Learning from Dynamic User Interaction Graphs to Forecast Diverse Social Behavior

Prasha Shrestha
Pacific Northwest National Laboratory
Richland, WA
prasha.shrestha@pnnl.gov

Suraj Maharjan
Pacific Northwest National Laboratory
Richland, WA
suraj.maharjan@pnnl.gov

Dustin Arendt
Pacific Northwest National Laboratory
Richland, WA
dustin.arendt@pnnl.gov

Svitlana Volkova
Pacific Northwest National Laboratory
Richland, WA
svitlana.volkova@pnnl.gov

ABSTRACT

Most of the existing graph analytics for understanding social behavior focuses on learning from static rather than dynamic graphs using hand-crafted network features or recently emerged graph embeddings learned independently from a downstream predictive task, and solving predictive (e.g., link prediction) rather than forecasting tasks directly. To address these limitations, we propose (1) a novel task – forecasting user interactions over dynamic social graphs, and (2) a novel deep learning, multi-task, node-aware attention model that focuses on forecasting social interactions, going beyond recently emerged approaches for learning dynamic graph embeddings. Our model relies on graph convolutions and recurrent layers to forecast future social behavior and interaction patterns in dynamic social graphs. We evaluate our model on the ability to forecast the number of retweets and mentions of a specific news source on Twitter (focusing on deceptive and credible news sources) with R^2 of 0.79 for retweets and 0.81 for mentions. An additional evaluation includes model forecasts of user-repository interactions on GitHub and comments to a specific video on YouTube with a mean absolute error close to 2% and R^2 exceeding 0.69. Our results demonstrate that learning from connectivity information over time in combination with node embeddings yields better forecasting results than when we incorporate the state-of-the-art graph embeddings e.g., Node2Vec and DeepWalk into our model. Finally, we perform in-depth analyses to examine factors that influence model performance across tasks and different graph types e.g., the influence of training and forecasting windows as well as graph topological properties.

CCS CONCEPTS

• **Networks → Online social networks**; • **Human-centered computing → Social network analysis**; • **Computing methodologies → Neural networks**.

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11
https://doi.org/10.1145/3357384.3358043

KEYWORDS

dynamic graphs, node-aware attention, social activity forecasting

ACM Reference Format:
Prasha Shrestha, Suraj Maharjan, Dustin Arendt, and Svitlana Volkova. 2019. Learning from Dynamic User Interaction Graphs to Forecast Diverse Social Behavior. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3357384.3358043

Neural network models have been successfully used to encode structural properties of graphs by learning representations over static graphs [14]. These low-dimensional embeddings have been useful as feature inputs in a variety of downstream prediction tasks such as node classification, clustering, and link prediction [12, 13, 23, 24]. However, learning graph embeddings and then applying them in downstream prediction tasks is a two-step process, and task-specific supervision has to be incorporated [13, 17]. Moreover, the fact that node embeddings are learned independently from the downstream tasks limits their predictive utility.

Another problem with limiting methods to work on static graphs only is that most real-world graphs are inherently dynamic in nature. Online social networks, for example, gradually change and evolve over time. In order for a model to simulate or forecast future behavior, it needs to take this dynamic nature of graphs into account. It has to incorporate information not only from the present but also from past time periods. Failing to include valuable past information will result in an inaccurate model. In this paper, we look at real-world dynamic social graphs and model several types of user interactions. We incorporate methods that have been shown to have worked for static graphs into our novel method for temporal dynamic graphs.

In this work, we propose a novel task – forecasting over dynamic graphs, and a novel node-aware attention model to forecast interactions in real-world social graphs. Our approach is inspired by recent advances in deep learning approaches for large scale graph analytics, namely graph convolutional networks [17], graph attention networks [26], graph RNNs [29], and recently emerged dynamic graph embeddings [21, 25, 30, 33]. Our node attention model focuses on forecasting propagation behavior for each node separately without having to build separate models for each. In contrast to previous work, our model:

- is capable of learning from *dynamic graphs* instead of static graphs [14];
- takes a sequence of *adjacency matrices* as input and learns *dynamic graph embeddings optimized towards specific forecasting tasks* instead of relying on hand-crafted network features or learning dynamic embeddings independently from the prediction problem [21, 33][1];
- *is generalizable* across social platforms with different types of communication behavior and interaction patterns;
- has *forecasting capabilities* instead of predictive capabilities (going beyond link prediction task);
- unlike any other work, allows *forecasting multiple types of interactions simultaneously* e.g., retweet and mention using a multi-task learning setup.

Furthermore, unlike most previous work we not only evaluate model generalizability across multiple graphs but also report novel insights on how model performance depends on the structural topology of social graphs, and the size of the training and forecasting windows.

1 NODE-AWARE ATTENTION MODEL

We present the overall architecture of the proposed multi-task node-aware attention model in Figure 1. The model takes daily social interaction graphs represented as adjacency matrices, day of the week information, and the distribution over the activity from the previous n days as inputs and forecasts the activity for each node in the network for the next day. Before calculating daily distribution vectors, we perform L1 normalization to a range between 0 and 1 for dependent variables for each day using min-max normalization.

[1]Unfortunately, a direct comparison with models for learning dynamic graph embeddings is not possible due to a specific way we formulate our forecasting task (going beyond link prediction).

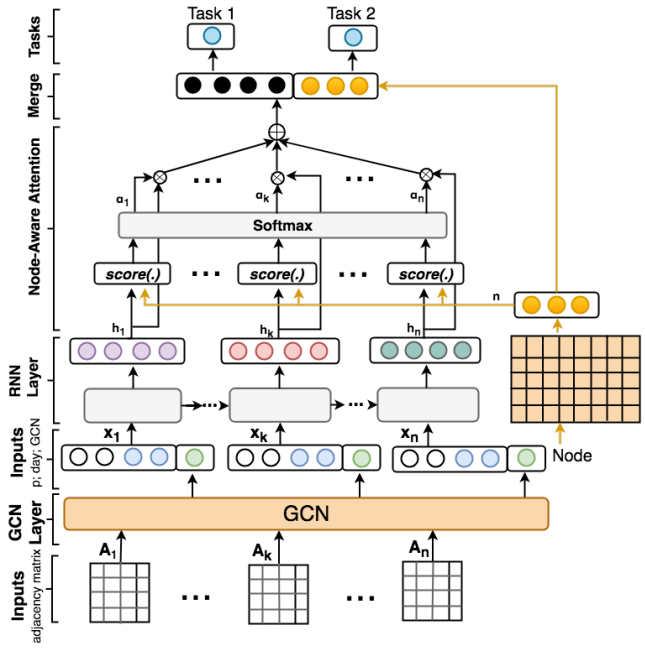


Figure 1: Node-aware attention model (NAAM). $A_1, A_2, \ldots, A_n$ represent the input sequence of past n days adjacency matrices.

