VAM: Supplemental Materials

Fred Mubang and Lawrence O. Hall

Supplemental information to the main VAM paper

University of South Florida
Department of Computer Science

Email: fmubang@usf.edu (Fred Mubang), lohall@usf.edu (Lawrence O. Hall)

CONTENTS

Twitter Avg. New and Old User Frequencies Per Day		
Topic	**New User Avg. Freq. (%)**	**Old User Avg. Freq. (%)**
other/censorship_outage	62.22	37.78
other/anti_socialism	52.15	47.85
military/desertions	43.94	56.06
maduro/cuba_support	41.61	58.39
international/aid_rejected	41.53	58.47
maduro/narco	39.21	60.79
other/chavez/anti	38.24	61.76
other/chavez	27.55	72.45
maduro/dictator	26.34	73.66
arrests/opposition	23.7	76.3
maduro/legitimate	22.84	77.16
arrests	21.88	78.12
international/respect_sovereignty	21.74	78.26
guaido/legitimate	20.64	79.36
international/aid	20.49	79.51
protests	20.43	79.57
violence	18.2	81.8
military	15.27	84.73

Table I: Twitter Avg. New and Old User Frequencies Per Day

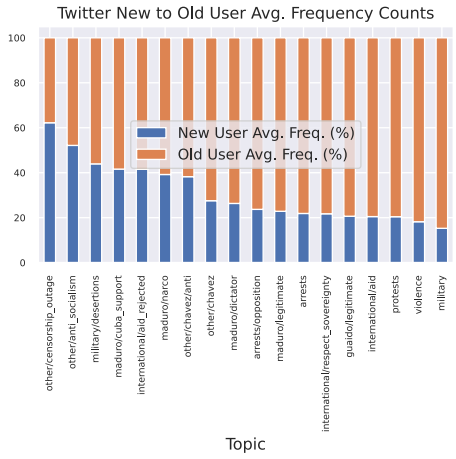


Figure 1: The new/old user proportion plots for Twitter.

I. Introduction

This document contains supplementary material for the VAM paper.

II. Additional Data Collection Information

A. New and Old User Average Daily Frequencies

Table I shows the percentages and Figure 1 shows the bar plot for the average daily new and old user ratio per topic. These ratios were obtained by calculating the average number of new users per day, and then calculating the average old users per day, per topic. These values were then normalized between 0 and 100%. In Figure 1, the orange bars represent the old user average frequencies, and the blue bars represent the new user average frequencies. In the plot it can be seen that on average per day for each topic, most of the users are old, however for some topics there are a considerable amount of new users. For example, the *other/censorship_outage* topic has about 60% new users on average per day, and the *other/anti_socialism* topic has roughly 50% new users on average per day.

B. Twitter Topic Graph Information

Table II on the next page contains the network statistics for all 18 topic networks in Twitter.

III. Recurrent Neural Network Additional Information

A. Hyperparameters

The hyperparameters of the RNN are as follows. All 4 RNNs had a batch size of 32, MSE loss function, 0.0001 learning rate, and RMSProp optimizer. Dropout layers with a rate of 20% were used as well. All RNN models had their epochs set to 100, with a *patience* parameter set to 10 epochs. Every epoch, the model's MSE loss was evaluated on the validation set. If this validation loss did not decrease for 10 epochs in a row, the model would stop training.

B. Architectures

Figures 10 on page 15, 11 on page 16 8 on page 13, and 9 on page 14 show the architectures for the VAM-GRU-TR-96, VAM-LSTM-TR-96, VAM-Bi-GRU-TR-96, and VAM-Bi-LSTM-TR-96 models, respectively. These models were made with Keras [1].

As seen in the diagrams, the input to each model was a tensor with the dimensions (?, 96, 7). In Keras, the question mark (?) simply refers to the batch size of the model, which was 32. The 96 represents the length of the input time series (96 hours). The 7 refers to the 7 features used in each TR model:

1) number of Twitter topic-level new users
2) number of Twitter topic-level old users
3) number of Twitter topic-level activities
4) number of Twitter global (all-topics) new users
5) number of Twitter global old users
6) number of Twitter global activities
7) number of Reddit activities

The final output of the model is a tensor with dimensions (?, 72). Once again, the question mark refers to the batch size of 32. The 72 refers to the total number of outputs: 24 hours * 3 output-types (new users, old users, activities).

Also, each model has a *static_input* layer with 18 inputs. These 18 inputs represent the 1-hot vector for the 18 topics of the Venezuela dataset. These static feature vectors are concatenated with the recurrent layer output in each model.

IV. State-of-the-Art Baselines Methodology

In this section we discuss the experimental setup and methodology for the state-of-the-art baseline models: *tNE-DeepWalk*, *tNE-node2vec-H*, and *tNE-node2vec-S*. As mentioned in the main paper, the 3 baselines used in this work are node embedding methods.

Twitter Graph Information			
Topic	**# Nodes**	**#Edges**	**Total Edge Weight (a.k.a. Total # of Activities)**
military	457,200	2,458,703	4,580,984
international/aid	484,405	2,018,902	3,530,265
protests	451,542	2,058,608	3,083,175
violence	400,141	1,957,442	3,031,137
guaido/legitimate	355,381	1,437,221	2,122,211
international/respect_sovereignty	205,180	815,250	1,635,717
maduro/dictator	355,552	1,137,656	1,528,799
other/chavez	222,025	697,542	1,154,887
arrests	175,685	687,628	935,191
arrests/opposition	147,454	551,617	718,539
international/aid_rejected	211,168	518,668	662,886
maduro/legitimate	94,424	351,705	655,588
other/chavez/anti	142,556	300,346	398,892
military/desertions	125,257	285,934	365,718
maduro/narco	92,208	190,973	244,958
other/anti_socialism	119,519	184,152	238,342
maduro/cuba_support	62,904	112,281	153,640
other/censorship_outage	62,603	110,097	122,581

Table II: Twitter Network Statistics

A. tNodeEmbed Additional Background Information

As mentioned in the main paper, the tNodeEmbed embeddings used in this work are initialized using the *DeepWalk* and *node2vec* embedding methods.

The *DeepWalk* algorithm uses random walks to create latent representations of nodes within a graph. These random walks are then used to create "sentences" that are fed into a *Word2Vec* embedding algorithm. *Node2vec* is a variation of *DeepWalk* that introduces two new parameters, p and q that can be used to bias the random walks. When $p = q = 1$, the node2vec embedding is the same as a DeepWalk embedding, meaning the random walks have not been biased in any way. When $p=1$ and q=0.5, the node2vec embeddings are considered *Homophilic*, meaning that nodes that are close to each other have similar embeddings. When p=1 and $q=2$, the embeddings are *Structural*, meaning nodes with a similar role in the network (such as being a "hub node"), will have similar embeddings to one another [2]. In this work, we use the terms *node2vec-H* and *node2vec-S* to refer to our node2vec models that use the Homophilic (p=1, q=0.5) and Structural (p=1, q=2) parameters, respectively.

B. Initial DeepWalk and Node2Vec Embeddings

Before creating tNodeEmbed embeddings, we had to firstly create DeepWalk and node2vec embeddings. We call these initial embeddings DeepWalk, node2vec-H, and node2vec-S models.

Each node embedding represents a user and topic on a particular day, or a *(user, topic, day)* tuple. Embeddings were made for the time period spanning February 12th up to March 7th, 2019 (24 days). The parameters of the node2vec/DeepWalk embeddings are as follows. The dimensions of each embedding were set to 32. The number of random walks per node was set to 10. The walk length of each random walk was set to 200. These values were found via grid search. The dimensions tried were 8, 16, 32, and 64. The walks tried were 10 and 50. The lengths tried were 50 and 200.

C. Creating the tNodeEmbed Embeddings

After creating the DeepWalk, node2vec-H, and node2vec-S embeddings, the next step was to create the tNodeEmbed embeddings. These embeddings were made by performing an "alignment" operation on the aforementioned node2vec and DeepWalk embeddings as discussed in [3]. In other words, the node embeddings on day 2 are aligned with the embeddings on day 1, and the embeddings on day 3 are aligned with day 2, etc. By performing this operation, performance is improved on downstream tasks as shown in [3]. The new embeddings created from this operation are known as "tNodeEmbed" embeddings. The newly aligned *DeepWalk* embeddings are referred to as *tNE-DeepWalk*. The newly aligned *node2vec-H* embeddings are referred to as *tNE-node2vec-H*, and the newly aligned *node2vec-S* embeddings are referred to as *tNE-node2vec-S*. For more details on how the alignment operation works, refer to [3].

D. Training and Testing Into Neural Networks

Once all 3 sets of user embeddings are created, they can then be used to perform prediction tasks with a neural network. The prediction task was as follows: given a *(child, parent, topic, day)* tuple (represented as a vector), predict the number of topic-related tweets/retweets that the child-parent edge will post over the next 24 hours.

To represent a child-parent edge, the node embeddings for the child and parent were concatenated with one another. Recall that the number of dimensions used for each node embedding was 32. Since, this was the case, the concatenated child-parent vector had 64 values. The output vector represented the time series, and had 24 values.

Three fully-connected neural networks were trained (1 for each embedding approach). The learning rate was 0.0001. The batch size was 32. The number of epochs was set to 100. Early stopping was used with a "patience" parameter set to 10. An Adam optimizer was used.

Each network was comprised of 3 hidden layers with 100, 50, and 25 hidden units respectively. Dropout layers were used after each layer with the rate set to 20%. The activation function used after each hidden layer was tanh. The linear activation function was used in the output layer. MinMax scaling was used to normalize the features.

The loss function used was *Weighted Cumulative MSE*. It is a variation of the *MSE* loss function that performs the MSE operation on the cumulative sum of the ground truth and prediction vectors. We took the cumulative sum because it does not punish the model as harshly for predicting the "correct value" but at the "wrong timestep". In our initial experiments we used MSE as a loss function, but our models predicted all 0's. We believe this is because most users perform no actions most of the time, so a model trying to minimize loss is incentivized to predict only 0's. Since MSE requires "exact timing", it exacerbates the tendency of a model to do this.

Furthermore, we weighted our loss function so that incorrectly predicting a 0 when an action did occur would incur a greater penalty. We used a cost parameter we call *C* and it was set to 2. So in other words, the loss would be twice as large for predicting a 0 instead of a non-zero value where appropriate.

E. Data Splits

Table III shows how the training, validation, and test datasets were created. Recall that the test period is comprised of 21 days (Feb 15th to March 7th). For each of these 21 days, for each of the 3 different embedding approaches, a fully-connected neural network was trained on a training period of 3 days preceding the test day of interest, and then tested upon the test day of interest. We used only 3 days for training data because we tried multiple days between 1-4, and found 3 to yield the best results during validation. Furthermore, since there are so many edges in a given day, a few days is all that is needed in order to generate the tens of thousands of samples sufficient to train a neural network properly.

For validation, the samples in each training period were split in an 85%/15% ratio for train/validation sets. For example, in test period 1, there were 616,881 total samples (edges) between the train period of Feb 12-14, 2019. After performing the split, 524,348 of these samples went into the training set, and 92,533 were used for validation. A neural network was then trained and validated on these edges, and tested with the test set of 83,095 samples.

F. Creating the Test Day Initial Conditions

An important decision to make is figuring out how many samples, or user-to-user edges, one will use in the test set for prediction. A simple solution would be to use the entire historical dataset of edges as the initial condition set (from Dec 28th 2018, the start of our dataset to the present). However, since there are so many edges in this timespan, that would be very expensive in terms of computational time and space.

We decided upon using a lookback period of 1 day (24 hours) to generate the initial condition period. For example, the 83,095 initial condition edges in the 1st test period (February 15th) is the set of all active edges from the previous day (February 14th). The assumption here is that the active future edges will be similar to the recently past active edges. We used this lookback period of 1 day because it is the same lookback period we used for the VAM user-assignment module. This allows for a fair comparison between the VAM and node-embedding approaches.

