DeepDiffuse: Predicting the 'Who' and 'When' in Cascades

Mohammad Raihanul Islam, Sathappan Muthiah, Bijaya Adhikari, B. Aditya Prakash, Naren Ramakrishnan
Department of Computer Science, Virginia Tech, USA
Discovery Analytics Center, Virginia Tech, USA
Email: {raihan8, sathap1, bijaya, badityap, naren}@cs.vt.edu

***Abstract*—Cascades are an accepted model to capturing how information diffuses across social network platforms. A large body of research has been focused on dissecting the anatomy of such cascades and forecasting their progression. One recurring theme involves predicting the next stage(s) of cascades utilizing pertinent information such as the underlying social network, structural properties of nodes (e.g., degree) and (partial) histories of cascade propagation. However, such type of granular information is rarely available in practice. We study in this paper the problem of cascade prediction utilizing only two types of (coarse) information, viz. which node is infected and its corresponding infection time. We first construct several simple baselines to solve this cascade prediction problem. Then we describe the shortcomings of these methods and propose a new solution leveraging recent progress in embeddings and attention models from representation learning. We also perform an exhaustive analysis of our methods on several real world datasets. Our proposed model outperforms the baselines and several other state-of-the-art methods.**

***Keywords*-Social Networks; Cascade and Diffusion Prediction**

I. Introduction

Content sharing (e.g., of photos and videos) through online social networks is so prevalent that cascades of propagation have become the unit of discourse to study such phenomena. Diverse applications in areas like social media, public health, cybersecurity, and critical infrastructure, now benefit from cascade models. A growing body of research has developed around understanding the structural properties of cascades, and engineering their growth, with applications in areas such as viral marketing, crowdsourcing, rumor detection, and immunization.

There has been a recent spate of interest in predicting cascades in various contexts, e.g., tweets, memes, videos, movie ratings and academic papers [1]–[3]. Most work has looked into predictions using epidemiologically-inspired generative models [4] or using a mixture of structural, content, and temporal features [5]. Some recent works have also used survival analysis [6], [7] or deep learning based models [8], [9]. Studies have looked into different aspects of cascades including macroscopic metrics like the possibility of becoming viral [7] or the future size/volume of the cascade [8], [10], [11], and microscopic aspects like predicting the *next* node taking part in the diffusive process [9]. In general, in all these works, more finer-grained analysis has in-turn needed varying degrees of more auxiliary information such as assuming that the source of infection is available [8], or knowing the degree of the participating nodes [10], or even the full underlying graph structure itself [9]. Much of this information is not necessarily available, or is noisy or hard to construct [7], [12]–[14]. Additionally, almost all of the above works ignore the associated temporal information [8], [9].

In this paper, we extend the state-of-the-art in purely data-driven cascade analysis in multiple directions. We present a new approach to explore the diffusion dynamics of the cascade comprehensively *by only leveraging the observable cascade information*. Broadly, we conduct a finer grained-analysis while using *less* information than some macroscopic approaches. At the same time, we leverage temporal infor-

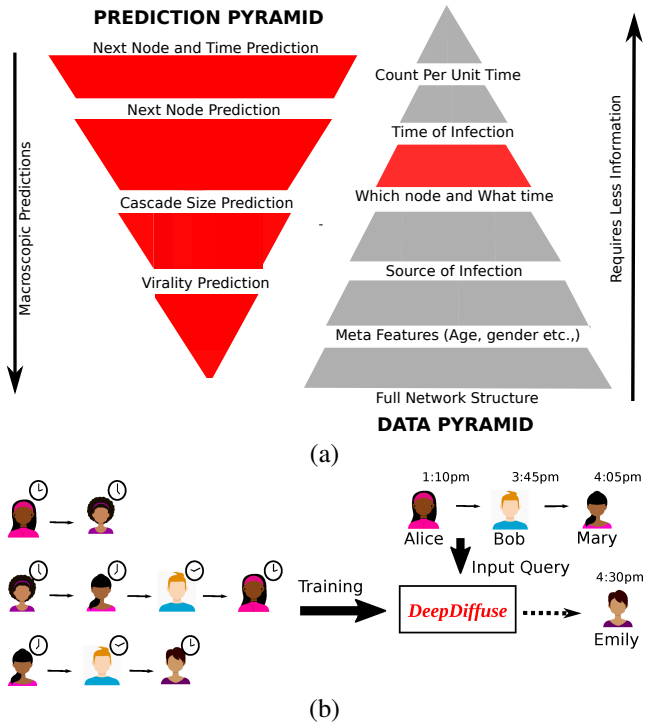

Fig. 1. (a) A pyramid view of the kind of data used and the type of problems pursued in cascade analysis. The left pyramid gives the four major tasks with granular level tasks at the top. The right pyramid details the type of data used where the number of data features increase as we go down. Our proposed method *DeepDiffuse* (in red color) is well situated at the middle of the data pyramid while capable of performing all tasks in the prediction pyramid. (b) A sample input and output of *DeepDiffuse*.