Our model first applies a graph convolution neural network layer (GCN) [17] on daily adjacency matrices for social interaction graphs to obtain *daily summary vectors*. We use a featureless version of GCN. Learning graph convolutions in the spatial domain is challenging because a concept of sequence or ordering of the nodes in the network does not exist. However, convolutions in the spatial domain can be represented by a multiplication operation in the spectral domain. Graph CNNs [6] are one of the earlier works to explore this idea which improves upon the earlier work [2] by offering strictly localized filters, which reduces the complexity from quadratic to linear.

We also learn vectors for each day of the week, referred to as *day of the week vectors* since user activity can be different based on the day of the week. For example, people might have higher social media activity on the weekends. A gradual change in activity is more likely than a drastic change in social media platforms. The activity of the future is likely to be close to the activity in the recent past. For this reason, we also include *historical distribution vectors* of activity in our model. Our model then combines the concatenation of the flattened output vectors obtained by applying the GCN on each day's adjacency matrix (daily summary vectors), with the corresponding day of the week's vector, and the historical distribution vectors. The sequence of combined vectors $(\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n})$ are fed through a Recurrent Neural Network (RNN) layer, which is either an LSTM or a bi-directional LSTM (BiLSTM) to obtain representations $\mathbf{h}_i$ for each timestep i . We learn the relative importance of these $\mathbf{h}_i$ to the task at hand by applying node-aware attention. Node-aware attention takes the appropriate node vector $\mathbf{v}$ and $\mathbf{h}_i$ and computes the score for each of the inputs as follows:

$$score(\mathbf{h_i}, \mathbf{v}) = \mathbf{v}^T selu(\mathbf{W_a h_i} + \mathbf{W_n v} + \mathbf{b_a}) \tag{1}$$

where, $\mathbf{W_a}$ and $\mathbf{W_n}$ are the weight matrices, $\mathbf{v}$ is the weight vector, and *selu* [18] is the nonlinear activation function. The node vectors are initialized as orthogonal vectors at first. However, we allow these vectors to be learned along with the rest of the model during training. This will let the node vectors to learn embeddings that capture the similarities in terms of the social activity for them. We take the scores for each node in our network and then normalize them using softmax.

$$\alpha_i = \frac{\exp(score(\mathbf{h_i}, \mathbf{v}))}{\sum_{i'} \exp(score(\mathbf{h_{i'}}, \mathbf{v}))} \tag{2}$$

Here, α_i are the weights measuring the importance of each of the previous n day's daily summary vectors. We take a weighted average of $\mathbf{h}_i$ using α_i to get the final feature representations $(\mathbf{r})$ from the previous n days.

$$\mathbf{r} = \sum_{i}^{n} \alpha_i \mathbf{h_i} \tag{3}$$

We then concatenate the final feature representation vector $\mathbf{r}$ with the node vector $\mathbf{v}$, and feed the combined representation through a non-linear layer to get the predicted variable. Since we are trying to predict the activity for each node separately, the use of

node-aware attention and node representation learning, in general, allows us to forecast future activity for each node separately without having to build separate models for each of them. We use mean squared error (MSE) as our loss function and train the network using the Adam optimizer [16]. Since for various social media, there can be many types of activities eg. retweet and mentions on Twitter, our model also works in a multi-task setting. For our multi-task setting (MT), the total loss is the addition of all individual task loses, i.e., $\mathcal{L}_{total} = \mathcal{L}_{task1} + \mathcal{L}_{task2}$, where for instance $task1$ is retweet counts and $task2$ is the number of mentions for Twitter. The additional tasks act as regularizers and help to learn a generalized model [5, 10].

2 DYNAMIC SOCIAL GRAPH

Because we are interested in forecasting specific type of behaviors in different social platforms related to information propagation, the existing datasets frequently used to learn dynamic embeddings from social networks are not directly applicable, and are often small (between 20K and 300K edges) [21]. We took advantage of a large publicly available Twitter dataset collected by Volkova et al. [27] that consists of 17,186,702 user posts that are either retweets and mentions of verified and deceptive news accounts e.g. @NYTimes, @infowars previously used for deception classification tasks. We looked at a time interval between Feb 1, 2016 to Jan 8, 2017.

Twitter interaction graph can be represented as a dynamic bipartite network $G = (U, N, E)$ that encodes user-news sources' interactions over time. Users U and news source accounts N are nodes, while edges E represent the relationships (retweets and mentions) between users and news source accounts with each edge being attributed with a type of interaction and a timestamp. We wanted to forecast the propagation patterns of these tweets and for that, we treated the retweet and mention behaviors separately. We will model them as closely-related, supportive tasks using our multi-task setting.

When looking at the actual retweet and mention counts, we found that for many news sources there was very sparse retweet or mention activity for most days in the timeframe. Unlike most of earlier work [33], we design our subsampling strategy from dynamic social networks in such a way to not only focus on interesting interaction behavior between users and repositories, user and video and users and news sources, but also cover a time period of one year and more. We selected news sources with either retweet or mention activity in at least 25% of the days in that timeframe, which resulted in a set of 202 news accounts and 1,779,138 users who retweeted or mentioned them. We decided on this number in order to be able to work with news sources having at least some signal for the model to learn from, while also being left with a substantial number of news sources to train on. For mentions, we count all tweets that @mention a particular news source per day. If some tweets mention more than one news source, then these tweets will be included in the counts for both news sources.

