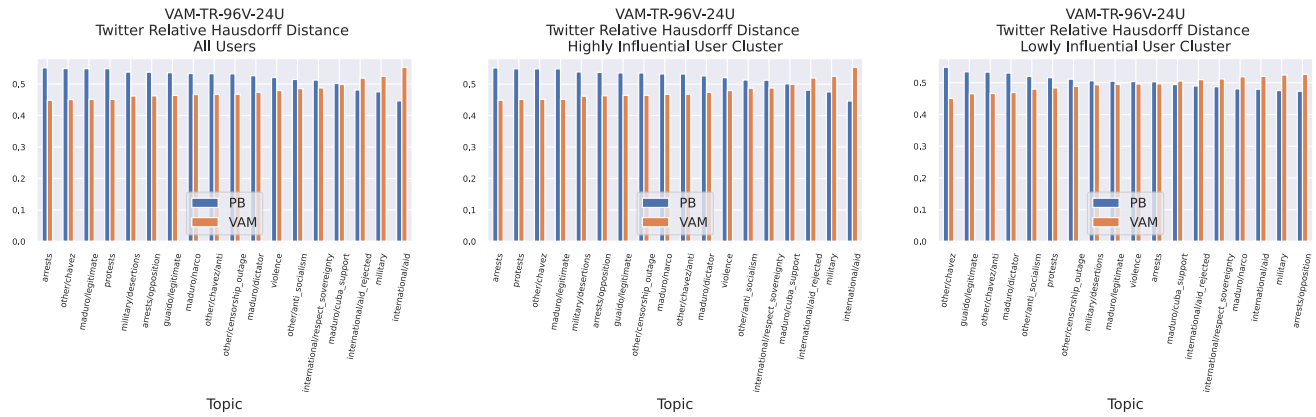
[11] R. Liu, F. Mubang, L. O. Hall, S. Horawalavithana, A. Iamnitchi, and J. Skvoretz, "Predicting longitudinal user activity at fine time granularity in online collaborative platforms," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2019, pp. 2535–2542.

[12] S. Horawalavithana, A. Bhattacharjee, R. Liu, N. Choudhury, L. O. Hall, and A. Iamnitchi, "Mentions of Security Vulnerabilities on Reddit, Twitter and GitHub," in *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI'19)*, Thessaloniki, Greece, Oct 2019.

[13] S. Saadat, C. Gunaratne, N. Baral, G. Sukthankar, and I. Garibay, "Initializing agent-based models with clustering archetypes," in *Social, Cultural, and Behavioral Modeling*, R. Thomson, C. Dancy, A. Hyder, and H. Bisgin, Eds. Cham: Springer International Publishing, 2018, pp. 233–239.

[14] N. Hajiakhoond Bidoki, M. Schiappa, G. Sukthankar, and I. Garibay, "Modeling social coding dynamics with sampled historical data," *Online Social Networks and Media*, vol. 16, p. 100070, 03 2020.

[15] G. Chen, Q. Kong, N. Xu, and W. Mao, "Npp: A neural popularity prediction model for social media content," *Neurocomputing*, vol. 333, pp. 221–230, 2019. [Online]. Available: https://www.sciencedirect.com/



(a) All Users - RHD (Lower is Better) (b) Highly Infl. Users - RHD (Lower is Better) (c) Lowly Infl. Users - RHD (Lower is Better)

Figure 5: These barplots show the Relative Hausdorff Distance for the full set of users, highly influential users, and lowly influential users.

science/article/pii/S0925231218314942

[16] R. Liu, F. Mubang, and L. O. Hall, "Simulating temporal user activity on social networks with sequence to sequence neural models," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020, pp. 1677–1684.

[17] A. Hernandez, K. W. NG, and A. Iamnitchi, "Using Deep Learning for Temporal Forecasting of User Activity on Social Media: Challenges and Limitations," in *Proceedings of Temporal Web Analytics Workshop, Companion Proceedings of The 2020 World Wide Web Conference (TempWeb'20)*, Taipei, Taipei, April 2020.