mation as well to do more meaningful predictions. In more detail, our approach only utilizes the sequence of infected nodes and their infection time to predict both (a) *which* node is going to get infected and (b) *when*. Our method does not require additional structural or content-based information (e.g., neighborhood information or number of friends) or which node is responsible for the infection (see Fig. 1(a)). To the best of our knowledge our work is the first of its kind to tackle such a problem.

Our contributions are as follows:

1) **Novel Problem Formulation:** We define the novel problem of predicting diffusion dynamics in real world scenarios based on realistic assumptions. The underlying problem is to predict when and who is going to be infected in a social network based on (only) previously observed cascades. We only observe the times when the nodes get infected, but we do not have any knowledge on who has infected them or who are their neighbors.

2) **New Solution Strategies:** We leverage recent progress in embeddings and attention models from representation learning to propose a family of new models which can effectively predict when the next infection will happen and who will be infected. First we develop a baseline model for the prediction. This helps identify its weaknesses and motivates our proposed solution *DeepDiffuse*.

3) **Rigorous Empirical Evaluation:** We conduct a comprehensive set of experiments to evaluate the effectiveness of *DeepDiffuse* over the baseline model and other state-of-the-art methods proposed for similar problems.

II. Our Approach

A. Problem Formulation

The input to our problem is a set of cascades $\mathcal{C} = \{c^1, c^2, \dots c^n\}$ on a set of nodes V. Each cascade $c^i \in \mathcal{C}$ is a sequence of tuples (v_j^i, t_j^i), where $t_j^i \in \mathbb{R}^+$ is the time when a previously uninfected node v gets infected. We define this node as v_j^i (j^{th} node in i^{th} cascade). Here infection means this node takes part in the cascade process (i.e. reshare a content in social media etc.) Moreover time point is monotonic within a cascade (i.e. $t_j^i < t_{j+1}^i$). We assume that a node can only be infected once. Then we define the cascade prediction problem in the following way:

Cascade Prediction Problem: Given a set of cascades $\mathcal{C}$, learn a model $\mathcal{M}_\mathcal{C}$ so that we can predict the next infection tuple (v_{j+1}^k, t_{j+1}^k) given a test cascade instance $c^k = \{(v_1^k, t_1^k), \dots, (v_j^k, t_j^k)\}$. Due to the lack of explicit network structure, we assume that the predicted node will be one of the observed nodes in $\mathcal{C}$ (i.e. $v_{j+1}^k \in V$). An illustrative example is shown in Fig. 1(b).

Solution Sketch: We adopt a temporal point process formulation: suppose within a small window between t and $t + dt$, $\lambda^*(t)dt$ is the probability of a new event occurring (i.e. node infection) given the history of infections: $\mathbf{H}_t$. Now $\lambda^*(t)dt = Prob\{\text{node infected in [t,t+dt)} \,|\, \mathbf{H}_t\}$ We can then specify the conditional density function since the last infection t_n by

$$f^*(t) = \lambda^*(t) \exp\left[-\int_{t_n}^{t} \lambda^*(t)dt\right] \tag{1}$$

We adopt a recurrent neural network (RNN) approach here that can capture more complex state transitions than classical probabilistic methods and further provide more flexible non-linear transition functions. Here in our problem we use a LSTM. Empirically it is shown that LSTM performs very well in terms of sequence learning tasks commonly encountered in natural language processing.

B. A vanilla LSTM Model:

Suppose we have a cascade sequence $S = \{(v_1, t_1) \dots (v_n, t_n)\}$. As long standing practice, we represent each node in the sequence as a vector rather than an index. The embedding of all the nodes are stored in $\mathbf{X}_v \in \mathbf{R}^{d_v \times |V|}$, where d_v is the size of embedding for nodes. Now to extract the embedding of a specific node v_q we can use a one-hot vector q.