2.1 Bipartite Graph Projection

We need to create an adjacency matrix from each day's graph in order to feed it to our model. The first option would be to create an adjacency matrix directly from a bipartite graph. This will produce a highly sparse matrix with at least half of the values being zeros, which can be problematic [4]. The second option is to perform a one-mode projection of the bipartite graph. For the Twitter graphs, since we are forecasting retweet and mention counts for news sources, we need our graphs to be projected onto them. The news sources are our top nodes and the users are our bottom nodes. A strongly connected and a weakly connected one can produce similar projected graphs if the frequency of common associations with the top nodes is not taken into consideration [34]. Avoiding this issue is even more important for our case since we are building daily graphs and these graphs may not change much from one day to another. For this reason, we also weighted the edges of each projected graph by the frequency of common associations of the two bottom nodes with the top node.

We also noticed that the original graph is not exactly bipartite. Some news sources can retweet other news sources. In a few cases, news sources can also mention each other. Although this only represented 0.031% of the total activity in the Twitter data, these direct associations show a closer relationship between the two news sources and they need to be given more importance. We first removed these edges to create a purely bipartite graph for Twitter and then obtained a weighted, projected graph by increasing the weight of edges between news sources with these connections. Since direct links represent stronger connections, each direct connection is given double the weight given to associations made through top nodes.

3 EXPERIMENTAL SETUP

3.1 Forecasting Experiments

In order to evaluate the proposed node-aware attention model, we run a set of forecasting experiments. We pass daily adjacency matrices from the past week as input to our model, which forecasts activity for the coming day. Specifically, we forecast daily distributions of retweets and mentions for each news source on Twitter (output daily vector size = 202). One advantage of our model being node-aware is that we do not need to train an individual model per news source to forecast its activity. We can train a single model that incorporates the whole network and the node attention, as well as the node embedding, tailors the representations to the news source, which we can then use to predict retweets and mention counts for that particular news source.

We experiment with two different setups: a single task (ST) and a multitask (MT) setup. In the single task setup, we built separate models for retweet and mention forecasting tasks. Whereas in the multitask setup, we built a single model to forecast both. Retweet and mention forecasting tasks are two independent but complementary tasks. If a news source is popular at a certain time, it will likely receive a high number of mentions as well as retweets. For complementary tasks like retweet and mention forecasting, a multitask setup can boost the performance on both tasks, and one task can act as a regularizer for another task to avoid overfitting.

We divided our data into training, validation, and test subsets by using the following partitions:

- Train period: February 1, 2016 to August 31, 2016 (213 days)
- Validation period: September 1, 2016 to October 31, 2016 (61 days)

- Test period: November 1, 2016 to January 8, 2017 (69 days)

Since we activity information for 202 news sources per day and a training window of 7 days, we actually have 41,612, 10,908, and 12,524 examples in train, validation, and test respectively.

3.2 Parameter Tuning

Using our validation set, we tuned the following parameters: dropout rates {0.2, 0.4, 0.5}, weight initialization schemes {Glorot Uniform [9], LeCun Uniform [19]}, Adam optimizer learning rate $\{10^{-4}, \ldots, 10^{-1}\}$, the number of hidden neurons in different layers {100, 200}, attention units {1, 100}, graph convolution hidden units {8, 16, 32}, and batch sizes {4, 32, 64} with an early stopping criteria that had a maximum epoch count of 200. We initialized daily vectors and node embeddings with orthogonal vectors of sizes 7 and 32, respectively. This initialization ensured that both day and node vectors do not have any similarities at the beginning of training. However, since we also learn these embeddings during training, if there are similarities among them, the iterative updates of the embedding vectors will capture them.

3.3 Baselines

We define the following three different baseline methods to evaluate against our proposed model.

3.3.1 Previous Day. This baseline ignores all past n-day features and only uses the past 1 day values to predict the next day's values of retweets and mentions on Twitter. Although simple, this is a very strong baseline as the change in the amount of activity received by a news source from day to day is small.

3.3.2 Autoregressive integrated moving average (ARIMA). The ARIMA model is a commonly used architecture when forecasting the price or return of stocks [1, 22, 32]. This model also considers the past history when making future predictions. We tuned the parameters of this model as well, using our validation set. We ran a grid search over the p {0-4}, d {0-2}, and q {0-2} parameters.

3.3.3 Node Embeddings. Our model learns node embeddings along with the main task together. We wanted to compare our model against a two-step process of learning the node embeddings first. We trained Node2Vec [12] and DeepWalk [23] embedding vectors for the nodes in our graph. We trained new representations for each day. In order to ensure a fair comparison, we sequentially use the node embeddings from the previous day to initialize the current day's embeddings, and then fine tune these embeddings. After obtaining the embeddings, We use a sequence of node embeddings from the previous $n = 7$ days as the input to an LSTM and use the final output from the LSTM layer for forecasting.

3.4 Evaluation Metrics

To evaluate the performance of our model, we report six metrics in the results tables. Specifically, we measure Pearson correlation, mean squared error (MSE), root mean squared error (RMSE), maximum absolute error (MAE), median absolute error (MEAE), and the coefficient of determination R^2. These evaluation metrics were calculated over the testing time period for each news source and then averaged.

4 FORECASTING RETWEETS AND MENTIONS ON TWITTER

We report experimental results for forecasting retweets and mentions of our 202 news sources in Table 1 for both single task and multitask settings. We can observe that the previous day baseline has a high R^2 value of 0.75 for the forecasting of mention distributions and 0.72 for the forecasting of retweet distributions for the next day. We can also see that the ARIMA model comes close to this baseline for both retweet and mention forecasting. However, our models outperform all baselines. Among our defined baselines, node embeddings with LSTM method perform the worst, showing that a two-step process of first learning node embeddings and then applying them to the forecasting task works worse than a one-step learning process. Instead, our model learns node embeddings along with the actual task. For retweet forecasting, we obtain the best R^2 of 0.79 from the NAAM with the BiLSTM layer in the single task setting. For mention forecasting, the same model yields the highest R^2 of 0.81 in the single-task setting. The models with the BiLSTM layers also have the lowest MSE, MAE, and MEAE values and the highest Pearson. Similar to the previous day baseline, it is more difficult to forecast retweets than mentions, although the performance is not too different.

