MUCA: A Multi-Content-Aware Recurrent Neural Network for Cascade Prediction

Zijie Huang
Shanghai Jiao Tong University
515021910157
hzj_jingjing@sjtu.edu.cn

Chunwei Wang
Shanghai Jiao Tong University
515021910183
weiwei0224@sjtu.edu.cn

Yihan Zeng
Shanghai Jiao Tong University
515021910542
zengyihan@sjtu.edu.cn

Adviser: Luoyi Fu
Shanghai Jiao Tong University
fu-ly@cs.sjtu.edu.cn

Adviser: Xinbing Wang
Shanghai Jiao Tong University
wang-xb@cs.sjtu.edu.cn

ABSTRACT

Information cascades, the representation of content reshares in various networks, are considered a major factor in predicting the future diffusion of a certain content. With the proliferation of deeplearning techniques, many learning-based methods have been proposed to predict the future size of a cascade, that is, the number of newly affected users in the future timestamps. However, those methods only considered single content diffusion without taking into account the impact of those simultaneously diffused contents. In this paper, we predict the future size of a certain cascade based on both the cascade itself and cascades of other interrelated contents that are diffused in the same network. As a result, it is critical to represent cascade graphs of other contents in each time step so as to remind the current cascade graph the existence of other contents' diffusion. Therefore, we proposed a novel multi-contents-aware recurrent neural network, namely MUCA-LSTM, to model the existence of other contents. We customized MUCA-LSTM for the cascade size prediction problem which significantly improves the performance of cascade size prediction compared with existing baselines such as feature-based methods and node embedding methods.

KEYWORDS

Cascade Prediction, Recurrent Neural Network, Information diffusion

1 INTRODUCTION

Information diffusion is a common phenomenon that happens every second in various networks which reveals how people discover, consume information online [5]. In most cases, a piece of information such as a photo, a twitter can get reshared multiple times via the network among users and a cascade of resharing can develop, potentially reaching a large number of people. Such cascades have been identified in plenty of applications ranging from product marketing, rumor spread, crowdsourcing, etc. If the size of a cascade can be predicted, that is the number of affected users in the future, the potential influence of a certain content can be captured so that one can make better decision about it. For example, a clothing manufacturer can gain more profits by producing clothing with higher popularity; figuring out the potential influence of a rumor helps the administrators to prevent social instability in advance; in academic network, the potential citation number of a paper helps researchers to identify its significance.

Early work usually assumes that diffusion model is given, such as independent cascade (IC)[19] and linear threshold (LT) [10]. Based on whether nodes or edges attributes, an activation probability function is established between two nodes for predicting the spread of a content. Those IC and LT models also enable an important research direction of influence maximization [18]. Nowadays, researchers try to learn the diffusion model directly from those given cascade graphs, i.e. the activated node sequence with activation timestamps. Those methods largely rely on explicitly experience-based, manual-extracted features to model the activation probability of a node, such as network structure, nodes' social roles , edge type between users, diffusion content, etc. Although these methods have shown significant improvements in diffusion prediction performance, they cannot be generalized to more applications for the performance largely depends on how good those selected features are. This requires much manual effort and extensive domain knowledge. With the recent proliferation of neural networks, another branch of work starts to utilize deep learning, so as to avoid explicit feature extraction for diffusion modeling. For example, Embedded-IC [3] considers each inactive node to be activated by the active nodes where each node is modeled as one of the following two roles: an active node serves as a"sender", and an inactive node serves as a "receiver". For each role, it learns a special vector as a node's embedding and then models an activation function based on the similarity between an inactive node's receiver embedding vector and the active nodes' sender embedding vectors. Another example is DeepCas [16], which aims at predicting the future cascade size. It models the cascade at each time step with an induced subgraph over the active nodes. Then, it decomposes the subgraph into some random walk paths, and uses Gated Recurrent Unit (GRU) to learn an embedding vector of the subgraph. Based on this subgraph embedding vector, it predicts the cascade size in the future.