Something that one must keep in mind is that the initial condition edges are from the previous day of ground truth edges. The true number of active edges is obviously not "known" until that day has passed. This is why the number of some of the test edges in Table III exceeds that of the training period edges.

For example, on test period 10 in the table, the number of edges in the training set is about 1.9 million, while the test set has about 2.2 million initial condition edges. This is because the final day of training for this test period in February 23rd. What this means is that during this training period, the model is taking as input in the set of old edges that exist up to February 22nd, and predicting whether or not they will be active on Febrary 23rd. Obviously, on February 22nd, the model will not "know" of new edges that will exist on February 23rd. It can only predict using the set of previously existing edges. It is not until the initial condition day of February 23rd that the model will "know" the full set of new and old edges it can use to predict the activity for the next day of February 24th.

G. Predicting New Users

The embedding methods do not have the ability to predict new users, so in order to incorporate new users into our predictions, within each daily temporal graph we created a node called "New User" which encapsulates all the activities that all new users would perform within a given day. For example, let us say for some daily temporal graph, G^{day}, all of the new users perform a total of 30,000 actions. Then we create a node for that day called "New User" that performs 30,000 actions. We found that adding this node to each daily graph improved prediction results for the tNodeEmbed models.

Even though this "New User Node" predicts how many activities new users will perform within a given day, we still need to assign these actions to actual new users. We do so in the following way.

Firstly, we predict the number of new users within a given day using the Persistence Baseline. So, if the Persistence Baseline predicted 10,000 new users, we use 10,000 new users as our prediction of the next day. We then generate new users and assign them different identifier strings.

Then, we randomly assign our prediction activities from the "New User" node to these newly generated users. So, let us say that the Persistence Baseline predicted 10,000 new users, and our "New User" node from the tNodeEmbed model predicted 30,000 new user activities. We then randomly assign these 30,000 activities to the 10,000 newly generated users.

V. Volume Module Additional Methodology Information

This section contains additional implementation details from VAM's Volume Prediction Module.

A. Feature Configuration By Platform and Volume Lookback Factor

As mentioned in the paper, the time series features differed across all VAM models. Table IV on page 7 describes the 7 potential time series features for a given sample, and Table V on page 8 describes which model had which time series features.

To better understand how the features are configured we shall describe an example in Table V on page 8. Take, for example, the fourth row for the model, *VAM-TR-96*. This is a *VAM* model trained on Twitter and Reddit time series features. The *Time Series Used* column illustrates which time series were fed in from Table IV on page 7. It says that features 1-7 were used. If you look at Table IV on page 7 you will see that these are all time series related to Twitter and Reddit, which explains the "TR" in the model tag. Note that each platform in V on page 8 is represented with a letter. "T" stands for "Twitter", "R" stands for "Reddit".

The *volume lookback factor* column for *VAM-TR* indicates it's "96 hours". So, for each time series category listed in the *time series used* category, a time series of 96 elements is placed into the feature set for the *VAM-TR*. Since there are 7 time series, a *volume lookback factor* of 96, and 18 topic features, the calculation for number of features is 7 * 96 + 18, which equals 690. Therefore, the dataset for the *VAM-TR* model was comprised of 690 features (as shown by the *Total Features* column).

B. Log Normalization in Volume Prediction Module

For all Twitter-related features, we rescaled the data by first taking the natural log of all samples twice. Before taking the logs, we added 1 to all values in order to avoid taking the natural log of 0. For the Reddit features, we only took the natural log once, because the magnitude of those features was not as large as the Twitter ones.

VI. Volume Module Additional Prediction Results

This section contains additional results for the Volume Prediction Module.

A. Volume Prediction Module Results Across Topics

Figure 3 on page 7 contains bar plots comparing the *VAM-XGB-TR-96* model against the best baseline model (Persistence Baseline) per each topic and metric pair. Tables VI on page 18, VII on page 18, VIII on page 18, IX on page 18, X on page 19, and XI on page 19 show per-topic results for the MAE, NC-RMSE, RMSE, S-APE, SkE, and VE metrics, respectively.

For the RMSE metric, VAM won against the best baselines on 18 out of 18 topics. For MAE, VAM won 17 out of 18 times; for Normalized Cumulative RMSE (NC-RMSE), VAM won 17 out of 18 times; for Symmetric Absolute Percentage Error (S-APE), VAM won 14 out of 18 times; for Skewness Error (SkE), VAM won 11 out of 18 times; and for Volatility Error (VE), VAM won 18 out of 18 times.

Overall, VAM outperformed the Persistence Baseline 97 out of 108, or 89.8% of the time. VAM performed particularly well at the "volume over time" metrics (RMSE MAE, and NC-RMSE), as well as the volatility metric (VE). It performed decently for the "magnitude" or "scale" metric (S-APE). Lastly, it struggled the most with the Skewness Error metric, which measures the asymmetry of the time series.

B. Issues with Using Only RMSE and MAE as Metrics

RMSE and MAE are commonly used metrics for time series regression problems, however we note their limitations. When plotting the VAM models against the baseline models, we found that there are some instances in which the baseline time series has better RMSE and MAE results than the VAM prediction, however when visually inspecting the time series plots, the VAM models seem to better match the ground truth time series. For this reason, we also utilized 4 more metrics that measure other elements of time series prediction performance besides just the "exact timing" measurement of RMSE and MAE. Figure 2 on the next page contains 2 examples of this phenomenon.

VII. User-Assignment Additional Methodology Information

A. User-Assignment: Various User Definitions

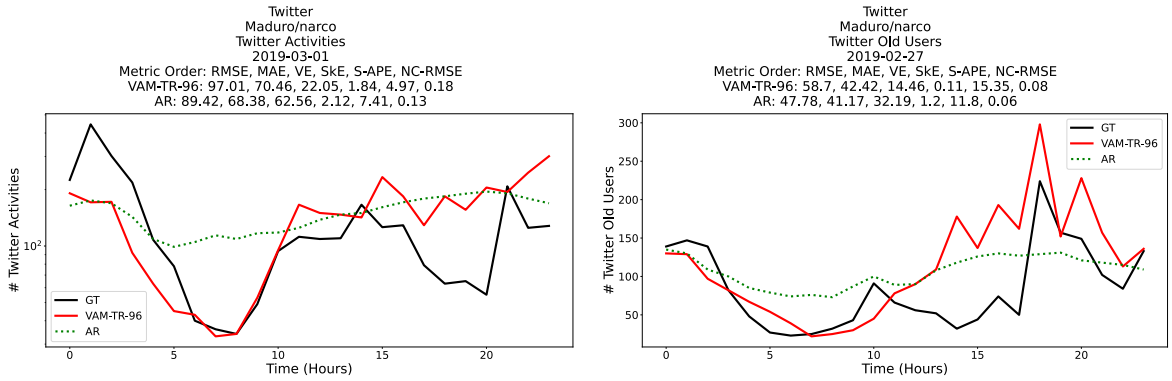
In order to better understand how the User-Assignment algorithm works, we further describe the differences between *new* and *old* users, as well as *child* and *parent* users.

B. Child and Parent Users

For any given edge, $(u, v, w(u, v, t))$, user u is the *child user*. This means that u either created a post (such as a tweet), or u reacted to a post created by user v. Likewise for an edge, $(u, v, w(u, v, t))$, user v is the *parent* user. This means that v has had her post reacted to in some way by u. Note that if u creates a post, she is both a child and parent user. In the edge list, E_t, this would would be a self loop, which could be written as $(u, u, w(u, u, t)) == (u, v, w(u, v, t))$.

Embedding Neural Network Dataset Info							
Test Period	**Train Start**	**Train End**	**Test Day**	**Total Train and Val Samples**	**Training Samples**	**Validation Samples**	**Test Samples**
1	2019-02-12	2019-02-14	2019-02-15	616,881	524,348	92,533	83,095
2	2019-02-13	2019-02-15	2019-02-16	592,584	503,696	88,888	121,373
3	2019-02-14	2019-02-16	2019-02-17	386,265	328,325	57,940	189,394
4	2019-02-15	2019-02-17	2019-02-18	393,862	334,782	59,080	175,432
5	2019-02-16	2019-02-18	2019-02-19	486,199	413,269	72,930	205,704
6	2019-02-17	2019-02-19	2019-02-20	570,530	484,950	85,580	293,092
7	2019-02-18	2019-02-20	2019-02-21	674,228	573,093	101,135	329,941
8	2019-02-19	2019-02-21	2019-02-22	828,737	704,426	124,311	452,037
9	2019-02-20	2019-02-22	2019-02-23	1,075,070	913,809	161,261	1,080,346
10	2019-02-21	2019-02-23	2019-02-24	1,862,324	1,582,975	279,349	2,161,546
11	2019-02-22	2019-02-24	2019-02-25	3,693,929	3,139,839	554,090	908,470
12	2019-02-23	2019-02-25	2019-02-26	4,150,362	3,527,807	622,555	379,520
13	2019-02-24	2019-02-26	2019-02-27	3,449,536	2,932,105	517,431	210,723
14	2019-02-25	2019-02-27	2019-02-28	1,498,713	1,273,906	224,807	184,576
15	2019-02-26	2019-02-28	2019-03-01	774,819	658,596	116,223	229,731
16	2019-02-27	2019-03-01	2019-03-02	625,030	531,275	93,755	91,076
17	2019-02-28	2019-03-02	2019-03-03	505,383	429,575	75,808	92,142
18	2019-03-01	2019-03-03	2019-03-04	412,949	351,006	61,943	136,468
19	2019-03-02	2019-03-04	2019-03-05	319,686	271,733	47,953	146,449
20	2019-03-03	2019-03-05	2019-03-06	375,059	318,800	56,259	77,073
21	2019-03-04	2019-03-06	2019-03-07	359,990	305,991	53,999	96,980

Table III: Embedding Neural Network Dataset Info. This is the train, validation, and test set information for the neural networks trained on the DeepWalk, node2vec, and tNodeEmbed embeddings.



(a) Maduro/Narco Activities on March 1st

(b) Maduro/Narco Old Users on Feb 27th

Figure 2: Here are some examples in which a baseline model had better RMSE and MAE performance than the *VAM-XGB-TR*, but worse performance in other metrics. In (a), the RMSE and MAE of the AR model is 89.42 and 68.38, respectively, whereas for VAM it's 97.01 and 70.46, respectively. However, as one can see, visually the VAM model prediction (red) looks more similar to the ground truth (black) curve within the first 15 hours or so of the simulation. The VAM prediction manages to capture the major dip in the ground truth, unlike the AR prediction. This might be captured in the prediction metrics in which VAM had a Volatility Error and Skewness Error of 22.05 and 1.84, versus the AR model's results of 62.56 and 2.12, respectively. In 2b, a similar phenomenon can be observed. The AR model has better RMSE and MAE metrics than VAM (47.78 and 41.17 vs. VAM's 58.7, and 42.42, respectively), however, VAM has better VE and SkE metric results (VAM has 14.46 and 0.11 vs. the AR's 32.19 and 1.11, respectively).