Visualizing actual versus predicted mentions for some example news sources on Twitter. In order to examine our results for individual news sources, we visualized the actual and predicted forecasting results. Figure 2 shows some of these visualizations for mentions of example verified and unverified news sources on Twitter. We can see that for most news sources where there is significant activity over time, our model is able to forecast mentions successfully. The forecasted trend is close to the actual trend for most news sources. For some news sources like *de sputnik* and *vesti news*, our model is also able to nearly match the actual mention values. The performance of our model for *USA Today* is worse than that for other news sources, most likely because the signal for this news source is low throughout the time period when compared with other news sources.

5 EVALUATING MODEL GENERALIZABILITY AND ROBUSTNESS

Our model was able to perform better than the baseline methods for the Twitter graph. We wanted to test generalizability of our model on other social graphs. We collected data from two more social environments where the mode and rate of interactions are different from Twitter, namely Youtube and GitHub (GitHub viewed as a collaborative social platform [3, 20]). We present network structure visualizations of Twitter, YouTube and GitHub graphs in Figure 5. From the figure, we can observe how interaction graphs for different social media platforms differ from each other. We can also see the dynamic nature of the graphs and how the graph for one day looks very different from the graph for another day.

5.1 Forecasting Video Comments on YouTube

For YouTube, we worked with a set of 80,155 videos and their corresponding comments posted by 2,363,861 users between November 2013 and February 2017 from a publicly available Youtube

Model	Task	Forecast	MSE $\times 10^{-5}$	MAE $\times 10^{-3}$	MEAE $\times 10^{-4}$	RMSE $\times 10^{-3}$	Pearson	R^2
Previous day baseline	–	mention	4.98	1.73	2.70	7.06	0.87	0.75
	–	retweet	6.56	1.93	2.24	8.10	0.86	0.72
ARIMA model	–	mention	5.76	1.63	**2.63**	7.59	0.86	0.74
	–	retweet	8.37	1.91	2.62	9.15	0.83	0.69
Node2Vec + BiLSTM	–	mention	9.23	3.91	19.5	9.61	0.77	0.53
	–	retweet	6.63	2.43	5.50	8.14	0.85	0.72
DeepWalk + BiLSTM	–	mention	7.58	3.47	16.9	8.71	0.79	0.61
	–	retweet	6.72	2.78	8.89	8.20	0.85	0.71
NAAM + 2 LSTM layers	ST	mention	4.38	1.75	3.04	6.62	0.88	0.78
	ST	retweet	5.79	**1.92**	**2.17**	7.61	0.87	0.75
	MT	mention	5.35	2.06	3.64	7.32	0.86	0.73
	MT	retweet	6.59	2.17	2.50	8.12	0.85	0.72
NAAM + BiLSTM layer	ST	mention	**3.77**	**1.60**	2.88	**6.14**	**0.90**	**0.81**
	ST	retweet	**4.90**	1.95	2.48	**7.00**	**0.89**	**0.79**
	MT	mention	4.30	1.79	3.31	6.56	0.88	0.78
	MT	retweet	5.11	1.93	2.68	7.15	0.88	0.78

Table 1: Next day forecasting results for retweets and mentions for Twitter (training window = 7 days) for a single task (ST) and multi task (MT) setups obtained using our node-aware attention model (NAAM). The results for the best performing models are highlighted in bold.

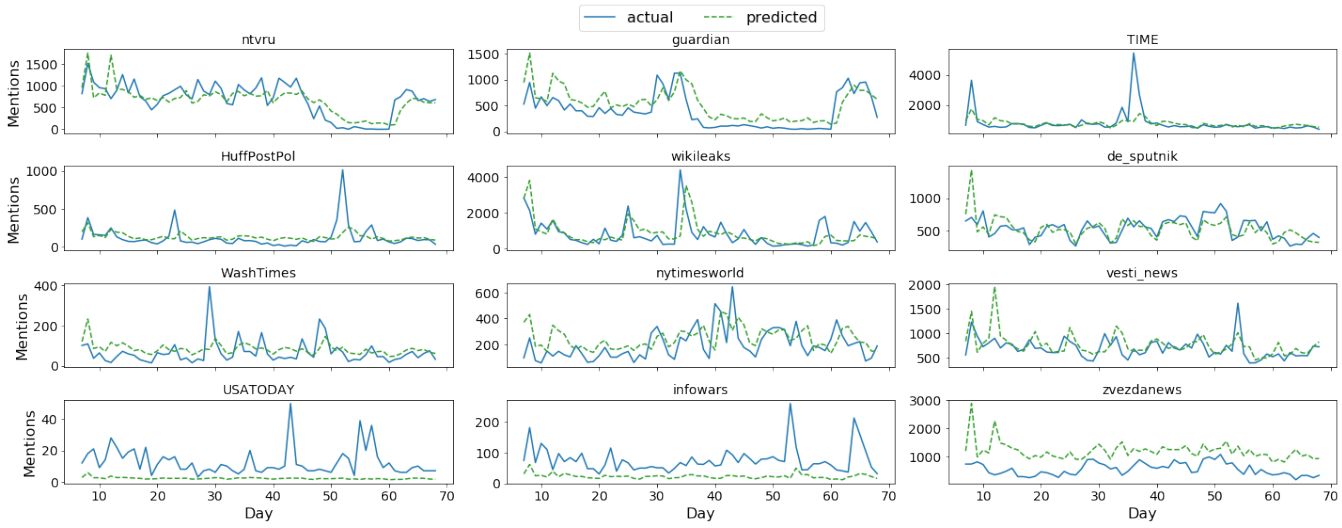


Figure 2: Forecasting results for the number of mentions (actual and predicted) for example news sources on Twitter.

8M dataset.[2] For our experiments, we subsampled videos with activity in at least 20% of the days over that timeframe and relied on the largest connected component, which resulted in a list of 320 videos and 398,989 users who commented on them. Similar to Twitter, we projected the resulting YouTube bipartite graph on the video side. For each video, we again took a week's worth of comment activity and forecasted the comments received on the next day. We set the train period from November 7, 2013 to November 30, 2015 and the test period from July 1, 2016 to February 21, 2017. The rest was used as validation.