$$x_q = \mathbf{X}_v \cdot q \tag{2}$$

where $|q| = |V|$. The embedding matrix $\mathbf{X}_v$ will be also be learned during training. Now for the j^{th} element of the input sequence (v_j, t_j), suppose the embedding of v_j is x_{v_j} and the temporal feature $d_{t_j} = t_j - t_{j-1}$ (i.e. inter-infection duration). Then the LSTM equations are as follows:

$$i_j = \sigma(W_v^i x_{v_j} + W_t^i d_{t_j} + U_i h_{j-1} + b_i) \tag{3a}$$
$$f_j = \sigma(W_v^f x_{v_j} + W_t^f d_{t_j} + U_f h_{j-1} + b_f) \tag{3b}$$
$$\tilde{C}_j = \tanh(W_v^c x_{v_j} + W_t^c d_{t_j} + U_c h_{j-1} + b_c) \tag{3c}$$
$$C_j = \tilde{C}_j \odot i_j + f_j \odot C_{j-1} \tag{3d}$$
$$o_j = \sigma(W_v^o x_{v_j} + W_t^o d_{t_j} + U_o h_{j-1} + b_o) \tag{3e}$$
$$h_j = o_j \odot \tanh(C_j) \tag{3f}$$

The hidden state vector h_j captures the dynamics of cascade sequence up to the latest point. We can use it to predict the next infection time and the corresponding node as described below.

Next Node Prediction: The score for a node getting infected can be computed by the following equation $w_{u_{j+1}} = V_u h_j + b_u$. Here $V_u \in \mathbb{R}^{|V| \times K}$ and $b_u \in \mathbb{R}^{|V|}$. Now we can use a softmax layer to compute the probability of each node v_u getting infected.

$$p_{u_{j+1}|h_j} = \frac{\exp(w_{u_{j+1}})}{\sum_{z \in V} \exp(w_{z_{j+1}})} \tag{4}$$

Time Prediction: For time prediction we use the approach described in Du et al. [15]. From the hidden sate h_j we can compute the conditional intensity the following way:

$$\lambda^*(j) = \exp\left(\alpha_t^T \cdot h_j + \beta_t(t - t_j) + \gamma_t\right) \tag{5}$$

Here $\alpha_t \in \mathbb{R}^k$ and $\beta_t \in \mathbb{R}$, $\gamma_t \in \mathbb{R}$ are scalars. The term $\alpha_t^T \cdot h_j$ determines the accrued influence from past infections, whereas $\beta_t(t - t_j)$ focuses on the current infection. The last

term represents the base intensity level. Now we can compute the conditional density function:

$$f^*(j) = \exp\left[\alpha_t^T \cdot h_j + \beta_t(t - t_j) + \gamma_t + \frac{1}{\beta}\exp(\alpha_t^T \cdot h_j + \gamma_t) - \frac{1}{\beta}\exp(\alpha_t^T \cdot h_j + \beta_t(t - t_j) + \gamma_t)\right] \tag{6}$$

We estimate the time of next estimation using the following equation:

$$\hat{t}_{j+1} = \int_{t_j}^{\infty} t \cdot f^*(j)dj \tag{7}$$

Now the full objective function for a single of cascade c^k, where $c^k = \{(v_1^k, t_1^k), \dots (v_n^k, t_n^k)\}$ including both timing and node prediction is as follows:

$$\mathcal{O}_k = \max_{\theta} \sum_{j=1}^{n} \left[\log(p^k_{u_{j+1}=v_{j+1}|h_j}) + \log f(d^k_{j+1}|h_j)\right]$$

where θ is the set of all model parameters. An alternative formulation is to minimize the negative log-likelihood. Therefore our objective becomes:

$$\mathcal{O} = \min_{\theta} \left(- \sum_{c_k \in \mathcal{C}} \sum_{j=1}^{n-1} \left[\log\left(p^k_{u_{j+1}=v_{j+1}|h_j}\right) + \log f(d^k_{j+1}|h_j)\right]\right) \tag{8}$$

C. Cascade Dynamics in Networks: *Introducing DeepDiffuse*

One of the solutions to this problem is to compute the output of the LSTM not just by the last state but as a weighted summation of all the hidden states. This type of technique is known as an *attention mechanism* and is used widely in language modeling like neural machine translation [16]. Our new model incorporating attention is termed *DeepDiffuse*. For this, we introduce $W_\alpha \in \mathbb{R}^{K \times K_\alpha}$ and $u_\alpha, b_\alpha \in \mathbb{R}^{K_\alpha}$. Here K_α is the attention size.

$$u_{js} = \tanh(W_\alpha^T h_{js} + b_\alpha) \tag{9a}$$

$$\alpha^a_{js} = \frac{\exp(u_{js}^T u_\alpha)}{\sum_z \exp(u_{jz}^T u_\alpha)} \tag{9b}$$

$$\tilde{h}_j = \sum_z \alpha^a_{jz} h_{jz} \tag{9c}$$

Here h_{js} is the s^{th} hidden state while processing the j^{th} element of the cascade ($s < j$). We can obtain a context vector $\tilde{h}_j$ as a weighted sum of previous hidden states using Eqn. 9c. In this way the model learns to attend which node to concentrate. Hence, despite the absence of graph structure the model will be able to make an informed choice to improve its predictive power.