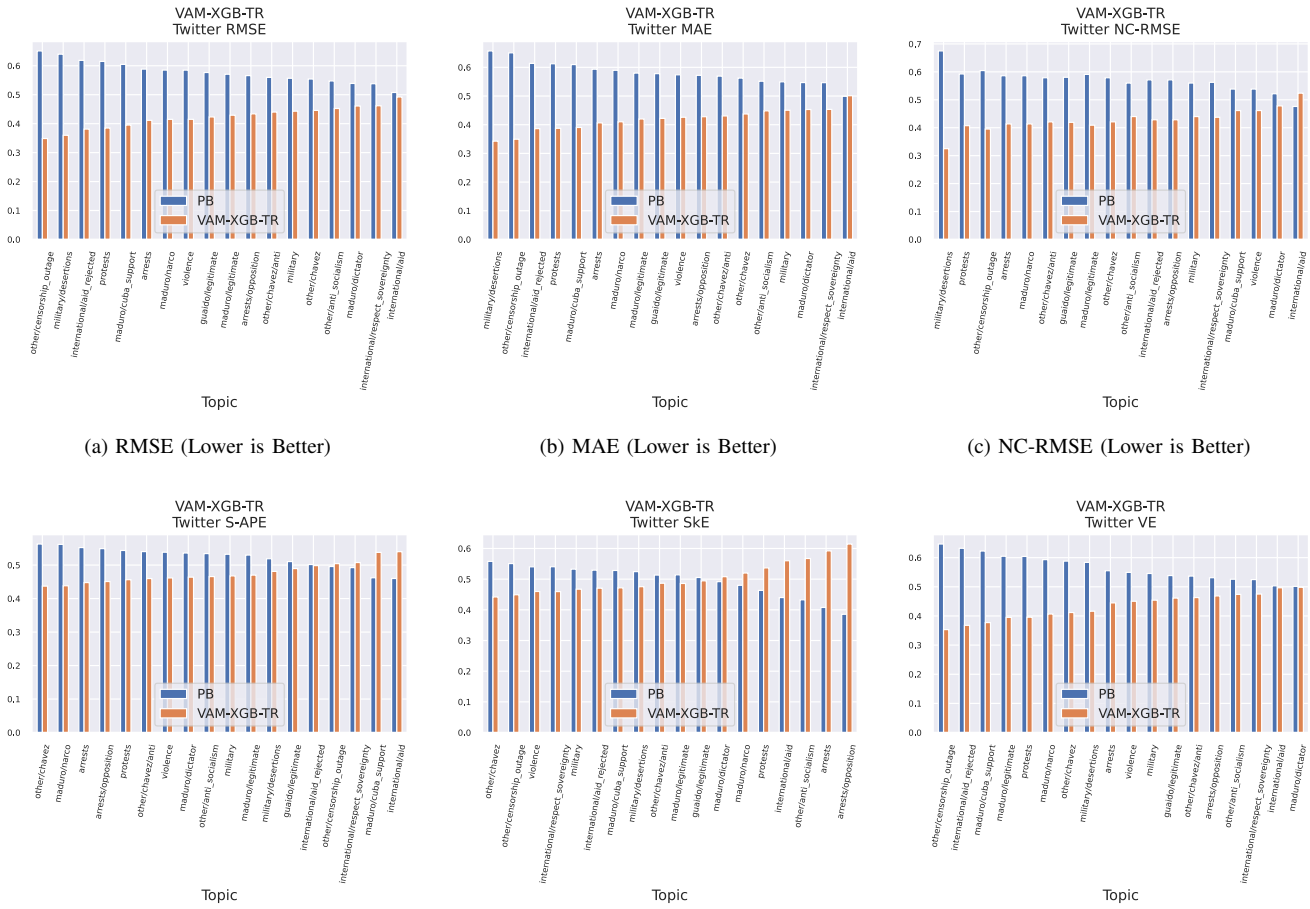
C. New and Old Users

An *old* and *new* user can be described as follows. A user, u is *old* at time step t if u appears in G_t in the temporal graph sequence G, and it also appeared in at least one of the previous graphs spanning G_1 up to G_{t-1}.

A user, u is *new* at time step t if u appears in G_t in the temporal graph sequence G, but *has not* appeared in any of the previous graphs from G_1 up to G_{t-1}.

D. User-Assignment Symbols

Table XII on page 20 contains various symbols used in the User-Assignment algorithm for quick reference.



(a) RMSE (Lower is Better)

(b) MAE (Lower is Better)

(c) NC-RMSE (Lower is Better)

(d) S-APE (Lower is Better)

(e) Skewness Error (Lower is Better)

(f) Volatility Error (Lower is Better)

Figure 3: These barplots show VAM (orange) vs. the Persistence Baseline, PB, (blue) across various topics. The metric result per topic for both models were normalized between 0 and 1 for easier visualization.

Time Series Index	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	Activity volume time series across all topics in Twitter.
5	New user volume time series across all topics in Twitter.
6	Old user volume time series across all topics in Twitter.
7	Activity volume time series in Reddit.

Table IV: All possible time series feature categories.

E. Additional User-Assignment Implementation Details

If the *VP-Module* predicts more users than actions at some future timestep, $T+1$, then VAM changes the number of actions to equal the number of users. For example, if 2 users and 1 action were predicted to occur at $T+1$, then VAM simply changes the number of predicted actions to be 2.

If, for some future timestep $T+1$, the *VP-Module* predicted more old users than exist in the recent history table, H^{recent}, then VAM simply changes the number of predicted active old users to equal the number of old users that do exist in H^{recent}. For example, if the *VP-Module* predicts 100 old users will be active at $T+1$, but H^{recent} only contains 90 old users, then VAM will change the number of predicted old users to be 90, and then sample all 90 active old users from H^{recent} for user-activity assignment.

VIII. User-Assignment Algorithm Step-by-Step in Detail

In this section we describe in more detail the steps to assign a user to an action via the user-assignment module.

A. User Assignment - Inputs and Outputs

The inputs to the algorithm are 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 $\hat{\boldsymbol{Y}} \in \mathbb{R}^{3\times S}$, the old user index old_idx, the new user index new_idx, and the activity index act_idx. The three indices are used to access the predicted number of old users, new users, and activities on the predicted timestep of interest

VAM Model Feature Configurations						
Model	**Platforms Used**	**Time Series Used**	**Num Time Series Used**	**Volume Lookback Factor**	**Num Static Fts.**	**Total Features**
VAM-T-96	('Twitter',)	(1, 2, 3, 4, 5, 6)	6	96	18	6 * 96 + 18 = 594
VAM-T-72	('Twitter',)	(1, 2, 3, 4, 5, 6)	6	72	18	6 * 72 + 18 = 450
VAM-T-48	('Twitter',)	(1, 2, 3, 4, 5, 6)	6	48	18	6 * 48 + 18 = 306
VAM-TR-96	('Twitter', 'Reddit')	(1, 2, 3, 4, 5, 6, 7)	7	96	18	7 * 96 + 18 = 690
VAM-TR-72	('Twitter', 'Reddit')	(1, 2, 3, 4, 5, 6, 7)	7	72	18	7 * 72 + 18 = 522
VAM-TR-48	('Twitter', 'Reddit')	(1, 2, 3, 4, 5, 6, 7)	7	48	18	7 * 48 + 18 = 354

Table V: The feature configurations for each VAM model.

from $\hat{\boldsymbol{Y}}$. The output of the *Assign_Users* algorithm is the temporal graph sequence, $\hat{G}^{future}$.

B. Initializations

At the beginning of the algorithm, G^{recent} is constructed with the $Get\_Recent\_Temporal\_Graph$ function using the full temporal graph, G, and the lookback parameter, L^{user}. As mentioned in the main paper, the lookback factor parameter L^{user} is used to determine the number of snapshots to use. For example, if $L^{user} = 24$, then only the 24 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.

Furthermore, note that the length of the full temporal graph sequence G is T. So, for example, if G contains $T = 100$ graphs, and $L^{user} = 5$, then only graphs 96 up to 100 will be in the sequence G^{recent}. Or, in other words, only graphs $T - L^{user} + 1$ up to T are included in G^{recent}. Then, an empty array, $\hat{G}^{future}$ is created.

Recall that the number of prediction timesteps of interest is S. There is a for loop that iterates S times for each $s \leq S$. This s represents the current prediction timestep of interest. At each timestep s, the number of old users, new users, and activities are retrieved from their respective location in the $\hat{\boldsymbol{Y}}$ matrix.

C. The History Table

The sequence G^{recent} is used to construct a history table, H^{recent}, that contains the event tuples from G. For each iteration s, the *UA-Module* first uses G^{recent} to construct a recent history table called H^{recent}. This table can be thought of as a hash table. Each key into the table is an integer representing a time step, t, spanning from $T - L^{user} + s$ up to $T + s - 1$. This range can be also be defined as the *current lookback period of interest* Each time step key maps to an event table, H_t^{recent}. This table can be thought of as an array of "event tuples", each of which with the following form:

$$(u, v, w(u, v, t), \mathbb{I}_{V_t^{new}}(u), \mathbb{I}_{V_t^{new}}(v))$$

. u is the child user and v is the parent user. $w(u, v, t)$ is a numerical value that represents how many times u interacted with v at time step t in G_t.

V_t is the set of all users in G at some particular timestep t. V_t^{new} is the set of all new users in V_t. $\mathbb{I}_{V_t^{new}}$ is an indicator function that returns 1 if a particular user, u was in the new user set, V_t^{new} at time step t, and 0 otherwise.

D. Retrieving Old User Candidates

The *Get_Active_Old_User_Candidates* function is used to construct a table of each old user candidates (W^{old_cand}) from H^{recent} and their likelihood of being active at time s.

This table can be thought of as an array of tuples, each with the following form: $(u, p^{old\_act}(u, T + s))$. The term $p^{old\_act}(u, T + s)$ represents the probability that user u will be active at time step $T + s$. This probability is obtained by calculating, for each user u, the normalized average activity frequency of u during the *current lookback period of interest*.

The assumption here is that a user's future probability of acting is equivalent to his past probability of acting.

E. Retrieving Most Likely Old Users From Candidates

The *Get_Most_Likely_Active_Old_Users* function is then used to retrieve the set of most likely old users ($\hat{O}_s$) from this table of candidates, as well as a table containing their re-weighted probabilities (W^{old}).

In order to retrieve ($\hat{O}_s$) and W^{old}, *VAM* performs a weighted random sample on the W^{old_cand} table in order to create the set of most likely active old users at time $T + s$. The weights used for this random sample are the activity probabilities ($p^{old\_act}(u, T + s)$) for each old user that were calculated in the previous step. We call the set of predicted active users, $\hat{O}_s$. Furthermore, *VAM* creates a new weight table, called W^{old}, which is the same as W^{old_cand} minus any users that were not chosen by the weighted random sample. This new table, W^{old} is needed when the time comes for *VAM* to predict how many actions each old user will perform. Old users with a higher probability of acting, a.k.a. $p^{old\_act}(u, T + s)$, are more likely to perform more actions.

F. Creating the New User Set

Next, the set of new users ($\hat{N}_s$) is generated using the *Generate_New_Users* function. Recall that the number of new users is known because that was predicted from the *Volume-Prediction Module* and it is contained in the matrix $\hat{\boldsymbol{Y}}$.

G. Assigning Attributes to New Users

The question that remains at this point is "How does one decide what actions the new users will perform?" *VAM* does this by constructing a *New User Archetype Table*. This table is comprised of *recently active users*. These are users that have appeared as new within the lookback factor period of $T + s - L^{user}$ up to $T + s - 1$, which can also be referred to

as the "recent history". The opposite of a *recently active* user would be a *long-acting user*, which would be a user who has appeared in G before timestep $T+s-L^{user}$. The assumption behind the archetype table is that new users in the future will behave in a similar manner to previous new users from the recent past.

The *New User Archetype Table* table contains the following information: (1) the name of the *recently active* a.k.a. *archetype user*, (2) the probability that this archetype would be active in any given timestep (e.g. via tweeting or retweeting), and (3) the probability that this archetype will be "influential" in any given timestep. In Twitter, probability of influence is measured by how often a user is retweeted.

With this in mind we now define a *new user archetype* record as follows:

$$(u^{arch}, p^{act\_arch}(u^{arch}), p^{infl\_arch}(u^{arch}))$$

. The term $p^{act\_arch}(u^{arch})$ is a weight that describes how likely it is that a user of archetype, u^{arch} will be active in future time step. The term, $p^{infl\_arch}(u^{arch})$ describes how likely it is that a user of archetype u^{arch} will be influential in some future time step. We define influence as the quantity by which other users will respond to a post created by u^{arch} in some social media platform with regards to topic q.

As an example, say there is some recently active user named Carol with an action probability of 0.2 and an influence probability of 0.1. Since Carol is a recently active user, she will be considered an archetype of new user and added to the new user archetype table, W^{new_arch}. The record for "Carol" will be as follows:

$$(\text{“}Carol'', 0.2, 0.1)$$

. Now, "Carol" archetype could be applied to a new generated user arbitrarily given the identifier "Phil". Even though Phil has been generated as a new user, he still needs to be assigned a probability of activity and probability of influence. *VAM* will randomly sample a user archetype from W^{new_arch} in order to assign attributes to Phil. If *VAM* randomly selects user archetype Carol, then *VAM* will assign Phil an activity probability of 0.2 and an influence probability of 0.1. These values are, of course, then normalized relative to the other generated new users so that all users' probabilities lie between 0 and 1.