For Youtube, table 2 shows the results for forecasting the comment volume for 320 videos. We also show results for the previous day baseline, which was the strongest baseline out of all four in the Twitter forecasting experiments. Unlike Twitter, we only have a single task setting for YouTube to forecast the comment activity for videos. Our model with a BiLSTM layer is again able to beat the baseline with an R^2 of 0.72. Except for MEAE, this model outperforms both the baseline and our model with the LSTM layer for all

[2]https://research.google.com/youtube8m/index.html

other metrics. The LSTM model also comes close to the baseline and is able to beat the baseline for MEAE. The model is good at picking up signals where there is enough activity in a video. For videos with low activity, the previous day baseline is strong since most days have zero activity.

Visualizing the number of actual versus predicted comments for some example videos on YouTube. We visualized the forecasting results for some example videos as well in Figure 3. Again, our results closely emulate the actual comment activity on YouTube for most videos. Although it is hard for our model to forecast high peaks, it can still match the points where there are likely to be spikes in user-video interactions. For videos with high activity, for example, the first video in the figure, our model does fairly well as the predicted comment activity is close to the actual comment activity for that video. For videos where the activity is low, as in the last video in the figure, the model forecasts periodic highs and lows with low variability as a consequence of incorporating the day of the week vector into the model.

5.2 Forecasting Repository Interactions on Github

For the GitHub social graph, we took advantage of a large publicly available GitHub dataset[3]. We considered the GitHub archive from January 2015 to September 2017, especially focusing on the 27,344 repositories that mentioned three cryptocurrencies in particular – Bitcoin, Monero, or Ethereum in the topic field or description. Cryptocurrencies have recently been in the forefront as something that holds much potential and yet at the same time seeming like prime examples of a disruptive piece of technology. We looked at 10 event types explained in [11] that include Watch, Fork, Issue and Issue Comment, Push, Commit Comment, Pull Request, and Pull Request Review Comment events. We subsampled repositories for the same reason as Twitter news sources, but since this data was collected over a longer time period, we selected the repositories that had activity in at least 10% of the days in that timeframe resulting in 364 repositories and 92,537 users. We projected the resulting GitHub bipartite graph on the repository side and forecasted the total event activity for the repositories. We set the train period from January 1, 2015 to July 31, 2016 and the test period: February 1, 2017 to August 31, 2017, using the rest as validation.

For Github, we present the results for forecasts one day in advance for 364 repositories in the cryptocurrency ecosystem in Table 3. Similar to YouTube, the model forecasts on a single task. We can see from the table that similar as in the Twitter domain, our NAAM with the BiLSTM layer obtains the best R^2 of 0.69 and MAE of 2%. The previous day baseline has an R^2 of 0.66. For Github event forecasting, the previous day baseline again has a strong performance because the level of activity across users for certain repositories remains consistent from day to day. Our model outperforms the previous day baseline in most metrics for this dataset as well. Our model with the LSTM layer also outperforms the baseline when the performance is measured using R^2 and comes very close to the baseline for most other metrics.

[3]https://www.githubarchive.org/

Visualizing actual versus predicted user-repository interactions for some example repositories on GitHub. Similar to the results for Twitter, we can see from Figure 4 that the forecasted trend of user-repository interaction events closely follows the actual trend for Github as well for most repositories. The model does very well for the Ethereum repository, which has a high level of interaction activity throughout the time period. For repositories that have very low activity throughout the time period, our model forecasts periodic highs and lows owing to the day of the week vector being fed to our model. When the actual activity signal is low, our model falls back on the trends for the days of the week.

5.3 Relating Graph Characteristics and Forecasting Performance

Even though our model performs well on all three datasets, the results vary across them, not only from our model but also from the previous day baseline model. It seems like there are inherent properties of some graphs that make them easier to forecast than the others. In order to investigate the differences in the graphs of our social environments, we also looked at their topological characteristics in Table 4. We noticed marked differences in the graphs from the three networks.

Graph size. The factor that seems to most heavily affect the performance of our model is the number of nodes in the projected graph. There are 202 news sources in the Twitter graphs, 320 videos for the Youtube graphs, and 364 repositories for the GitHub graphs. Our model's performance is highest for Twitter (MAE 1.6%) and is lower in the order of node counts for YouTube and GitHub (2.19% and 2%). However, we should also note that although the GitHub graphs have nearly double the number of repositories than the number of news sources for Twitter graphs, the difference in performance is comparatively lower between them. Our node attention method creates tailored representations for each node from the aggregate graph structure representations and this is likely the reason why the performance does not see a large decline with the increase in the number of nodes.

Density. The Twitter graph has the highest *density* (0.89), compared to GitHub (0.49) and YouTube (0.42), respectively. The average weighted degree is also higher by an order of magnitude over the GitHub and YouTube graphs. We also obtain the best results for the Twitter graph when compared across models. Although the baseline performance is higher for the Twitter graph, the improvement over the baseline is also highest for Twitter. The Twitter graphs have more user activity in general, also evidenced by the average degree and the average weighted degree, and so there are more connections (the Twitter graph is denser compared to YouTube and GitHub graphs), which in turn means there is a stronger signal from which our model can learn. There are fewer users in our GitHub graph than in the YouTube graph. Since we are exclusively dealing with cryptocurrency repositories on GitHub, it is likely that the same groups of users contribute to the similar repositories. However, the *average degree and the average weighted degree* for GitHub is higher compared to that for YouTube, likely because there are less shared interests between the users on YouTube.

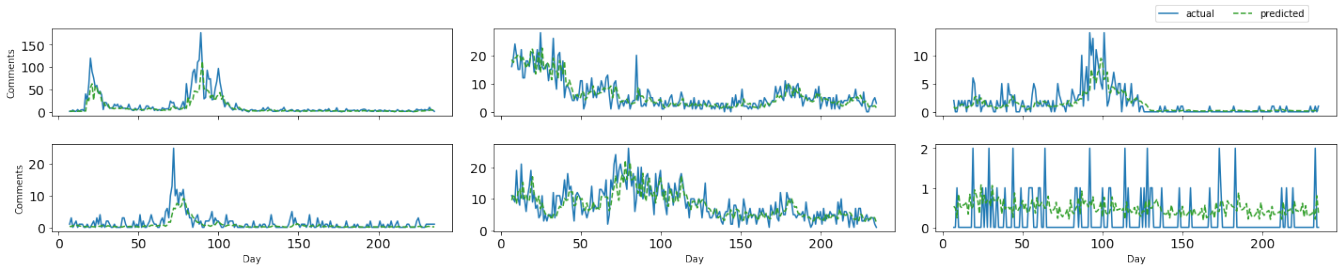


Figure 3: Forecasting results for the number of comments (the actual and predicted) for example videos on YouTube.