D. Bringing Flexibility to the model: *Enhancing DeepDiffuse*

First, the attention mechanism, unlike text modeling applications, is quite expensive to compute for cascade analysis (compare cascade lengths versus sentence lengths). **Second,** the attention model does not provide enough flexibility to the model. For instance when analyzing a long sequence of cascades the model does not provide a provision to leverage or build off an earlier portion of the cascade.

To deal with these shortcomings, we propose some enhancements to our model *DeepDiffuse* to provide more flexibility to its attention mechanism. Inspired from the mechanism of how human pay attention [17], our proposed model analyze one portion of the cascade at a time and based on its evaluation moves to the next location of the cascade. In this way we can efficiently analyze cascades which are typically longer than sentences. A high-level overview of the enhanced *DeepDiffuse* model is presented in Fig. 2.

The enhanced version consists of two modules. The first module **Cascade Analyzer Network (CAN)** takes a location $l_{j-1} \in \mathbb{N}$ and a cascade sequence s (shown in green in Fig. 2) and extracts a subsequence of the cascade starting from location l_{j-1}. The length of the subsequence is defined by the ratio ρ. For example if $\rho = 0.5$ the length will be $\rho \times |s|$, where $|s|$ is the length of the sequence. The extracted subsequence is processed similar to the basic *DeepDiffuse* model. The context vector of this model along with the location l_{j-1} are converted into two fixed-length vectors $l'_j \in \mathbb{R}^K$ and $g'_j \in \mathbb{R}^K$ respectively (shown in blue color in Fig. 2) using a fully connected network. They are added to get a combined vector representation $r_j \in \mathbb{R}^K$ using a Rectified Linear Unit (ReLU). This vector r_j can be considered as a compressed representation of the extracted cascade subsequence (cascade along with its location). Equations for generating l'_j, g'_j, r_j are given below:

$$l'_j = W_l^r l_{j-1} + b_l^r \tag{10a}$$

$$g'_j = W_g^r \tilde{h}_j^A + b_g^r \tag{10b}$$

$$r_j = \text{ReLU}(l'_j + g'_j) \tag{10c}$$

Here, $W_l^r \in \mathbb{R}^{K \times 1}$, $W_g^r \in \mathbb{R}^{K \times K}$ and $b_l^r, b_g^r \in \mathbb{R}^{K \times 1}$ and $\tilde{h}_j^A$ is the context vector from the attention layer (computed from Eqn. 9c).

The representation vector r_j is now fed to the **Cascade Predictor Network (CPN)**. The CPN has three components. The first component contains an LSTM decoder which takes the vector r_j as input to produce an output vector (hidden state) h_j^P. The second component of CPN contains a location network (fully connected network) which takes h_j^P as input and return the next location l_j as output. The output of the location network is clipped such a way that it always gives a valid location w.r.t. the cascade sequence. The last part of the CPN is the output layer which computes the probability distribution of the node infection and the conditional density function similar to previously introduced models.

The benefit of the enhanced version of *DeepDiffuse* is that it can learn which part of the cascade to analyze without any external supervision. Moreover, since it only requires a subsequence of the cascade (defined by a ratio ρ), the cost of attention mechanism is much lower than that of the basic *DeepDiffuse* model. These two factors make the model more flexible and efficient.