[18] P. Shrestha, S. Maharjan, D. Arendt, and S. Volkova, "Learning from dynamic user interaction graphs to forecast diverse social behavior," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, ser. CIKM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2033–2042. [Online]. Available: https://doi.org/10.1145/3357384.3358043

[19] I. Garibay, T. A. Oghaz, N. Yousefi, E. C. Mutlu, M. Schiappa, S. Scheinert, G. C. Anagnostopoulos, C. Bouwens, S. M. Fiore, A. Mantzaris, J. T. Murphy, W. Rand, A. Salter, M. Stanfill, G. Sukthankar, N. Baral, G. Fair, C. Gunaratne, N. B. Hajiakhoond, J. Jasser, C. Jayalath, O. Newton, S. Saadat, C. Senevirathna, R. Winter, and X. Zhang, "Deep agent: Studying the dynamics of information spread and evolution in social networks," 2020.

[20] J. Blythe, E. Ferrara, D. Huang, K. Lerman, G. Murić, A. Sapienza, A. Tregubov, D. Pacheco, J. Bollenbacher, A. Flammini, P.-M. Hui, and F. Menczer, "The darpa socialsim challenge: Massive multi-agent simulations of the github ecosystem," in *AAMAS*, 2019.

[21] G. Murić, A. Tregubov, J. Blythe, A. Abeliuk, D. Choudhary, K. Lerman, and E. Ferrara, "Massive cross-platform simulations of online social networks," ser. AAMAS '20. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2020, p. 895–903.

[22] P. Goyal, S. R. Chhetri, and A. Canedo, "dyngraph2vec: Capturing network dynamics using dynamic graph representation learning," in *Knowledge-Based Systems,Volume 187*, January 2020.

[23] U. Singer, I. Guy, and K. Radinsky, "Node embedding over temporal graphs," in *Proceedings of the 28th International Joint Conference on AI (IJCAI-19)*, August 2019.

[24] D. Dunlavy, T. Kolda, and E. Acar, "Temporal link prediction using matrix and tensor factorizations." in *ACM Trans. Knowl. Discov. Data (TKDD) 5(2)*, 2011.

[25] X. Ma, P. Sun, and G. Qin, "Nonnegative matrix factorization algorithms for link prediction in temporal networks using graph communicability," *Pattern Recognition*, vol. 71, p. 361–374, 2017.

[26] S. Gao, L. Denoyer, and P. Gallinari, "Temporal link prediction by integrating content and structure information," *In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, p. 1169–1174, 2011.

[27] P. Sarkar, D. Chakrabarti, and M. Jordan, "Nonparametric link prediction in large scale dynamic networks," *Electronic Journal of Statistics*, vol. 8, pp. 2022–2065, 2014.

[28] N. Ahmed and L. Chen, "An efficient algorithm for link prediction in temporal uncertain social networks." *Information Science*, vol. 331, pp. 120–136, 2016.

[29] F. Mubang and L. Hall, "*VAM*: Supplemental materials," 2021. [Online]. Available: https://fmubang.github.io/pdfs/VAM_Supplemental_Materials_3_19_IEEE.pdf

[30] T. Chen and C. Gestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 2016, pp. 785–794.

[31] S. O'Grady. The u.s. says maduro is blocking aid to starving people. the venezuelan says his people aren't beggars. [Online]. Available: https://www.washingtonpost.com

[32] K. Leetaru and P. A. Schrodt, "Gdelt: Global data on events, location, and tone," *ISA Annual Convention*, 2013.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[34] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[35] D. Doane and L. Seward, "Measuring skewness: A forgotten statistic?" *J. Stat. Educ.*, vol. 19, 07 2011.

[36] S.-H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," *Int. J. Math. Model. Meth. Appl. Sci.*, vol. 1, 01 2007.

[37] S. Brin and L. Page., "The anatomy of a large-scale hypertextual web search engine," *Computer networks and ISDN systems*, pp. 30(1–7):107–117, 1998.

[38] Y. Rubner, C. Tomasi, and L. J. Guibas, "A metric for distributions with applications to image databases." in *IEEE Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, 1998, pp. 59–66.

[39] S. G. Aksoy, K. E. Nowak, E. Purvine, and S. J. Young, "Relative hausdorff distance for network analysis," *Appl Netw Sci 4*, p. 80, 2019.

[40] O. Simpson, C. Seshadhri, and A. McGregor., "Catching the head, tail, and everything in between: A streaming algorithm for the degree distribution." *2015 IEEE International Conference on Data Mining*, pp. 979–984, 2015.