Model	MSE × 10^{-5}	MAE × 10^{-3}	MEAE × 10^{-4}	RMSE × 10^{-3}	Pearson	R^2
Previous day baseline	2.57	2.52	7.81	5.07	0.85	0.69
NAAM + 2 LSTM layers	2.84	2.38	**1.27**	5.33	0.84	0.66
NAAM + BiLSTM layer	**2.36**	**2.19**	9.93	**4.85**	**0.86**	**0.72***

Table 2: Next day comment forecasting results for Youtube (training window = 7 days). The results for the best performing models are highlighted in bold. *Significant over the previous day baseline at $p < 0.05$

Model	MSE × 10^{-5}	MAE × 10^{-3}	MEAE × 10^{-4}	RMSE × 10^{-3}	Pearson	R^2
Previous day baseline	3.98	**1.89**	**3.30**	6.31	0.83	0.66
NAAM + 2 LSTM layers	3.84	2.05	7.54	6.20	0.82	0.67
NAAM + BiLSTM layer	**3.62**	1.98	7.00	**6.02**	**0.83**	**0.69****

Table 3: Next day event forecasting results for GitHub (training window = 7 days). The results for the best performing models are highlighted in bold. **Significant over the previous day baseline at $p < 0.01$

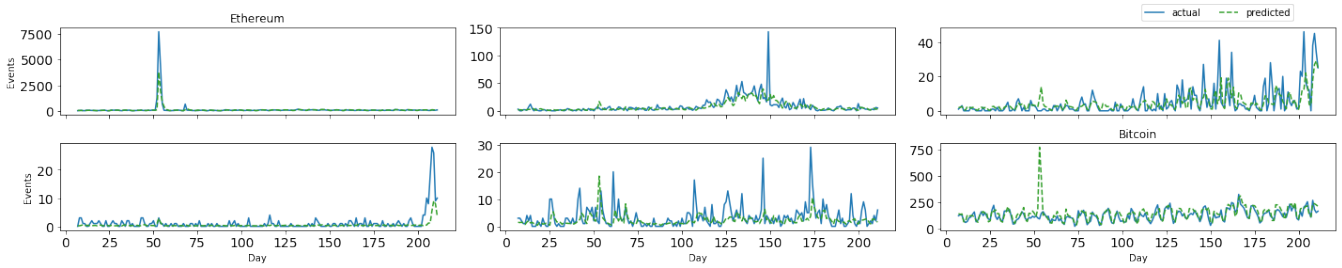


Figure 4: Forecasting results for the number of user-repo interactions (the actual and predicted) for example repositories on GitHub.

Homophily. The *degree assortativity coefficient* represents to what extent nodes in a graph associate with other nodes of similar degree, being of similar sort or being of opposing sort. We found that nodes on Twitter and GitHub have less homophily compared to YouTube. GitHub has the lowest degree assortativity coefficient for GitHub of -0.13. On GitHub there we have a main set of popular repositories for each coin, like the Bitcoin and Ethereum repositories. Many users might be interested in only one coin and thus there might not be many links between these repositories. Also, the most attention and activity must be received by these popular repositories for each coin rather than their less popular forks, for example. We can also see this in Figure 4, where Ethereum and Bitcoin have a very high volume of events compared to others. This might make it hard for our model to forecast on the low signal repositories. Interestingly, the YouTube graph has the highest homophily and the longest average path (5 degrees of separation compared to 2 for Twitter and GitHub) - but the forecasting performance is lower than that for Twitter and higher than that for GitHub.

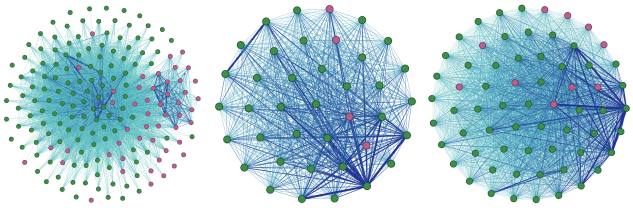
Social relationships. Twitter graph has higher *average clustering coefficient and transitivity* compared to GitHub and YouTube. It demonstrates that Twitter is a small-world network with a large clustering coefficient, small average path (1.93 compared to 2.23

Graph	Density	Average Degree	Average Weighted Degree	Degree Assortativity Coefficient	Average Clustering Coefficient	Transitivity
Twitter	0.89	178.93	57,905.06	-0.08	0.93	0.93
YouTube	0.42	44.10	62.88	0.21	0.52	0.75
GitHub	0.49	176.81	1,888.24	-0.13	0.83	0.78

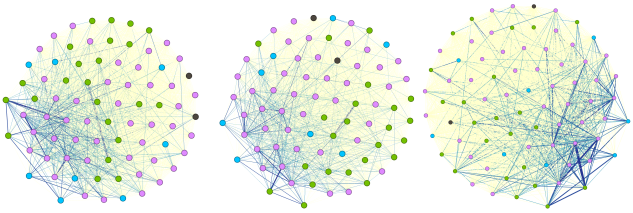
Table 4: Twitter, GitHub and YouTube graph topological characteristics.

and 5.02 for GitHub and YouTube, respectively) compared to two other graphs. Twitter has tighter social relationships than YouTube and GitHub.

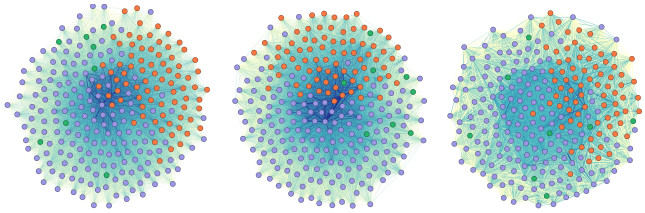
Training period length. The YouTube graph has a training period that is over five months longer than that of GitHub and more than three times than that of Twitter. This does seem to be of an advantage since the results on the YouTube graphs are better than for GitHub, even though the YouTube graphs are sparser. Although there is more training data from longer periods of time for both GitHub and YouTube than for Twitter, our model cannot overcome the sparseness of the adjacency matrix for the former two datasets, as these are its main sources of input.