VAM then iterates over every new user $u^{new} \in \hat{N}_s$ and performs a weighted random sample to select a new user archetype tuple from W^{new_arch}. This process then yields a new table, called the *new user attribute table*, or W^{new}. This table can be thought of as any array of tuples of the following form.

$$(u^{new}, u^{arch}, p^{act\_arch}(u^{new}), p^{infl\_arch}(u^{new}))$$

u^{new} is the new user of interest. u^{arch} is the archetype that this new user was created from. $p^{act\_arch}(u^{new})$ and $p^{infl\_arch}(u^{new}))$ are u^{new}'s probabilities of activity and influence, respectively. They are equivalent to the probability and influence probabilities of user archetype, u^{arch}. In other words, each user $u^{new} \in \hat{N}_s$ was assigned probability and activity attributes from some user archetype, $u^{arch} \in U^{arch}$. Note that U^{arch} is the set of all user archetypes.

The *New User Archetype Table* is then used to assign the activity and influence probabilities to each new user in $\hat{N}_s$. These probabilities are stored in W^{new}.

H. Creating the Old and New User Parent Tables

Now, *VAM* needs the most likely sets of parents that the old and new users will interact with. To that end, it creates what we call *parent distribution tables*.

Firstly, using the $Create\_Old\_User\_Parent\_Table$ function, the old parent distribution table, D^{old_parent}, is created. This table can be thought of as a hash table, in which each key is a user, u, and each user maps to a parent distribution table for that particular user. Each table has the form: $(v, p^{edge}(u, v, T+s))$. The term $p^{edge}(u, v, T+s))$ represents the probability that an edge will form between u and v at time $T+s$.

Next, VAM must create a parent distribution table for the new users using the *Create_New_User_Parent_Table*. In order to do so, it iterates over every new user record in W^{new}. It then checks which user archetype, u^{arch} the new user u^{new} was created from. Recall that each new user archetype was created from users who have actually existed in the data, so it is possible for *VAM* to collect information regarding who their previous parents were. *VAM* will then create a parent distribution table for each u^{arch} and will assign this parent distribution table to the appropriate new user u^{new}. The final table created, D^{new_parent}, will be a new user parent distribution table, similar to D^{old_parent}. Each key of the hash table is a new user, u^{new}, and each key hashes into a new user parent distribution table of the form $(v, p^{edge}(u^{new}, v, T+s))$. The term $p^{edge}(u^{new}, v, T+s))$ represents the probability that an edge will form between u^{new} and v at time $T+s$.

I. Creating the Links

At this point, *VAM* now has the information it needs to perform link prediction. To that end, it uses the function, $Create\_Links$ to perform link prediction and create the final graph, G_s^{future}. The arguments to $Create\_Links$ are $\hat{O}_s$,$\hat{N}_s$,num_acts , W^{old}, W^{new}, D^{old}, and D^{new}. Note that *VAM* "knows" the total edge weight of all links in G_s^{future} because the *Volume-Prediciton Module* predicted the total number of activities for each timestep $s \leq S$, hence the use of the argument, num_acts.

J. Updating the Recent Temporal Graph Sequence

The predicted graph, $\hat{G}^{future}$ is then used to update G^{recent}. The user-assignment for-loop then continues $S-1$ more times until the full $\hat{G}^{future}$ graph is predicted such that $\hat{G}^{future} = \{\hat{G}_1^{future}, \hat{G}_2^{future}, ..., \hat{G}_S^{future}\}$.

K. User-Assignment Graphic

Figure 12 on page 17 is an illustration of all 7 steps of the User-Assignment Algorithm.

IX. User-Assignment Full Results

This section contains the full results of the User Assignment Module. The main per-topic results from the paper are shown, in addition to clustering analysis that was performed on the results.

A. User Assignment Clustering Results

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.

B. Weighted Jaccard Similarity Results

Figures 4a, 4b, and 4c 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.

Table XVI on page 21 contains the weighted JS results for the full set of old users. Table XVII on page 22 contains the weighted JS results for the highly influential cluster of old users. Table XVIII on page 22 contains the weighted JS results for the lowly influential cluster of old users.

C. Unweighted Jaccard Similarity Cluster Results

We also measured VAM's per-topic performance against the baseline for the Unweighted Jaccard Similarity. On all 3 user clusters, VAM outperformed the Persistence Baseline across all 18 out of 18 topics, for a total of 54 out of 54 (100%) wins.

Table XIII on page 20 contains the unweighted Jaccard Similarity results for the full set of old users. Table XIV on page 20 contains the unweighted JS results for the highly influential cluster of old users. Table XV on page 21 contains the unweighted JS results for the lowly influential user cluster.

D. Earth Mover's Distance Results

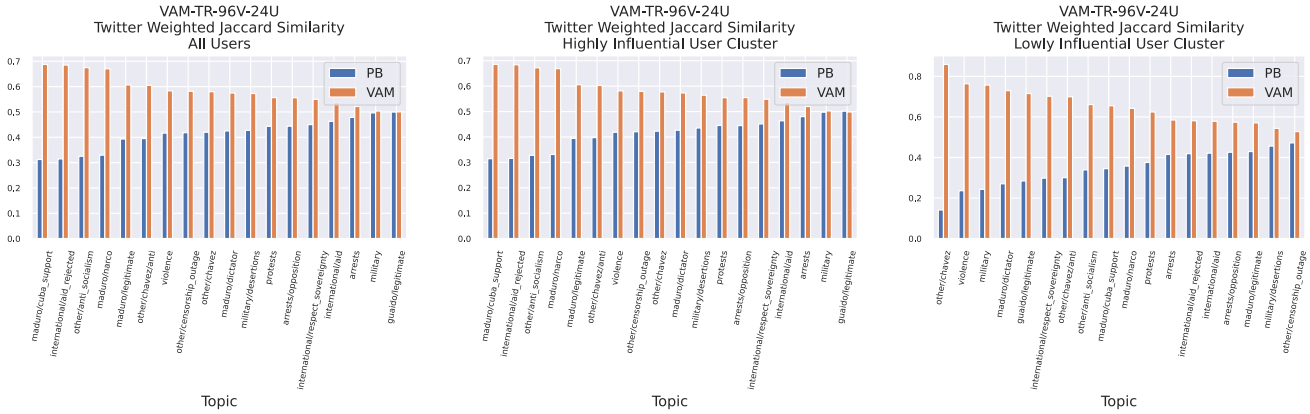
Figure 6 contains the Earth Mover's Distance results for the full user set, highly influential users, and lowly influential users. In these plots, the lower bars are better. For the full user set, VAM outperformed the baseline 17 out of 18 times. For the highly influential cluster, VAM won 18 out of 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.

Tables XIX on page 23, XX on page 23, and XXI on page 23 contain the Earth Mover's Distance results for the full set, highly-influential cluster, and lowly-influential cluster of users.

E. Relative Hausdorff Distance Results

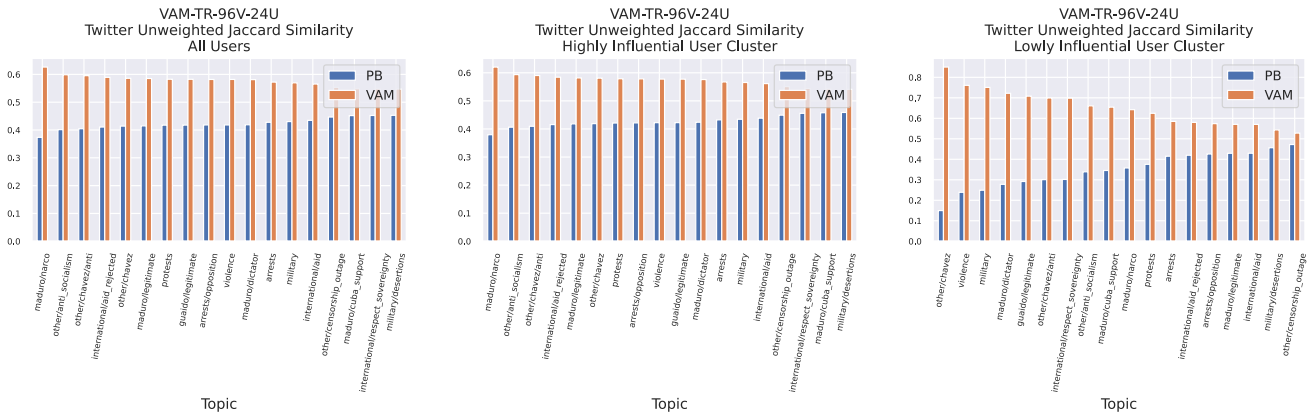
Similar to the Earth Mover's Distance plots, Figure 7 contains the Relative Hausdorff Distance results for the full user set, highly influential users, and lowly influential users. For the full user set and highly influential cluster, VAM won 15 out of 18 times. For the lowly influential cluster, VAM 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.

Tables XXII on page 23, XXIII on page 24, and XXIV on page 24 contain the contain the Relative Hausdorff Distance results for the full set, highly-influential cluster, and lowly-influential cluster of users.



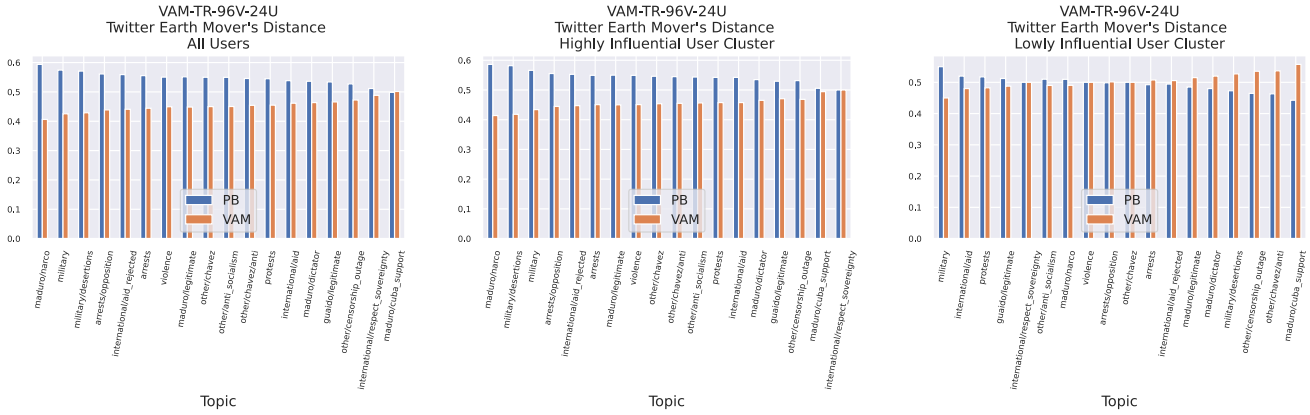
(a) All Users - Weighted JS (Higher is Better) (b) Highly Influential Cluster - Weighted JS (Higher is Better) (c) Lowly Influential Cluster - Weighted JS (Higher is Better)

Figure 4: These barplots show the weighted Jaccard similarity results for the full user set, highly influential cluster, and lowly influential cluster.