Despite the proliferation of deeplearning methods to deal with cascade prediction, we find that the impact of other simultaneously diffused contents are often underexplored in those models. In reality, the diffusion of one content in a network is not independent with others; the fact is that there are usually multiple contents diffused together in a network during the same period. For example, consider the purchase of Apple Watch and iPhone8, it is vital to recognize that Apple Watch generally needs an iPhone to be usable, and iPhone's user experience can be greatly enhanced by a pairing Apple Watch. In other words, they can exert strong influence on each

other. Moreover, the relationship between different propagating contents is more general than pure competition. Therefore, how to incorporate cascades of other contents over the network into the predicted cascade lies in the heart of our paper. Without loss of generalization, we focus on two content diffusion process and leave the other settings as our future work.

We present a simple example to further explain our intuition as follows. Consider a network graph as shown in Fig. 1 where two contents I_1 and I_2 are propagated. We represent each cascade as a set of sampled cascade sequences and incorporate cascades of another content I_2 into each sampled cascade sequences of the targeted content I_1. Assume $A \rightarrow B \rightarrow C \rightarrow D$ is the cascade sequence c_i^1 of I_1 that we are currently dealing with, and I_2 has k sampled sequences where the 1^{st} sequence c_1^2 is $E \rightarrow B \rightarrow G$ and the k^{th} sequence c_k^2 is $D \rightarrow A \rightarrow G \rightarrow C$. We would like to incorporate the diffusion of I_2 into the targeted cascade sequence c_i^1 . Particularly, from Fig. 1(a) we know that before A is activated for I_1, $\{E\}$ has already been activated by I_2 in c_1^2 and $\{D\}$ is activated by I_2 in c_k^2. So E has the potential chance to influence its neighbor $\{B\}$ in G about whether B would be activated by I_1 and the same is with D to influence A, C. Therefore we draw an arrow from E to B and D to A, C, to denote the potential influence attempts. As a consequence, we form the multi-content-aware diffusion graph (MUCA-diffusion graph) before timestamp t(A,I_1) as shown in Fig 1(a), where t(A,I_1) is the time when A is activated by I_1. Similarly, before B is activated by I_1, both $\{B, E\}$ in c_1^2 and $\{D, A\}$ in c_k^2 have the potential influence towards whether B would be activated by I_1. As a result, we get the MUCA-diffusion graph in Fig 1(b). Such MUCA-diffusion graphs are helpful in capturing the influence of simultaneously diffused content, e.g. on the one hand, a piece of information is more likely to be propagated from a user to her friends; on the other hand, a user's willingness of buying an iPhone8 may be influenced by the popularity of Apple Watch among her friends. By utilizing MUCA-diffusion graphs, we can make better prediction of the concerned content. However, existing deeplearning methods for diffusion modelling do not take the influence of simultaneously diffused content into consideration, instead, they only focus on the cascade of the targeted content.

In this paper, we demonstrate how to wisely incorporate cascade graph of another content into the predicted cascade (more specifically, into a set of sampled cascade sequences of the concerned cascade) and utilize deeplearning techniques to predict the future size of the concerted cascade. Generally, we first sample several cascade sequences to represent each cascade graph. Then, we utilize all the sequences of content I_2 to construct MUCA-diffusion graphs in different timestamps so as to incorporate I_2's influence into each sampled cascade sequence of I_1. Finally, we integrate the learning results of every incorporated cascade sequence of I_1 to get a final result. The main challenge lies in how to utilize the sampled sequences of content I_2 to form MUCA-diffusion graph and then incorporate it into each sampled sequence of I_1.

Our main contributions are summarized as follows:

- We proposed a new diffusion data model, namely multi-content-aware diffusion graph (MUCA-diffusion graph) to better explore the interaction of two simultaneously diffused contents in the same network.
- We designed a novel MUCA-LSTM model that is full aware of the influence of the simultaneously diffused content, so as to better predict the growth of the concerned cascade in the future.
- We conduct extensive experiments on real and synthetic datasets and verify the effectiveness of our methods.