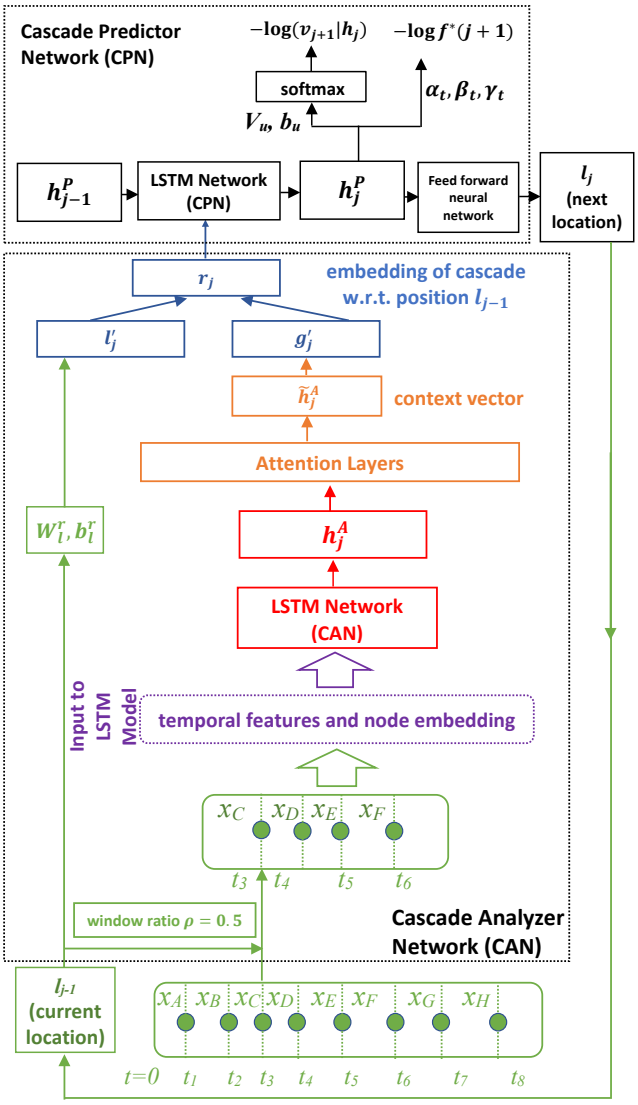

Fig. 2. Architecture of enhanced *DeepDiffuse*. The model has two major components: Cascade Analyzer Network (CAN) and Cascade Predictor Network (CPN). CAN produces an internal vector representation (r_j) of a subsequence of the cascade (in blue color) as per Eqn. 10a-c. This subsequence is extracted from a location l_{j-1} computed in the previous time step of CPN. The processing of the subsequence is same as the basic *DeepDiffuse* model as shown through the green (cascade sequence), violet (input feature), red (LSTM network), and orange (attention) colors. CPN takes this representation r_j and feeds it to another LSTM network which calculates the next location of the cascade to be analyzed (l_j) as well as compute the next node and timing information.

III. Experiments

A. Experimental Setup

Datasets: We conduct our empirical analysis on three publicly available real world datasets. For all the datasets we picked 75% as training and the rest for testing.

- Digg [11] is a news aggregator. The dataset contains votes by its users for 3,553 news articles. The timestamp of votes along with the anonymized user-id form the cascade. It has 82,778 nodes (users) and has an average cascade length of 30.0.
- Twitter [18] dataset contains 569 URLS shared by 5,942 users. The reshare sequence for a URL creates a cascade where the average cascade length is 5.70.
- Memetracker [19] dataset contains 54,847 popular memes published on 4,403 news websites. Sharing time along with the website names form the cascade. The average cascade length is 17.0.

Baselines: We consider several state-of-the-art methods to compare with our proposed *DeepDiffuse*. Since ours is the first in line for this type of cascade prediction, we use modified version of similar methods along with the baseline we developed earlier for comparison. Another notable point is although many cascade prediction models exist not all methods have the same goal or data features as ours. Therefore we consider only the models which use similar data to ours for comparison. It should be mentioned that not all the methods can predict both the node and the time of infection together. Therefore we compare the node and time prediction separately. The short description of the competing methods are given below:

- **RMTP** [15] is an RNN model designed to predict marked temporal point process. This model can predict the timing and the type of next event. For our purpose, we input the infected node and the corresponding infection time as the event marker and the time of the event respectively.
- **TopoLSTM** [9] is an LSTM based model which only predicts the next node given a cascade and its network structure. Since we do not exploit any network information we only input the cascade. We only use it for node prediction since it can not predict the timing.
- **DeepCas** [8] is a Bidirectional GRU based model designed to predict the size of the cascade. We use a softmax layer in the output layer to predict the next node but, it can not predict the infection timing.
- **LSTM** is our vanilla LSTM based model described in Section II-B, which can predict both the next node and timing of the infection.
- ***DeepDiffuse* Variants:** *DeepDiffuse* is our proposed family of methods for both node and infection time prediction. It has two variants–*DeepDiffuse*-Basic (***DeepDiffuse*-B**), described in Sec. II-C, which uses the entire cascade information at all time steps and *DeepDiffuse*-Enhanced (***DeepDiffuse*-E**), described in Sec. II-D, which uses two network modules: a predictor network, to identify the subsequence to analyze and an analyzer network, to process the identified subsequence of the cascade[1].

Parameter setting: Unless otherwise stated, we set the state size K of all the methods to 256, batch size to 50, sequence length to 100, and window to sequence ratio ρ to 0.9.