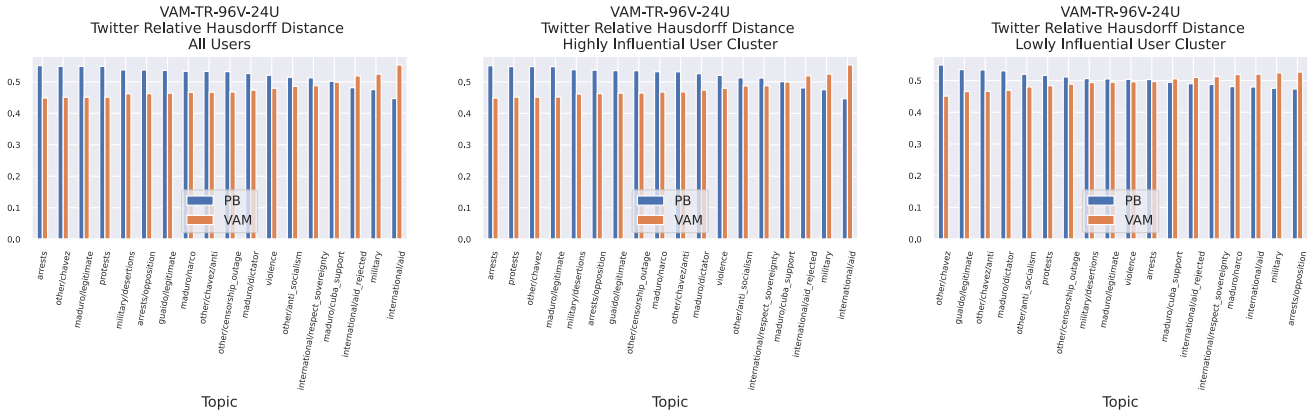
(a) All Users - Unweighted JS (Higher is Better) (b) Highly Influential Cluster - Unweighted JS (Higher is Better) (c) Lowly Influential Cluster - Unweighted JS (Higher is Better)

Figure 5: These barplots show the unweighted Jaccard similarity results for the full user sets, highly influential clusters, and lowly influential clusters.



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

Figure 6: These barplots show the Earth Mover's Distance for the full set of users, highly influential users, and lowly influential users.



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

Figure 7: These barplots show the Relative Hausdorff Distance for the full set of users, highly influential users, and lowly influential users.