The rest of the paper is organized as follows. Sec. 2 presents a formal definition of the cascade size prediction problem. Sec. 3 addresses the framework of our proposed MUCA method and further explained it into three substeps: cascade sequence sampling, MUCA-diffusion graph construction and finally the MUCA-LSTM. Sec. 4 evaluates our proposed methods and presents the evaluation results compared with other state-of-art methodologies. Sec. 5 presents the visualization of our result and the conclusion is drawn in Sec. 7.

2 PROBLEM FORMULATION

In this section, we study the problem of cascade size prediction that aims to predict the number of newly affected nodes during a future time period in a cascade by analyzing the available cascade graphs. To begin with, we first formally define the problem and then explain our methodology. In this paper, we consider two-content scenario where there are two simultaneously diffused contents in a network and denotes these two contents as I_1, I_2 respectively.

Definition 2.1 (Network Graph). A network graph is a directed graph $G = (V, E)$, where V is the set of nodes in the network and $E \subset V \times V$ is the set of edges among nodes.

A node $v \in V$ represents a user (e.g. a researcher in the academic network, a blogger in Twitter). A weighted edge from a source node to a target node indicates a relationship tie (e.g. retweeting or citation) between them. The reason the network graph is directed is that, in reality, the influence from the source to target nodes is asymmetric to the influence from the target to source nodes. For example, a celebrity in Twitter may have millions of followers while she may only follows a limited number of users. An undirected network graph is an approximation of the directed form.

Definition 2.2 (Cascade Graph). A cascade graph of content m is an ordered sequence of tuples that consists of a timestamp t_i and a node set V_i^m including newly activated nodes at timestamp t_i. Those time-relevant tuples records all the activated nodes along with their activation time from the beginning of the cascade until time t. It is denoted as $g_t^m = \{(V_1^m, t_1), (V_2^m, t_2) \cdots (V_n^m, t_n)\}$ where V_i^m is the set of nodes that are activated by information I_m at timestamp t_i. Each node v appears only in one node set V_i and each t_i is a timestamp such that $t_i < t_{i+1}$. We denote the set of activated nodes until time t as $V_t^m = \bigcup_i V_i^m$.

In reality, we may only observe the relative orders of those timestamps t_i without knowing the exact time information (e.g., we know who cited a paper in t_i but not know exactly when and where she found the paper).

Definition 2.3 (Cascade Graph Tuple). A cascade graph tuple is a set of two cascade graphs that represents two simultaneously diffused contents along with their integrated network graph, denoted as $g_t = \{g_t^1, g_t^2, G^*(V^*, E^*)\}$. $V^* = \bigcup_m V_t^m$ is the union

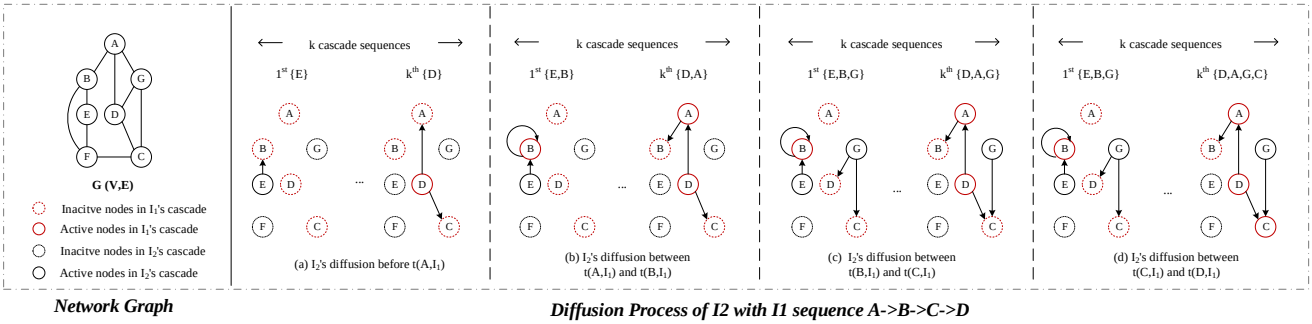

Figure 1: Illustration of MUCA-diffusion graph. At each timestamp t, we construct the corresponding MUCA-diffusion graph which depicts the influence of another content's current diffusion (I_2) on the targeted content I_1. In each MUCA-diffusion graph, solid circles and dotted circles are active and inactive nodes of I_2 respectively. Red circles are nodes in an I_1 cascade sequence. An arrow is a possible influence attempt of I_2 to nodes of the targeted cascade sequence in I_1.

of nodes in these two cascade graphs and $E^* = \{(u,v)|(u,v) \in G(V,E)$ and $u,v \in V^*\}$ is the relationship tie among those nodes.