B. How effective is DeepDiffuse in predicting the node and timing of infection?

Evaluation Metric: Given a cascade sequence obtaining the next infected node can be viewed as a retrieval task since there

[1]code: https://github.com/raihan2108/deep-diffuse

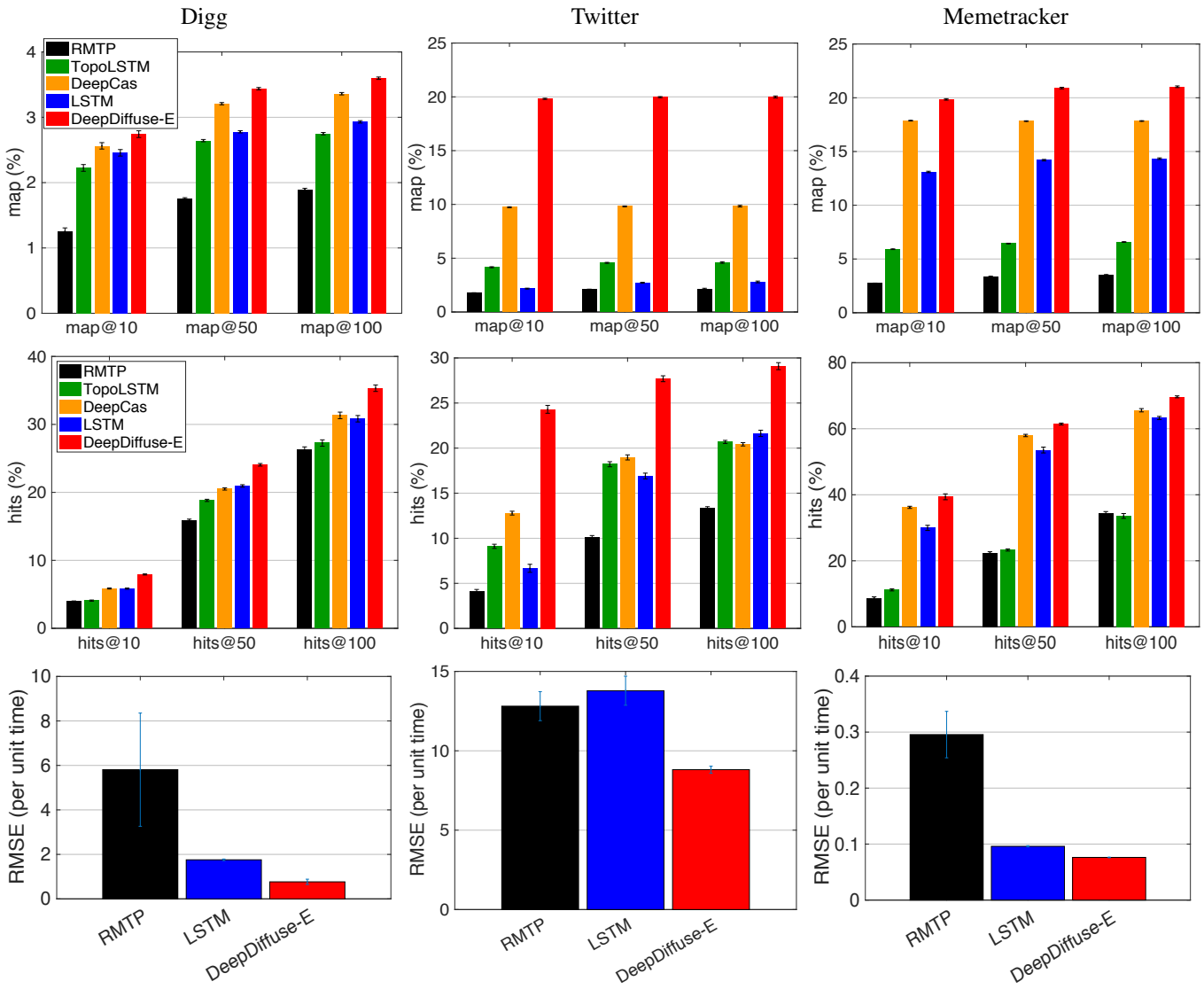

Fig. 3. Comparison of node and time prediction for different datasets. The first and second rows show the map@κ and hits@κ score for node prediction (higher the better). The bottom row shows RMSE score for timing prediction (lower the better). We can see our proposed *DeepDiffuse*-E outperforms all other methods in all metrics. The performance gain is 6–107% across all the methods for hits@100. For timing prediction we achieve a minimum of 20% reduction in RMSE.

can be an arbitrarily large number of potential candidates [9]. Therefore an intuitive way for evaluation is to apply ranking metrics used in information retrieval. To apply such metrics we rank the uninfected nodes by their infection probabilities (from Eqn. 4) and consider the relevant item to be the actual node to get infected. We use two widely popular ranking methods:

- **MAP@κ:** This is the Mean Average Precision used in information retrieval
- **HITS@κ:** The rate of top κ ranked nodes containing the next infected node.