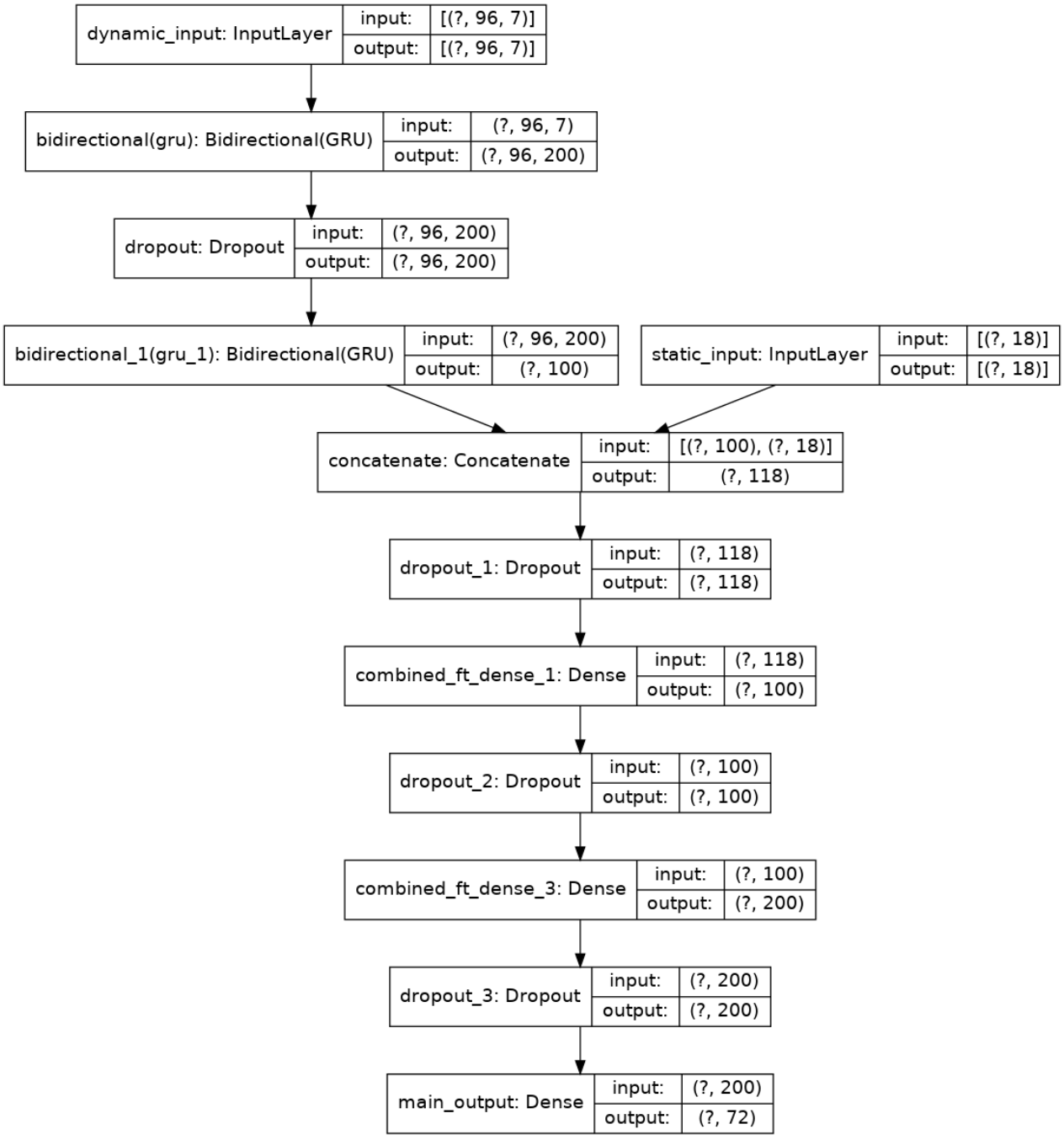


Figure 8: VAM-Bi-GRU-TR-96 Architecture

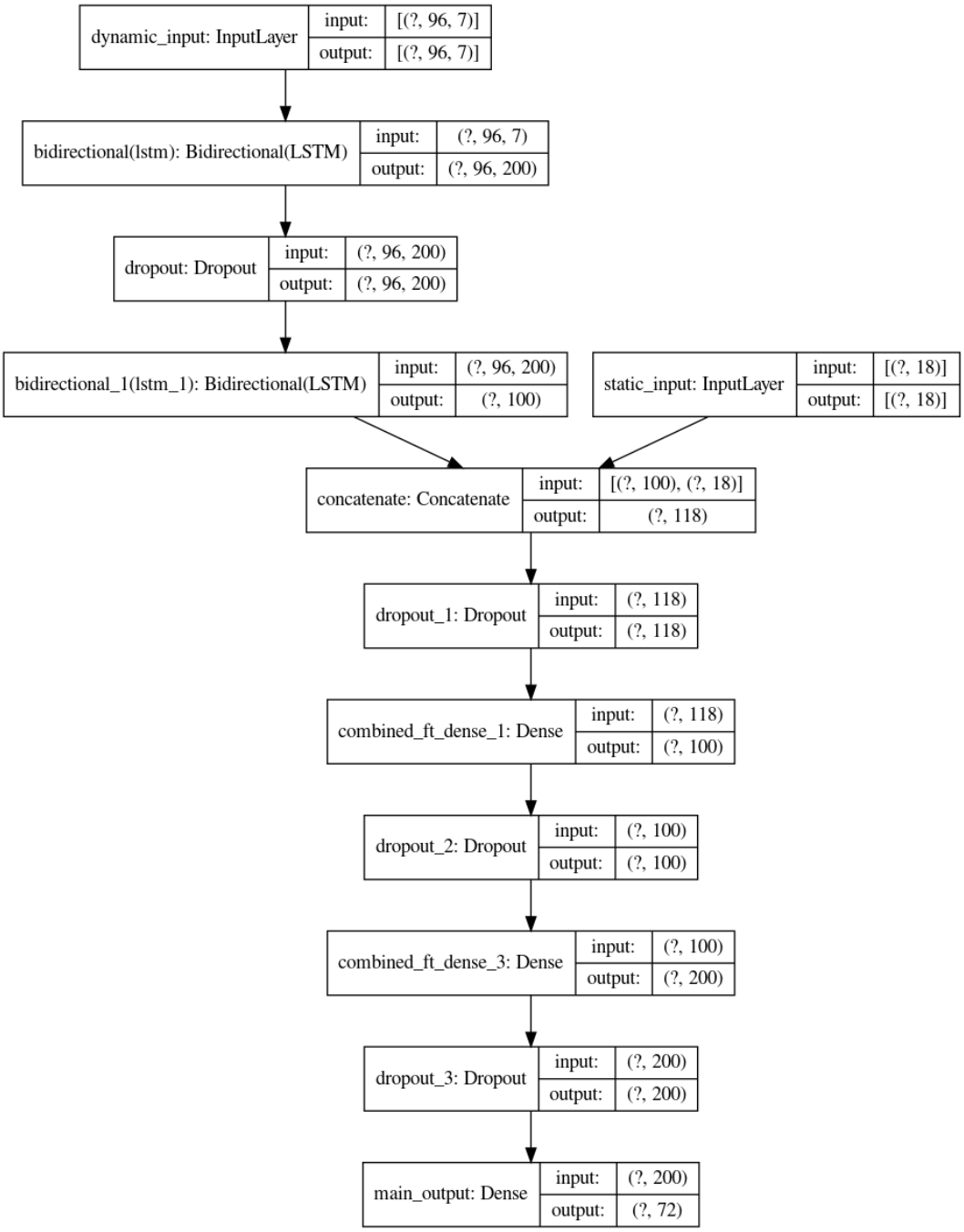


Figure 9: VAM-Bi-LSTM-TR-96 Architecture

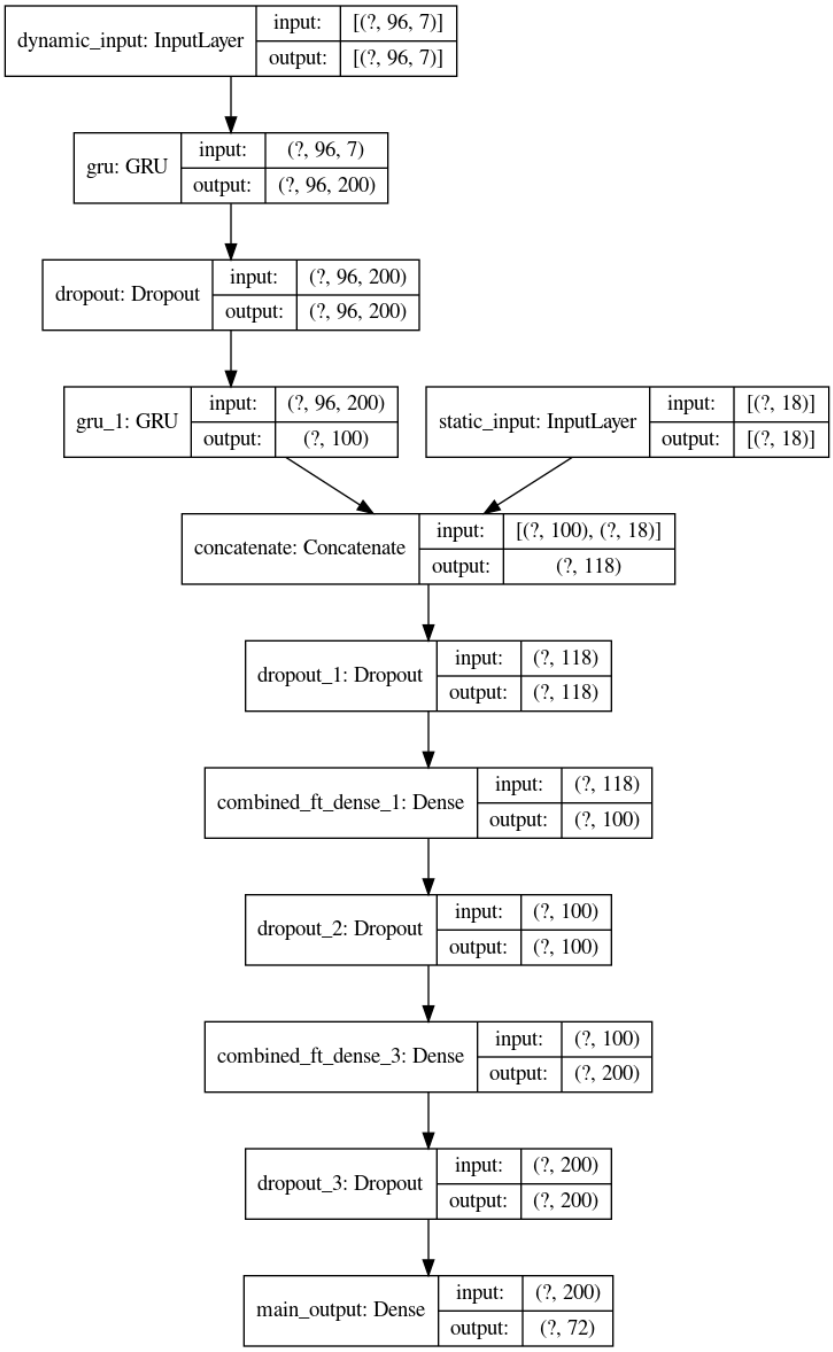


Figure 10: VAM-GRU-TR-96 Architecture

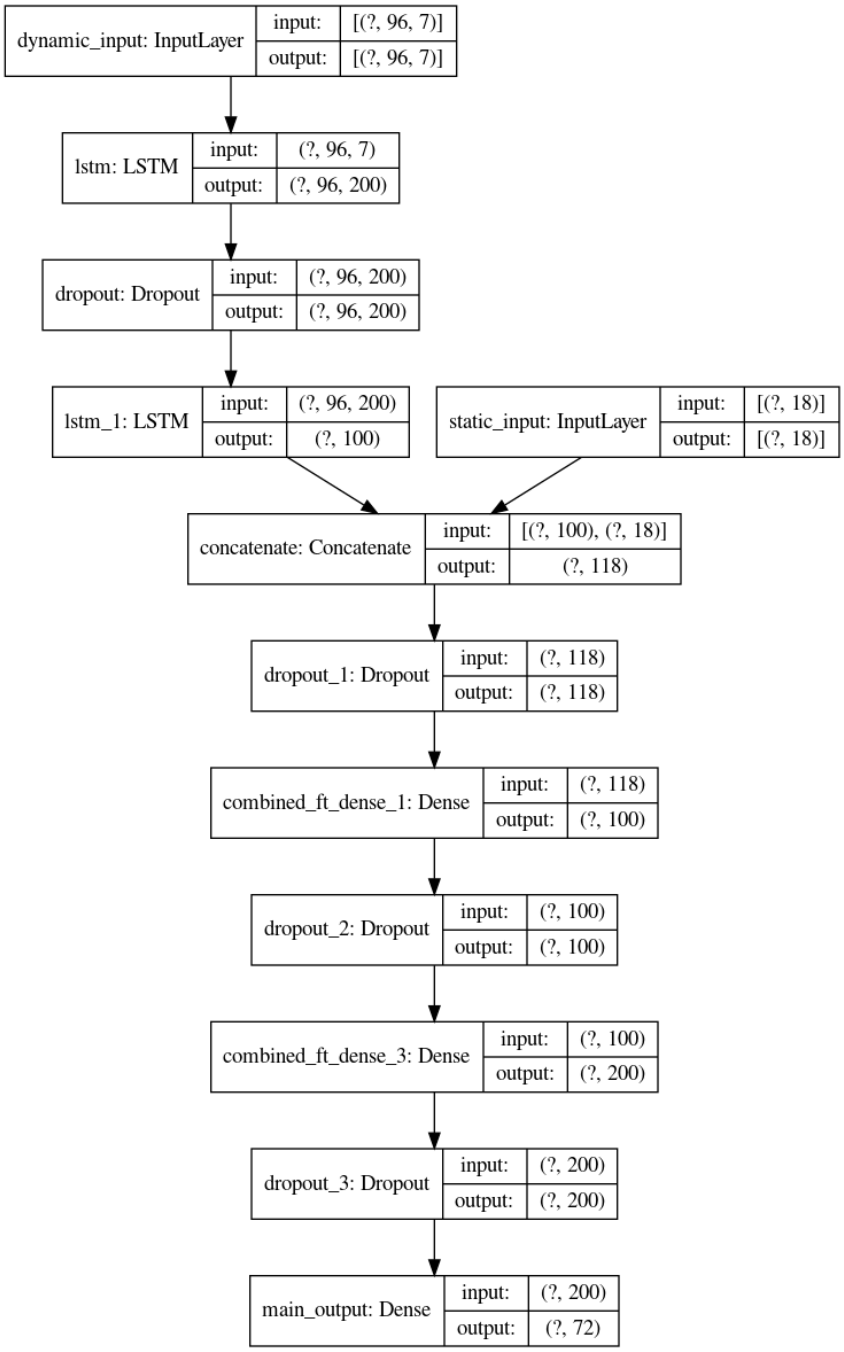
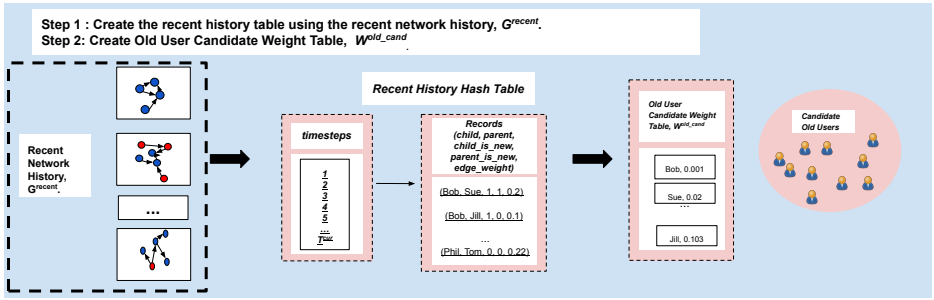
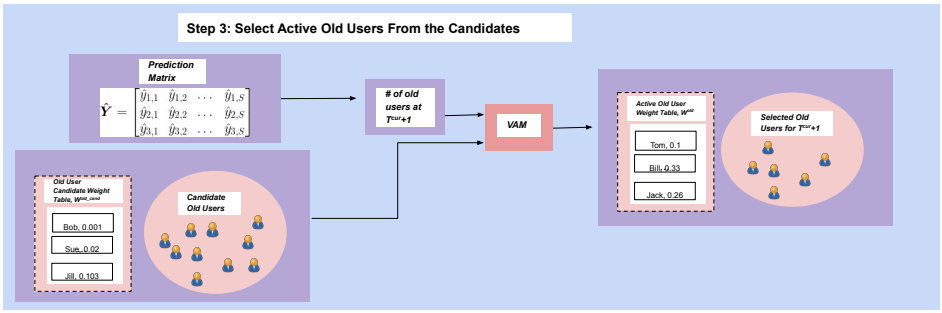


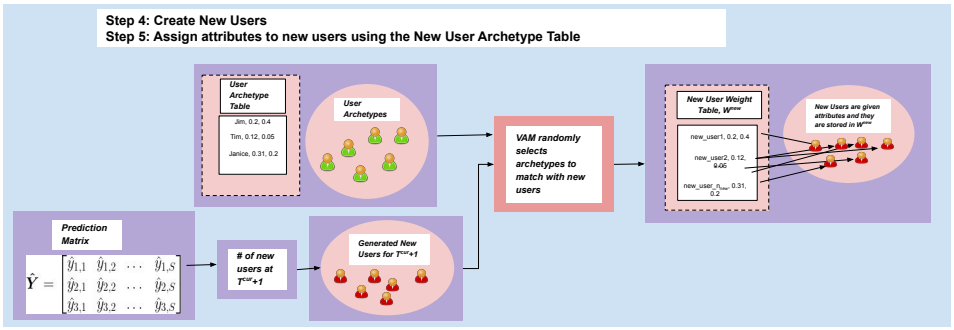
Figure 11: VAM-LSTM-TR-96 Architecture



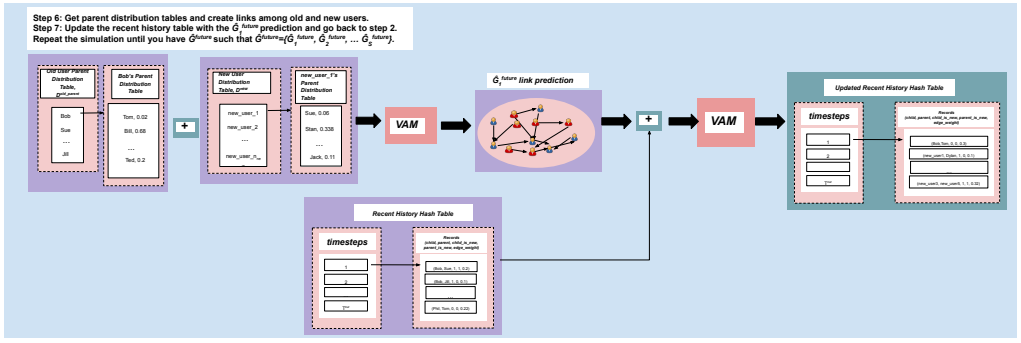
(a) Step 1: Use the recent network history, G^{recent} to create the recent history hash table, H^{recent}. Step 2: Use the recent history table to create the *Old User Candidate Weight Table*, W^{old_cand}.



(b) Step 3: Use the old user candidate weight table and the $\hat{Y}$ volume matrix to select the active old users from the candidates.



(c) Step 4: Use the $\hat{Y}$ volume matrix to generate new users. Step 5: Create the new user archetype table to assign attributes to the generated new users.



(d) Step 6: Create the new and old user parent distribution tables. Use these tables to assign edges among the old and new users using probabilities. This new set of links makes up the graph, $\hat{G}_1^{future}$. Step 7: Use $\hat{G}_1^{future}$ to update the recent history table, H^{recent}. Go back to step 2 and continue the simulation algorithm until *VAM* has created $\hat{G}_1^{future}$ such that $\hat{G}^{future} = \hat{G}_1^{future}, \hat{G}_2^{future}, \ldots \hat{G}_S^{future}$.

Figure 12: Steps 1-7 of the VAM User-Assignment Algorithm.

Twitter MAE VAM and Persistence Baseline Comparisons by Topic			
Topic	**VAM-TR MAE**	**PB MAE**	**PIFB (%)**
military/desertions	**285.53**	547.49	**47.85**
other/censorship_outage	**100.99**	188.03	**46.29**
international/aid_rejected	**549.71**	873.29	**37.05**
protests	**509.04**	804.59	**36.73**
maduro/cuba_support	**85.93**	134.2	**35.97**
arrests	**141.29**	206.29	**31.51**
maduro/narco	**86.52**	124.36	**30.43**
maduro/legitimate	**115.66**	159.71	**27.58**
guaido/legitimate	**553.26**	757.79	**26.99**
violence	**1258.09**	1694.71	**25.76**
arrests/opposition	**102.0**	136.39	**25.22**
other/chavez/anti	**83.63**	110.6	**24.39**
other/chavez	**175.38**	225.52	**22.23**
other/anti_socialism	**46.48**	57.19	**18.73**
military	**1726.54**	2106.43	**18.03**
maduro/dictator	**334.07**	403.28	**17.16**
international/respect_sovereignty	**419.45**	505.66	**17.05**
international/aid	2120.06	**2111.25**	-0.42

Table VI: Twitter MAE VAM and Persistence Baseline Comparisons by Topic

Twitter VAM and Persistence Baseline RMSE Comparisons by Topic			
Topic	**VAM-TR RMSE**	**PB RMSE**	**PIFB (%)**
other/censorship_outage	**231.58**	432.57	**46.46**
military/desertions	**447.63**	796.13	**43.77**
international/aid_rejected	**947.85**	1538.11	**38.38**
protests	**698.22**	1115.51	**37.41**
maduro/cuba_support	**135.34**	207.06	**34.64**
arrests	**216.02**	308.89	**30.07**
maduro/narco	**119.25**	168.2	**29.1**
violence	**1733.91**	2444.25	**29.06**
guaido/legitimate	**879.21**	1198.03	**26.61**
maduro/legitimate	**167.25**	222.42	**24.81**
arrests/opposition	**169.93**	221.65	**23.33**
other/chavez/anti	**115.65**	147.11	**21.39**
military	**2265.17**	2845.83	**20.4**
other/chavez	**238.67**	296.74	**19.57**
other/anti_socialism	**66.41**	80.38	**17.37**
maduro/dictator	**490.78**	573.66	**14.45**
international/respect_sovereignty	**551.22**	642.09	**14.15**
international/aid	**2677.36**	2761.74	**3.06**