As problem input, we have a network graph $G = (V,E)$ and a set of training cascade graph tuples. The cascade size prediction problem is then defined as follows.

Definition 2.4 (Cascade Size Prediction Problem). Give a network graph $G = (V,E)$ and a set of training cascade graph tuples, the cascade size prediction problem is to find a matching function f to output the increased size $\Delta V^m = |V^m_{t+\Delta t}| - |V^m_t|(m = 1,2)$ of two cascade graphs in a tuple after time period Δt (i.e. the number of newly affected nodes in a cascade). That is, $f : (g_t, \Delta t) \rightarrow (\Delta V^1, \Delta V^2)$.

Table 1 summarizes the commonly used symbols.

Table 1: Symbols and Descriptions

Symbol	Description
g_t	a cascade graph tuple until time t
g^m_t	a cascade graph of content m until time t
h_i	output embedding from MUCA-LSTM for the i^{th} node in a cascade sequence
x_v	embedding vector for node v
c^m_j	the j^{th} sampled cascade sequence from cascade graph of content m

3 MUCA: A MULTI-CONTENT-AWARE LSTM NETWORK

3.1 Framework of MUCA

We proposed a multi-content-aware LSTM network framework that takes the influence of simultaneously diffused content into consideration as shown in Fig. 2. The input is a cascade graph tuple consists of two cascade graphs along with their integrated network graph and the output is the increased size of these two cascades $(\Delta V^1, \Delta V^2)$. The framework first samples k cascade sequences to represent each of the two cascade graphs and then iteratively feeds one cascade sequence of I_1 along with all the k sequences of I_2 into a $MUCA-LSTM$ network. Finally, by utilizing pooling techniques we get an aggregated single vector to represent I_1 and feeds it to a MLP network to predict the final size increase of I_1. The same procedure is taken as well to obtain the size increase of I_2.

3.2 Cascade Graph Representation–Cascade Sequence Sampling

Given a cascade graph g^m_t, we would like to learn an embedding vector to fully represent the development of the cascade diffusion as the input of MUCA-LSTM. Therefore, we first represent the cascade graph as a set of cascade sequences (a set of nodes sequences).

Definition 3.1 (Cascade Sequence). A cascade sequence of content m is an ordered time-relevant sequence of several node-time tuples $c^m_i = \{(v_1,t_1),(v_2,t_2)\cdots(v_n,t_n)\}$, where each node v is an distinct node in a cascade graph and each timestamp t_j satisfies $t_i < t_{i+1}$. The subscript i indicates it is the i^{th} sampled cascade sequence of content m.

In order to preserve the structural information of the cascade graph, we adopt the similar approach proposed in DeepCas [16] and Deepwalk [20], i.e. perform random walk over a cascade graph g^m_t. However, the major difference in our random walk is that, nodes in a cascade graph are clustered by their affected years. Supposed a cascade sequence is given by $v_1 \rightarrow v_2 \rightarrow \cdots v_n$, it must always satisfy:

$$t(v_i) < t(v_j), if \quad i < j$$

where $t(v_i)$ returns the affected year of v_i. It means that the early nodes in a cascade sequence is affected in early years compared with nodes behind them. Therefore, after we first extract the cascade graph edge set $E_{cascade} = \{(u,v)|(u,v) \in G(V,E)$ and $u,v \in V^m_t\}$, we filter those edge $(u,v) \in E_{cascade}$ where the affected year of source node equals to or later than that of the target node, i.e. $t(u) \geq t(v)$. Then, as shown in Fig. 3, we perform a random walk on the cascade graph and its filtered edge set $E_{cascade}$.