(a) Dynamic social graphs for Twitter (degree > 100): February 2, 2016 (left), September 1, 2016 (middle), November 1, 2016 (right).



(b) Dynamic social graphs for Youtube (degree >50): November 7, 2013 (left), December 1, 2015 (middle), and July 1, 2016 (right).



(c) Dynamic social graphs for GitHub (degree >134): January 1, 2015 (left), August 1, 2016 (middle), and February 1, 2017 (right).

Figure 5: Daily social interaction graphs for the Twitter, Youtube, and GitHub for three example days: the beginning of the training, validation and testing periods. Nodes are colored as red = unverified, green = credible news sources for Twitter; pink = art and entertainment videos, green = game videos, blue = food and drink videos, black = beauty and fitness videos, orange = sports videos for YouTube; green = Monero repositories, red = Ethereum repositories, purple = Bitcoin repositories for GitHub. Edge color and width represent the edge weights.

Model	Forecast	Pearson	R^2
Our best model – historical	mention	0.84	0.70
	retweet	0.85	0.72
NAA replaced with self-attention	mention	**0.90**	0.80
	retweet	0.88	0.78
NAAM + BiLSTM	mention	**0.90**	**0.81**
	retweet	**0.89**	**0.79**

Table 5: Ablation study of different NAAM model variations showing forecasting performance.

6 ANALYZING OTHER FACTORS AFFECTING MODEL PERFORMANCE

We saw that our models are generalizable across various social environments, although there were certain factors that made the forecasting problem fundamentally harder for some graphs. We also wanted to investigate how changes in the problem setup and in the model itself might affect the forecasting performance. Here we look at the effects of removing some components of our model as well as the effects of varying training as well as forecasting window sizes.

Experimenting with NAAM model variants. We first investigated the effects of removing or replacing parts of our model (ablation study). Table 5 shows the results from our best model and compares them to the different variations. The first variant is the removal of the historical information, including old retweet and mention counts as well as the day of the week information. Although the model still performs well, the results clearly show that historical information is crucial to the problem. The previous day baseline results, which uses only the previous day's retweet and mention distributions also support this finding. On the other hand, replacing node-aware attention (NAA) by self-attention does not seem to have a similar amount of deprecating effect on the results. The results are lower than our model's best performance, but not by a large margin. This could possibly be attributed to the fact that the final representation still retains the learned node embedding

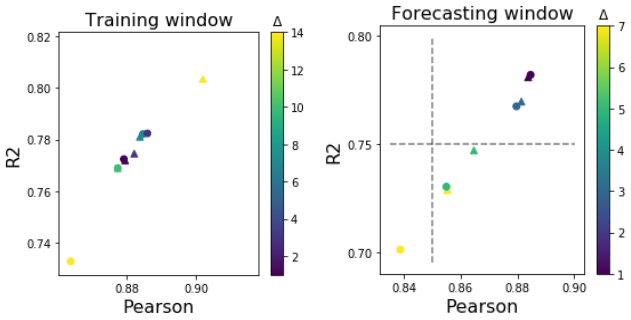


Figure 6: The effects of varying training and forecasting window sizes on model performance for the for NAAM + BiLSTM layer. The results for mention forecasts are shown as triangles, and retweet forecasts as circles.

information in this variant of NAAM and thus it is still node-aware in some way. Finally, the results also show that the combination of historical information and node attention is beneficial to our model.

Evaluating the effect of the training and forecasting window size on model performance. In our original setup, we used a training window of a week (7 days) and a forecasting window of one day i.e. we forecasted the activity for the next day given the previous 7 days. We wanted to see how the model performance changes when we vary the training window size. We evaluated the effect of both the increase and the decrease of the training window size. The change in the size of the training window can have contradictory effects on model performance. On one hand, if we take a longer training window, the model can see additional past behavior and can learn from it to make more informed decisions. On the other hand, the retweets and mentions for a given day are more likely to resemble those of the days closest to it. Taking this into consideration, a longer training window can also hinder the decisions of the model by creating confusion due to unrelated past behavior.

To investigate this effect, we ran our models with training window sizes of 14, 10, 3, and 1 as well. The results for each of these window sizes are shown in Figure 6. For mentions, the largest window size of 14 results in the best R^2 of 0.80 and a Pearson of 0.91 while for the retweets, a window size of 3 has the best R^2 of 0.78. Mentions are sparser than retweets in our data, which is probably why a longer training window helps mention prediction. The window size of 1 gives an R^2 value of 0.77 for both mentions and retweets. This analysis shows that although past history is important, it is not always the case that a longer look into the past is going to help forecasting.

Similar to experimenting with variable training window sizes, we also varied the forecasting window size. Different from the above experiment with training window size, forecasting further into the future has a clear effect on model performance – it becomes a gradually more difficult task as we increase the forecasting window. We can see these effects in Figure 6 (right). It is easiest to forecast both retweets and mentions when doing so for just one day in advance. As expected, forecasting user behavior the next day yields the highest R^2 and Pearson correlation than forecasting user behavior a higher number of days in advance. Although the performance does decrease as we predict further into the future, the performance degradation is much slower than what we had expected, evidence that our model is robust to generating forecasts for more than one day in advance.

7 RELATED WORK

There have been many research forays into learning representations for nodes in a graph. Some well-known methods to learn node representations of static graphs include Node2Vec [12], DeepWalk [23], LINE [24], and SINE [31]. Since these methods are transductive and are unable to generate representations for nodes that were not seen during training, Hamilton et al. [13] proposed GraphSAGE, which can use node features and neighborhood information to deduce representations for unseen nodes. All these methods aim at static graphs and do not take into account the evolving nature of real-world social interaction graphs – dynamic changes in graph connectivity information.

Recently, the focus has shifted towards dynamic graphs and methods have also been proposed to learn node embeddings for these graphs. The continuous-time dynamic networks method extends the concept of a RandomWalk used in both Node2Vec and DeepWalk by introducing temporal random walks [21] as a way to learn time-preserving embeddings. Similarly, the DynamicTriad method uses the inclination of the two unconnected nodes in an open triad to connect in the future based on their proximity to each other in order to learn dynamic network embeddings [33]. Although these methods work with dynamic graphs, they mostly serve as a first step towards specific downstream prediction tasks such as node classification, link prediction, and community detection. In contrast, our model learns dynamic changes in graph connectivity information for the forecasting task directly, while also learning node embeddings in the process.