Table VIII: Twitter VAM and Persistence Baseline RMSE Comparisons by Topic

Twitter NC-RMSE VAM and Persistence Baseline Comparisons by Topic			
Topic	**VAM-TR NC-RMSE**	**PB NC-RMSE**	**PIFB (%)**
military/desertions	**0.13**	0.27	**50.91**
protests	**0.11**	0.16	**33.69**
other/censorship_outage	**0.17**	0.26	**33.57**
arrests	**0.12**	0.17	**31.44**
maduro/narco	**0.12**	0.17	**29.95**
other/chavez/anti	**0.08**	0.11	**28.35**
guaido/legitimate	**0.13**	0.18	**27.17**
maduro/legitimate	**0.09**	0.13	**26.93**
other/chavez	**0.08**	0.11	**26.55**
other/anti_socialism	**0.11**	0.14	**25.04**
international/aid_rejected	**0.15**	0.2	**24.65**
arrests/opposition	**0.15**	0.2	**23.81**
military	**0.11**	0.14	**22.79**
international/respect_sovereignty	**0.07**	0.09	**20.05**
maduro/cuba_support	**0.12**	0.14	**13.12**
violence	**0.12**	0.14	**10.54**
maduro/dictator	**0.11**	0.12	**8.4**
international/aid	0.11	**0.1**	-7.14

Table VII: Twitter NC-RMSE VAM and Persistence Baseline Comparisons by Topic

Twitter VAM and Persistence Baseline S-APE Comparisons by Topic			
Topic	**VAM-TR S-APE**	**PB S-APE**	**PIFB (%)**
other/chavez	**17.46**	22.48	**22.34**
maduro/narco	**23.16**	29.69	**22.01**
arrests	**25.14**	31.01	**18.93**
arrests/opposition	**30.99**	37.74	**17.89**
protests	**32.2**	38.35	**16.04**
other/chavez/anti	**19.82**	23.29	**14.9**
violence	**31.39**	36.61	**14.27**
maduro/dictator	**19.05**	22.01	**13.44**
other/anti_socialism	**20.83**	23.9	**12.87**
military	**26.8**	30.5	**12.14**
maduro/legitimate	**17.12**	19.3	**11.31**
military/desertions	**36.12**	38.96	**7.31**
guaido/legitimate	**28.32**	29.53	**4.12**
international/aid_rejected	**38.04**	38.29	**0.66**
other/censorship_outage	39.67	**38.99**	-1.73
international/respect_sovereignty	16.74	**16.24**	-3.06
maduro/cuba_support	32.04	**27.53**	-16.39
international/aid	29.58	**25.19**	-17.42

Table IX: Twitter VAM and Persistence Baseline S-APE Comparisons by Topic

Twitter SkE VAM and Persistence Baseline Comparisons by Topic			
Topic	**PB SkE**	**VAM-TR SkE**	**PIFB (%)**
other/chavez	**0.77**	0.61	**20.39**
other/censorship_outage	**1.41**	1.15	**18.43**
violence	**1.21**	1.03	**15.12**
international/respect_sovereignty	**0.8**	0.68	**14.26**
military	**1.06**	0.93	**12.43**
international/aid_rejected	**1.08**	0.96	**10.85**
maduro/cuba_support	**0.84**	0.75	**9.89**
military/desertions	**1.07**	0.97	**9.16**
other/chavez/anti	**0.77**	0.73	**5.46**
maduro/legitimate	**0.73**	0.69	**4.61**
guaido/legitimate	**1.39**	1.36	**2.16**
maduro/dictator	0.91	**0.94**	-2.31
maduro/narco	0.84	**0.91**	-7.97
protests	1.01	**1.17**	-15.81
international/aid	0.77	**0.98**	-27.61
other/anti_socialism	0.9	**1.18**	-30.67
arrests	0.84	**1.22**	-44.85
arrests/opposition	1.01	**1.61**	-58.97

Table X: Twitter SkE VAM and Persistence Baseline Comparisons by Topic

Twitter VAM and Persistence Baseline VE Comparisons by Topic			
Topic	**VAM-TR VE**	**PB VE**	**PIFB (%)**
other/censorship_outage	**196.26**	359.4	**45.39**
international/aid_rejected	**643.88**	1106.68	**41.82**
maduro/cuba_support	**80.97**	133.67	**39.43**
maduro/legitimate	**70.94**	108.6	**34.68**
protests	**370.8**	566.07	**34.5**
maduro/narco	**56.08**	81.79	**31.44**
other/chavez	**83.82**	119.72	**29.99**
military/desertions	**323.45**	453.57	**28.69**
arrests	**101.6**	126.81	**19.88**
violence	**1021.98**	1246.48	**18.01**
military	**1177.11**	1414.2	**16.77**
guaido/legitimate	**469.69**	548.33	**14.34**
other/chavez/anti	**43.92**	50.92	**13.74**
arrests/opposition	**85.57**	96.97	**11.76**
other/anti_socialism	**25.69**	28.51	**9.89**
international/respect_sovereignty	**231.09**	255.26	**9.47**
international/aid	**1228.38**	1244.66	**1.31**
maduro/dictator	**244.29**	245.79	**0.61**

Table XI: Twitter VAM and Persistence Baseline VE Comparisons by Topic

User-Assignment Module Symbols	
Symbol	***Meaning***
G	A temporal graph. It is a series of static graphs.
u	a child user in edge $(u, v, w(u, v, t))$
v	a parent user in edge $(u, v, w(u, v, t))$
$w(u, v, t)$	The number of times child user u interacted with parent v at time t.
T	The prediction time step of interest
S	The length of the output time series we wish to predict
$\hat{\mathbf{Y}}$	The time series matrix consisting of the (1) number of actions, (2) number of new users, and (3) number of old users from time $T+1$ to $T+S$
$\hat{G}^{future}$	The temporal graph sequence that the user-assignment module must predict
L^{user}	The user-assignment lookback factor.
L^{vol}	The volume prediction lookback factor.
s	Any given time step between 1 and S, inclusive.
$T+s$	The current timestep of interest within the scope of the user assignment algorithm. $T+s = T+s-1$.
p^{old_act}	The probability that an old user will be active in a given time step.
W^{old_cand}	The probability weight table for old user candidates.
W^{old}	The probability weight table for selected users from the W^{old_cand} table.
$\hat{O}_s$	The set of predicted active old users in time step $T+s$
$\hat{N}_s$	The set of newly generated active users in time step $T+s$
W^{new_arch}	The new user archetype weight table. Used to select the mostly likely archetypes a newly generated user will behave like.
u^{arch}	A user archetype.
p^{act_arch}	How likely a user of a particular archetype will be active in $T+s$
p^{infl_arch}	How likely a user of a particular archetype will be influential in $T+s$
u^{new}	A newly generated user.

Table XII: A table of the various symbols used in the user-assignment section of this work.

Twitter Unweighted Jaccard Similarity VAM-TR-96V-24U Old Users Full (Unweighted)			
Topic	**PB**	**VAM-TR-96V-24U**	**PIFB**
maduro/narco	0.0914	**0.1532**	**67.6657**
other/anti_socialism	0.0724	**0.108**	**49.1753**
other/chavez/anti	0.088	**0.1295**	**47.1248**
international/aid_rejected	0.1177	**0.1689**	**43.5021**
other/chavez	0.0896	**0.1268**	**41.537**
maduro/legitimate	0.095	**0.134**	**40.9659**
protests	0.1002	**0.1398**	**39.6176**
guaido/legitimate	0.1137	**0.1585**	**39.3256**
arrests/opposition	0.1348	**0.1875**	**39.1254**
violence	0.1265	**0.176**	**39.0475**
maduro/dictator	0.1084	**0.1504**	**38.7333**
arrests	0.1397	**0.1869**	**33.7607**
military	0.1325	**0.1754**	**32.3579**
international/aid	0.1301	**0.1692**	**30.0468**
other/censorship_outage	0.1259	**0.156**	**23.8988**
maduro/cuba_support	0.1026	**0.1245**	**21.2528**
international/respect_sovereignty	0.137	**0.1659**	**21.0456**
military/desertions	0.1554	**0.1878**	**20.8761**

Table XIII: Twitter VAM-TR-96V-24U Old Users - unweighted

Twitter Unweighted JS VAM-TR-96V-24U Old Users Highly Infl.			
Topic	**PB**	**VAM-TR-96V-24U**	**PIFB**
maduro/narco	0.0986	**0.1612**	**63.4799**
other/anti_socialism	0.0777	**0.1135**	**45.9964**
other/chavez/anti	0.094	**0.1355**	**44.1014**
international/aid_rejected	0.1256	**0.1766**	**40.5989**
maduro/legitimate	0.0986	**0.1372**	**39.2171**
other/chavez	0.0962	**0.1334**	**38.6994**
protests	0.1063	**0.1461**	**37.4643**
arrests/opposition	0.1417	**0.1944**	**37.1511**
violence	0.1345	**0.1839**	**36.7106**
guaido/legitimate	0.1225	**0.1674**	**36.6382**
maduro/dictator	0.1177	**0.1599**	**35.8666**
arrests	0.1489	**0.1954**	**31.2357**
military	0.139	**0.1809**	**30.0771**
international/aid	0.1375	**0.1761**	**28.0816**
other/censorship_outage	0.13	**0.1594**	**22.6478**
international/respect_sovereignty	0.1428	**0.1706**	**19.4916**
maduro/cuba_support	0.1116	**0.1322**	**18.4438**
military/desertions	0.1646	**0.1942**	**17.9705**

Table XIV: Twitter VAM-TR-96V-24U Old Users Highly Influential Cluster (Unweighted)

Twitter Unweighted JS VAM-TR-96V-24U Old Users Lowly Infl.			
Topic	**PB**	**VAM-TR-96V-24U**	**PIFB**
other/chavez	0.0043	**0.0244**	**466.3653**
violence	0.0141	**0.0449**	**217.9803**
military	0.0144	**0.0434**	**201.2047**
maduro/dictator	0.0165	**0.0429**	**160.1118**
guaido/legitimate	0.0166	**0.0403**	**142.4018**
other/chavez/anti	0.0575	**0.1337**	**132.4138**
international/respect_sovereignty	0.0162	**0.0375**	**131.8387**
other/anti_socialism	0.1329	**0.2591**	**94.9254**
maduro/cuba_support	0.0915	**0.1734**	**89.5422**
maduro/narco	0.1567	**0.2813**	**79.4937**
protests	0.0478	**0.0795**	**66.4434**
arrests	0.139	**0.1959**	**40.9633**
international/aid_rejected	0.1836	**0.2537**	**38.1899**
arrests/opposition	0.3115	**0.4199**	**34.7953**
maduro/legitimate	0.2144	**0.2849**	**32.9002**
international/aid	0.0229	**0.0304**	**32.6733**
military/desertions	0.5281	**0.6286**	**19.0338**
other/censorship_outage	0.7282	**0.8143**	**11.8256**

Table XV: Twitter VAM-TR-96V-24U Old Users Lowly Influential Cluster (Unweighted)