The random walk starts from the starting state S, which is followed by the state N, where the walker transits to a neighbor of the current node. With probability 1-p_j, it goes on walking to the neighbor. With a jumping probability p_j, it jumps to an arbitrary node in the cascade graph that is affected later than itself, leading

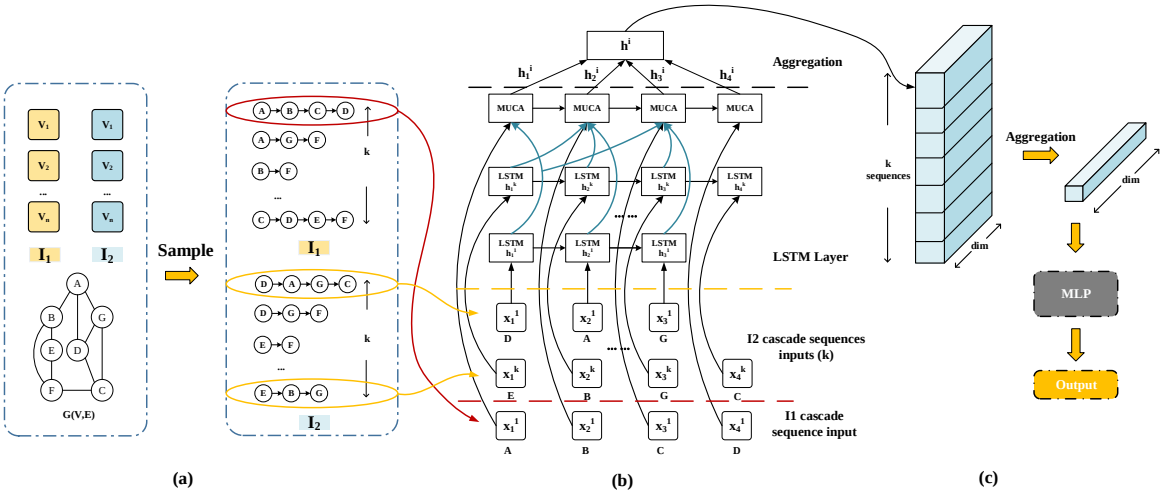

Figure 2: Illustration of MUCA-LSTM Framework.

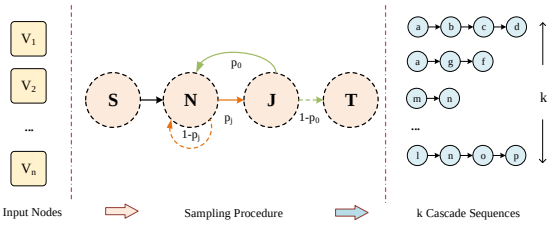

Figure 3: Illustration of cascade sequence sampling

the walker to the jump state J. With continuation probability p_0, it walks to a neighbor of the current node, thus going back to state N. With probability 1-p_0, it goes to the terminal state T, terminating the entire random walk process.

When the walker is in state N, it follows a transition probability $p(u \in Nbr(v)|v)$ to go to one of its outgoing neighbors $u \in Nbr(v)$, where $Nbr(v)$ is the neighbor set of the current node v. The transition probability can be calculated in multiple ways such as the degree of u, the edge weight $weight(v, u)$ between them, etc.

When the walker is in state J, it follows a transition probability to jump to an arbitrary node who are affected later than itself. The transition probability $p(u|v)$ can also be calculated by the degree of u and other proper ways.

3.3 MUCA-Diffusion Graph

After we represent each cascade graph as a set of cascade sequences, the main challenge lies in how to embed each node in a sequence so that the embedding vector of the concerned cascade graph, which is obtained by pooling all the node embeddings in each cascade sequence, is fully aware of the influence of another simultaneously diffused content in the network. To address this challenge, we introduce a new data model, namely MUCA-diffusion graph, to explore the interaction of two simultaneously diffused cascades. Then in Sec. 3.4 we design a novel MUCA-LSTM model to learn the cascade graph embedding by utilizing MUCA-diffusion graph.