In addition to node-embedding based models, other types of deep learning models have also been applied for graph analytics [7]. These include convolutional neural networks (CNNs) [6, 17], graph attention networks [26], graph recurrent neural networks (RNNs) [29], and neural message passing [8, 15]. Recently, Graph Isomorphism network has been proposed, which provably maximizes the discriminative power of graph neural network [28]. However, these approaches have mostly been evaluated on predictive tasks over static graphs and benchmark datasets except some recent work [25, 29]. Yu et al. [30] use the information from historical adjacency matrices to directly predict future adjacency matrix by using matrix factorization. Although this method can perform link prediction directly, for all other tasks, it still requires a two-step process. To the best of our knowledge, there is limited prior work on applying deep learning models to enable *forecasting capabilities over dynamic social graphs*. Moreover, most of the existing models for making inference from dynamic graphs neither focused on a diverse set of social interactions graphs nor applied their modeling techniques to truly forecasting tasks of user behavior and future interactions.

8 SUMMARY

In this paper, we proposed a novel task – forecasting over dynamic social graphs and a novel node-aware attention model that forecasts user interactions in real-world dynamic graphs. Our model yields the best performance when forecasting mentions (MAE 1.6%, Pearson 0.90, R^2 0.81) and retweets (Pearson 0.89, R^2 0.79) of news sources on Twitter. We also showed that it is beneficial to use a unified model rather than a two-step process of first learning embeddings for nodes independently using the state-of-the-art embedding techniques such as Node2Vec and DeepWalk, and then using them for the downstream forecasting task. We also demonstrated the generalizability of our model on two more social graphs with different characteristics. We were able to obtain better performance over the baseline for these graphs when we forecasted video comments on YouTube (MAE 2.19%, Pearson 0.86, R^2 0.72), and repository interactions on GitHub (MAE 1.89%, Pearson 0.83, R^2 0.69). We also saw that graph and dataset characteristics directly affect the forecasting difficulty for a social environment. We report that when forecasting social interactions on a smaller size, denser graphs like Twitter our model performs significantly better than forecasting interactions on GitHub and YouTube. We also demonstrate the interesting effects that the training and forecasting window sizes can have on the performance and the effect is not always linear with the training window.

REFERENCES

[1] Peter J Brockwell, Richard A Davis, and Matthew V Calder. 2002. *Introduction to time series and forecasting*. Vol. 2. Springer.

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014.*

[3] Dorota Celińska. 2018. Coding Together in a Social Network: Collaboration Among GitHub Users. In *Proceedings of the 9th International Conference on Social Media and Society (SMSociety '18)*. ACM, New York, NY, USA, 31–40. https://doi.org/10.1145/3217804.3217895

[4] Hsinchun Chen, Xin Li, and Zan Huang. 2005. Link prediction approach to collaborative filtering. In *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries, 2005(JCDL)*. IEEE, 141–142.

[5] Ronan Collobert and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*. 160–167.

[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*. 3844–3852.

[7] Li Deng, Dong Yu, et al. 2014. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing* 7, 3–4 (2014), 197–387.

[8] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. *arXiv:1704.01212* (2017).

[9] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research)*, Yee Whye Teh and Mike Titterington (Eds.), Vol. 9. 249–256.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[11] Georgios Gousios and Diomidis Spinellis. 2012. GHTorrent: GitHub's data from a firehose. In *Proceedings of Mining Software Repositories*. 12–21.

[12] Aditya Grover and Jure Leskovec. 2016. Node2Vec: Scalable Feature Learning for Networks. In *Proceedings of ACM SIGKDD*. 855–864.

[13] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1025–1035.

[14] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584* (2017).

[15] Isaac Henrion, Johann Brehmer, Joan Bruna, Kyunghun Cho, Kyle Cranmer, Gilles Louppe, and Gaspar Rochette. 2017. Neural Message Passing for Jet Physics. (2017).

[16] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). http://arxiv.org/abs/1412.6980

[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[18] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-Normalizing Neural Networks. *CoRR* abs/1706.02515 (2017). http://arxiv.org/abs/1706.02515

[19] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. 1998. Efficient BackProp. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*. 9–50.

[20] Antonio Lima, Luca Rossi, and Mirco Musolesi. 2014. Coding together at scale: GitHub as a collaborative social network. In *Eighth International AAAI Conference on Weblogs and Social Media*.

[21] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-time dynamic network embeddings. In *3rd International Workshop on Learning Representations for Big Networks (WWW BigNet)*.

[22] Ping-Feng Pai and Chih-Sheng Lin. 2005. A hybrid ARIMA and support vector machines model in stock price forecasting. *Omega* 33, 6 (2005), 497–505.

[23] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *Proceedings of ACM SIGKDD*. 701–710.

[24] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web (WWW '15)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 1067–1077. https://doi.org/10.1145/2736277.2741093

[25] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-Evolve: Deep Temporal Reasoning for Dynamic Knowledge Graphs. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Doina Precup and Yee Whye Teh (Eds.), Vol. 70. 3462–3471.

[26] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph Attention Networks. *arXiv preprint arXiv:1710.10903* (2017).

[27] Svitlana Volkova, Kyle Shaffer, Jin Yea Jang, and Nathan Hodas. 2017. Separating Facts from Fiction: Linguistic Models to Classify Suspicious and Trusted News Posts on Twitter. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 647–653.

[28] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations (ICLR)*.

[29] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Model. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*. 5694–5703.

[30] Wenchao Yu, Charu C Aggarwal, and Wei Wang. 2017. Temporally factorized network modeling for evolutionary network analysis. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 455–464.

[31] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2018. SINE: Scalable Incomplete Network Embedding. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 737–746.

[32] G Peter Zhang. 2003. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing* 50 (2003), 159–175.

[33] Le-kui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. 2018. Dynamic Network Embedding by Modeling Triadic Closure Process. In *AAAI*.

[34] Tao Zhou, Jie Ren, Matúš Medo, and Yi-Cheng Zhang. 2007. Bipartite network projection and personal recommendation. *Physical Review E* 76, 4 (2007), 046115.