Twitter Weighted JS VAM-TR-96V-24U Full User Set			
Topic	**PB**	**VAM-TR-96V-24U**	**PIFB**
maduro/cuba_support	0.0465	**0.1023**	**120.1467**
international/aid_rejected	0.0329	**0.0716**	**117.7645**
other/anti_socialism	0.0329	**0.0683**	**107.6227**
maduro/narco	0.0502	**0.1021**	**103.3784**
maduro/legitimate	0.0497	**0.0767**	**54.2822**
other/chavez/anti	0.044	**0.0674**	**53.0084**
violence	0.0582	**0.0815**	**40.0279**
other/censorship_outage	0.0724	**0.1007**	**38.9522**
other/chavez	0.0459	**0.0634**	**38.1265**
maduro/dictator	0.0474	**0.0641**	**35.0402**
military/desertions	0.0595	**0.0798**	**34.1568**
protests	0.0473	**0.0594**	**25.5639**
arrests/opposition	0.0668	**0.0837**	**25.3776**
international/respect_sovereignty	0.0814	**0.0994**	**22.1611**
international/aid	0.0867	**0.1006**	**15.9882**
arrests	0.0742	**0.0809**	**9.0074**
military	0.0802	**0.0814**	**1.5735**
guaido/legitimate	0.0654	**0.0655**	**0.1112**

Table XVI: VAM-TR-96V-24U Old Users Weighted (Full)

Twitter Weighted JS VAM-TR-96V-24U Old Users Highly Infl. Cluster			
Topic	**PB**	**VAM-TR-96V-24U**	**PIFB**
maduro/cuba_support	0.0478	**0.1041**	**117.723**
international/aid_rejected	0.0332	**0.0719**	**116.413**
other/anti_socialism	0.0336	**0.0689**	**105.3473**
maduro/narco	0.0508	**0.1026**	**101.7409**
maduro/legitimate	0.0502	**0.0771**	**53.5985**
other/chavez/anti	0.0448	**0.0679**	**51.5688**
violence	0.059	**0.082**	**39.0396**
other/censorship_outage	0.0734	**0.1011**	**37.8058**
other/chavez	0.0468	**0.0639**	**36.6019**
maduro/dictator	0.048	**0.0645**	**34.232**
military/desertions	0.0618	**0.0801**	**29.5377**
protests	0.048	**0.0598**	**24.7245**
arrests/opposition	0.0675	**0.0842**	**24.603**
international/respect_sovereignty	0.0821	**0.0998**	**21.6542**
international/aid	0.0872	**0.1009**	**15.8175**
arrests	0.0752	**0.0814**	**8.2824**
military	0.081	**0.0818**	**1.0194**
guaido/legitimate	**0.0662**	0.0658	-0.6144

Table XVII: VAM-TR-96V-24U Old Users Weighted - High Cluster

Twitter Weighted JS VAM-TR-96V-24U Old Users Lowly Infl.			
Topic	**PB**	**VAM-TR-96V-24U**	**PIFB**
other/chavez	0.004	**0.0243**	**507.1694**
violence	0.0139	**0.0449**	**222.8647**
military	0.0139	**0.0433**	**211.1303**
maduro/dictator	0.0159	**0.0429**	**169.3563**
guaido/legitimate	0.016	**0.0402**	**150.9818**
international/respect_sovereignty	0.0159	**0.0374**	**135.193**
other/chavez/anti	0.0575	**0.1337**	**132.4138**
other/anti_socialism	0.1329	**0.2591**	**94.9254**
maduro/cuba_support	0.0913	**0.1734**	**89.9963**
maduro/narco	0.1567	**0.2813**	**79.4937**
protests	0.0478	**0.0795**	**66.3686**
arrests	0.1389	**0.1957**	**40.8761**
international/aid_rejected	0.1829	**0.2537**	**38.7267**
international/aid	0.022	**0.0302**	**36.8813**
arrests/opposition	0.3115	**0.4198**	**34.7777**
maduro/legitimate	0.2143	**0.2848**	**32.914**
military/desertions	0.5278	**0.6286**	**19.0961**
other/censorship_outage	0.7282	**0.8143**	**11.8256**

Table XVIII: Twitter VAM-TR-96V-24U Old Users Lowly Influential Cluster (Weighted)

Twitter VAM-TR-96V-24U Earth Mover's Distance Full User Set Results			
Topic	**PB**	**VAM-TR-96V-24U**	**PIFB (%)**
maduro/narco	0.0497	**0.034**	**31.58**
military	0.0027	**0.002**	**25.94**
military/desertions	0.1817	**0.1365**	**24.85**
arrests/opposition	0.0609	**0.0476**	**21.8**
international/aid_rejected	0.035	**0.0276**	**21.1**
arrests	0.023	**0.0184**	**19.97**
violence	0.006	**0.0049**	**18.77**
maduro/legitimate	0.0198	**0.0161**	**18.65**
other/chavez	0.0066	**0.0054**	**18.18**
other/anti_socialism	0.0365	**0.0299**	**18.09**
other/chavez/anti	0.0173	**0.0144**	**17.16**
protests	0.0121	**0.0101**	**17.08**
international/aid	0.0028	**0.0024**	**15.98**
maduro/dictator	0.0059	**0.0051**	**12.8**
guaido/legitimate	0.0047	**0.0041**	**12.21**
other/censorship_outage	0.269	**0.241**	**10.43**
international/respect_sovereignty	0.0022	**0.0021**	**2.94**
maduro/cuba_support	**0.048**	0.0483	-0.59

Table XIX: VAM Earth Mover's Distance Full User Set Results

Twitter VAM-TR-96V-24U Earth Mover's Distance Lowly Influential Cluster			
Topic	**PB**	**VAM-TR-96V-24U**	**PIFB (%)**
military	0.0011	**0.0009**	**15.67**
international/aid	0.0013	**0.0012**	**9.76**
protests	0.0074	**0.0069**	**5.58**
guaido/legitimate	0.0021	**0.002**	**5.48**
international/respect_sovereignty	0.0014	**0.0014**	**4.46**
other/anti_socialism	0.0307	**0.0295**	**4.08**
maduro/narco	0.024	**0.0231**	**3.84**
violence	0.0028	**0.0028**	**2.62**
arrests/opposition	**0.0178**	0.0179	-0.5
other/chavez	**0.0046**	0.0046	-1.42
arrests	**0.0097**	0.01	-2.39
international/aid_rejected	**0.0178**	0.0182	-2.44
maduro/legitimate	**0.0161**	0.0171	-6.44
maduro/dictator	**0.0024**	0.0026	-8.02
military/desertions	**0.0394**	0.0439	-11.63
other/censorship_outage	**0.0616**	0.071	-15.25
other/chavez/anti	**0.0107**	0.0124	-15.31
maduro/cuba_support	**0.0197**	0.0248	-25.68

Table XXI: Twitter VAM-TR-96V-24U Earth Mover's Distance Lowly Influential Cluster

Twitter VAM-TR-96V-24U Earth Mover's Distance Highly Influential Cluster			
Topic	**PB**	**VAM**	**PIFB (%)**
maduro/narco	0.0511	**0.0361**	**29.29**
military/desertions	0.1842	**0.1324**	**28.08**
military	0.003	**0.0023**	**23.73**
arrests/opposition	0.0617	**0.0494**	**19.94**
international/aid_rejected	0.0357	**0.0289**	**18.93**
arrests	0.0245	**0.0201**	**18.22**
maduro/legitimate	0.0204	**0.0167**	**18.07**
violence	0.0067	**0.0055**	**17.84**
other/chavez	0.0071	**0.0059**	**17.13**
other/chavez/anti	0.0182	**0.0152**	**16.48**
other/anti_socialism	0.0362	**0.0304**	**15.96**
protests	0.0128	**0.0108**	**15.48**
international/aid	0.0032	**0.0027**	**15.1**
maduro/dictator	0.0069	**0.006**	**12.29**
guaido/legitimate	0.0054	**0.0048**	**12.01**
other/censorship_outage	0.2687	**0.2368**	**11.87**
maduro/cuba_support	0.0494	**0.0483**	**2.18**
international/respect_sovereignty	0.0023	**0.0023**	**1.99**

Table XX: Twitter VAM-TR-96V-24U Earth Mover's Distance Highly Influential Cluster

Twitter VAM-TR-96V-24U Relative Hausdorff Distance Full User Set Results			
Topic	**PB**	**VAM**	**PIFB (%)**
arrests	1.2153	**0.9883**	**18.68**
other/chavez	0.9996	**0.8205**	**17.92**
maduro/legitimate	0.9915	**0.8148**	**17.82**
protests	1.3615	**1.1192**	**17.8**
military/desertions	1.4612	**1.255**	**14.11**
arrests/opposition	1.1051	**0.9508**	**13.96**
guaido/legitimate	1.1751	**1.017**	**13.45**
maduro/narco	1.0597	**0.9269**	**12.53**
other/chavez/anti	0.9505	**0.8329**	**12.37**
other/censorship_outage	0.9662	**0.8483**	**12.19**
maduro/dictator	1.0244	**0.9213**	**10.06**
violence	1.3646	**1.2564**	**7.93**
other/anti_socialism	0.8227	**0.7764**	**5.63**
international/respect_sovereignty	0.9451	**0.8986**	**4.92**
maduro/cuba_support	0.8429	**0.8376**	**0.63**
international/aid_rejected	**1.1422**	1.2309	-7.77
military	**1.0861**	1.1989	-10.38
international/aid	**1.1475**	1.4215	-23.87

Table XXII: Twitter VAM-TR-96V-24U Relative Hausdorff Distance Full User Set Results

Twitter VAM-TR-96V-24U Relative Hausdorff Distance Highly Influential Cluster			
Topic	**PB**	**VAM-TR-96V-24U**	**PIFB (%)**
arrests	1.2161	**0.9897**	**18.62**
protests	1.3625	**1.121**	**17.72**
other/chavez	1.0004	**0.8238**	**17.65**
maduro/legitimate	0.9922	**0.8172**	**17.63**
military/desertions	1.4662	**1.2556**	**14.36**
arrests/opposition	1.1065	**0.9536**	**13.82**
guaido/legitimate	1.1751	**1.0183**	**13.34**
other/censorship_outage	0.9713	**0.8424**	**13.27**
maduro/narco	1.0614	**0.9326**	**12.13**
other/chavez/anti	0.9499	**0.8358**	**12.01**
maduro/dictator	1.0246	**0.9233**	**9.89**
violence	1.3645	**1.257**	**7.88**
other/anti_socialism	0.8252	**0.7831**	**5.11**
international/respect_sovereignty	0.9453	**0.8998**	**4.81**
maduro/cuba_support	0.8411	**0.8389**	**0.27**
international/aid_rejected	**1.143**	1.2347	-8.02
military	**1.0863**	1.2	-10.47
international/aid	**1.1475**	1.4227	-23.98

Table XXIII: Twitter VAM-TR-96V-24U Relative Hausdorff Distance Highly Influential Cluster

Twitter VAM-TR-96V-24U Relative Hausdorff Distance Lowly Influential Cluster			
Topic	**PB**	**VAM-TR-96V-24U**	**PIFB (%)**
other/chavez	0.5363	**0.4406**	**17.85**
guaido/legitimate	0.5398	**0.47**	**12.93**
other/chavez/anti	0.453	**0.3957**	**12.63**
maduro/dictator	0.4676	**0.4135**	**11.57**
other/anti_socialism	0.3573	**0.3299**	**7.67**
protests	0.4845	**0.454**	**6.31**
other/censorship_outage	0.1012	**0.0967**	**4.37**
military/desertions	0.1978	**0.1929**	**2.44**
maduro/legitimate	0.3073	**0.301**	**2.04**
violence	0.4814	**0.4742**	**1.51**
arrests	0.3678	**0.3632**	**1.23**
maduro/cuba_support	**0.3701**	0.3784	-2.24
international/aid_rejected	**0.3535**	0.3678	-4.06
international/respect_sovereignty	**0.4867**	0.5111	-5.0
maduro/narco	**0.2137**	0.2306	-7.88
international/aid	**0.5191**	0.5628	-8.42
military	**0.4737**	0.5224	-10.29
arrests/opposition	**0.2264**	0.2519	-11.25

Table XXIV: Twitter VAM-TR-96V-24U Relative Hausdorff Distance Lowly Influential Cluster

REFERENCES

[1] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[2] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[3] 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.