We use $\kappa = 10, 50, 100$ for our evaluation. For the time prediction we compute the estimated time of next infection using Eqn. 7 for the test. Then we compute the RMSE score. The comparison of map@κ and hits@κ (top two rows) and the RMSE score for timing prediction (bottom row) for different datasets is shown in Fig. 3. We can make the following observations.

1) *DeepDiffuse*-E outperforms TopoLSTM and DeepCas by a very good margin. The performance gain ranges from 29% to 107% in case of hits@100 against TopoLSTM whereas w.r.t. DeepCas it is 6% to 42%. This demonstrates the superiority of *DeepDiffuse*-E over current state-of-the-arts methods.
2) *DeepDiffuse*-E also outperforms the LSTM model in all cases. Here performance gain range is 10%–34% in case of hits@100.
3) Performance improves as κ increases since the true infected node is more likely to be in the set of possible candidates if its size increases.
4) *DeepDiffuse*-E also outperforms all the other methods in terms predicting the next time of infection (bottom row, lower the better) demonstrating its effectiveness. As mentioned before TopoLSTM and DeepCas are omitted from this com-

parison as they cannot predict time.
5) Among other observations, DeepCas performs well compared to TopoLSTM. TopoLSTM requires the network structure to create an additional hidden layer. Our hypothesis is because of the absence of network structure in our study TopoLSTM is underperforming.
6) In both node and timing prediction RMTP performs the worst, as it uses simple RNN architecture.

IV. Additional Related Work

Information Diffusion: Established research provides diffusion models to study notions of influence in the network [20], [21] and applications to inference and link prediction [22], [23]. Cascade prediction research is typically one of classifying whether a cascade will go viral in the future [24], [25] or one of regression of size of cascade [26]. **Temporal Point Processes:** Temporal point processes [27] while having been applied to modeling information diffusion [28], [29] in a wide range of domains (e.g. finance [30], sociology [31]), model only the temporal dynamics of the diffusion networks unlike ours. **Attention Mechanisms:** Attention mechanisms are one of the most recent and exciting advances in deep learning but have been primarily utilized for NLP tasks (e.g. [16], [32]) or computer vision (e.g. [33], [34]) unlike our application to cascade prediction.

V. Conclusion

In this paper, we introduced the novel problem of cascade prediction involving the prediction of both future nodes and timing. Our model *DeepDiffuse* was designed to focus on specific nodes for prediction by learning to jump to any part of a cascade as necessary. We evaluate our model in some real world datasets against state-of-the-art methods proposed for similar problems. Our model demonstrates outstanding results in every experimental scenario. One of the future research directions involves estimating how effective *DeepDiffuse* can be in predicting the future growth of cascades or identifying viral cascades. Another direction can be incorporating the content of the cascade (meta-features) into the prediction problem.

Acknowledgements

This paper is based on work partially supported by US NSF grants IIS-1750407 (CAREER), IIS-1633363, and DGE-1545362, by the NEH (HG-229283-15), by the Army Research Laboratory under grant W911NF-17-1-0021, ORNL and a Facebook faculty gift.

References

[1] H.-W. Shen, D. Wang, C. Song, and A.-L. Barabási, "Modeling and predicting popularity dynamics via reinforced poisson processes," in *AAAI*, 2014, pp. 291–297.
[2] L. Weng, F. Menczer, and Y.-Y. Ahn, "Predicting successful memes using network and community structure," in *ICWSM*, 2014, pp. 535–544.
[3] C. Bauckhage, F. Hadiji, and K. Kersting, "How viral are viral videos," in *ICWSM*, 2015, pp. 22–30.
[4] Y. Matsubara, Y. Sakurai, B. A. Prakash, L. Li, and C. Faloutsos, "Rise and fall patterns of information diffusion: model and implications," in *KDD*, 2012, pp. 6–14.
[5] J. Cheng, L. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec, "Can cascades be predicted?" in *WWW*, 2014, pp. 925–936.
[6] L. Yu, P. Cui, F. Wang, C. Song, and S. Yang, "From micro to macro: Uncovering and predicting information cascading process with behavioral dynamics," in *ICDM*, 2015, pp. 559–568.
[7] K. Subbian, B. A. Prakash, and L. Adamic, "Detecting large reshare cascades in social networks," in *WWW*, 2017, pp. 597–605.
[8] C. Li, J. Ma, X. Guo, and Q. Mei, "Deepcas: An end-to-end predictor of information cascades," in *WWW*, 2017, pp. 577–586.
[9] J. Wang, V. Zheng, Z. Liu, and K. Chang, "Topological recurrent neural network for diffusion prediction," in *ICDM*, 2017, pp. 475–484.
[10] Q. Zhao, M. Erdogdu, H. He, A. Rajaraman, and J. Leskovec, "Seismic: A self-exciting point process model for predicting tweet popularity," in *KDD*, 2015, pp. 1513–1522.
[11] K. Lerman and R. Ghosh, "Information contagion: An empirical study of the spread of news on digg and twitter social networks." in *ICWSM*, vol. 10, 2010, pp. 90–97.
[12] M. Gomez Rodriguez, J. Leskovec, and A. Krause, "Inferring networks of diffusion and influence," in *KDD*, 2010, pp. 1019–1028.
[13] C. L. Barrett, K. R. Bisset *et al.*, "Episimdemics: an efficient algorithm for simulating the spread of infectious disease over large realistic social networks," in *ACM/IEEE SC*, 2008, pp. 1–12.
[14] S. Sundareisan, J. Vreeken, and B. A. Prakash, "Hidden hazards: Finding missing nodes in large graph epidemics," in *SDM*, 2015, pp. 415–423.
[15] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song, "Recurrent marked temporal point processes: Embedding event history to vector," in *KDD*, 2016, pp. 1555–1564.
[16] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.
[17] V. Mnih, N. Heess, A. Graves *et al.*, "Recurrent models of visual attention," in *NIPS*, 2014, pp. 2204–2212.
[18] N. O. Hodas and K. Lerman, "The simple rules of social contagion," *Scientific reports*, vol. 4, p. 4343, 2014.
[19] J. Leskovec, L. Backstrom, and J. Kleinberg, "Meme-tracking and the dynamics of the news cycle," in *KDD*, 2009, pp. 497–506.
[20] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *KDD*, 2003, pp. 137–146.
[21] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, "Maximizing social influence in nearly optimal time," in *SODA*, 2014, pp. 946–957.
[22] S. Myers and J. Leskovec, "On the convexity of latent social network inference," in *NIPS*, 2010, pp. 1741–1749.
[23] S. Bourigault, S. Lamprier, and P. Gallinari, "Representation learning for information diffusion through social networks: An embedded cascade model," in *WSDM*, 2016, pp. 573–582.
[24] M. Jenders, G. Kasneci, and F. Naumann, "Analyzing and predicting viral tweets," in *WWW*, 2013, pp. 657–664.
[25] P. Cui, S. Jin, L. Yu *et al.*, "Cascading outbreak prediction in networks: a data-driven approach," in *KDD*, 2013, pp. 901–909.
[26] O. Tsur and A. Rappoport, "What's in a hashtag?: Content based prediction of the spread of ideas in microblogging communities," in *WSDM*, 2012, pp. 643–652.
[27] D. J. Daley and D. Vere-Jones, *An Introduction to the Theory of Point Processes*. Springer Science & Business Media, 2007.
[28] N. Du, L. Song, M. Yuan, and A. J. Smola, "Learning networks of heterogeneous influence," in *NIPS*, 2012, pp. 2780–2788.
[29] Y. Tang, Y. Shi, and X. Xiao, "Influence maximization in near-linear time: A martingale approach," in *SIGMOD*, 2015, pp. 1539–1554.
[30] E. Bacry, A. Iuga, M. Lasnier, and C.-A. Lehalle, "Market impacts and the life cycle of investors orders," *Market Microstructure and Liquidity*, vol. 1, no. 02, 2015.
[31] A. Ferraz Costa, Y. Yamaguchi *et al.*, "RSC: Mining and modeling temporal activity in social media," in *KDD*, 2015, pp. 269–278.
[32] C. Gulcehre, F. Dutil, A. Trischler, and Y. Bengio, "Plan, attend, generate: Planning for sequence-to-sequence models," in *NIPS*, 2017, pp. 5480–5489.
[33] H. Larochelle and G. E. Hinton, "Learning to combine foveal glimpses with a third-order boltzmann machine," in *NIPS*, 2010, pp. 1243–1251.
[34] M. Denil, L. Bazzani, H. Larochelle, and N. de Freitas, "Learning where to attend with deep architectures for image tracking," *Neural Computation*, pp. 2151–2184, 2012.