Finally, we feed the cascade graph embedding vector as the input of a BP network to predict the future size increase of our concerned cascade.

When predicting the future size increase of content I_1, the MUCA-diffusion graph is useful in representing the potential influence from another simultaneously diffused content I_2 to I_1. Since we represent each cascade graph as a set of cascade sequences, we would like to construct MUCA-diffusion graph for each sampled sequence c_i^1 of content I_1.

Consider a set of k cascade sequences of content I_2, $C^2 = \{c_1^2, c_2^2 \cdots c_k^2\}$ and a sampled sequence of content I_1, $c_i^1 = \{(v_1, t_1), (v_2, t_2) \cdots (v_n, t_n)\}$, we denote $N_{1:t_i-1}$ as the set of nodes in any sampled sequences of content I_2 that are activated before time t_i. For example in Fig. 1, $N_{1:t_A-1} = \{E, D\}$ and $N_{1:t_B-1} = \{E, B, D, A\}$ where t_A, t_B refers to the activation time of node A, B by content I_1 respectively. We also use $O_{1:t_i-1}$ to denote the set of nodes in c_i^1 who are activated by I_1 before t_i. Nodes in $N_{1:t_i-1}$ have the potential influence on those inactivated nodes till $t_i - 1$ in I_1 sequences and we would like to demonstrate how they can exert such potential influence to those inactivated nodes in I_1 sequences. Intuitively, since a user in a network would be more easily influenced by her friends, we utilize those edge information in the network graph $G(V, E)$ to construct MUCA-diffsuion graphs that reveal the interaction between I_1 and I_2.

Definition 3.2 (MUCA-Diffusion Graph). Given a network graph $G(V, E)$, a MUCA-diffusion graph $G_t^{c_i^1}(V_t^{c_i^1}, E_t^{c_i^1})$ for cascade sequence c_i^1 of content I_1 until time t is a directed graph, where $V_t^{c_i^1}$ is the set of nodes in the concerned cascade sequence c_i^1 of I_1 and those appears in any of the k sampled sequences of I_2. $E_t^{c_i^1} = \{(u_i, v)|(u_i, v) \in V, u_i \in N_{1:t-1}, v \in N_{c_i^1} \backslash O_{1:i-1}\}$ is the set of edges representing the potential influence from I_2 to I_1 where $N_{c_i^1}$ is the node set of c_i^1.

Every edge $(u_i, v) \in E_t^{c_i^1}$ represents a potential influence from an active node in I_2 to an inactive node in I_1. Therefore, we call u_i a potential influencer of v and we denote the potential influencer set of v until time t as $PI_{v,t} = \{v_i | (v_i, v) \in E_t^{c_i^1}\}$. As we can see, a MUCA-diffusion graph fully characterizes the influence exerted from another content, thus it is helpful in learning the cascade graph embedding. Fig. 1 represents the MUCA-diffusion graphs at four different timestamps and we remark that the MUCA-diffusion graph at any given timestamp is unique.

Moreover, the MUCA-diffusion graph is monotonically growing over time and the graph at timestamp t is always a subgraph of that at timestamp $t-1$. This indicates that the node embedding with a sequence can be learned recurrently from earlier embedding vectors, which naturally yields a LSTM model to implement our prediction. However, existing LSTM model cannot incorporate the MUCA-diffusion graph so we design a novel MUCA-LSTM described in the following section.

3.4 MUCA-LSTM

Once we have sampled k sequences to represent each of the two cascade graphs, we design a novel MUCA-LSTM model to embed every sequence of content I_1 with corresponding MUCA-diffusion graphs so as to get an embedding vector for each sequence. Then we use pooling techniques to aggregate those sequence embedding vectors to get an embedding vector for the whole cascade graph. This is because we can view each cascade sequence as a sentence, every node in a sequence as a unique word and a cascade graph as an article. Therefore, after we have embedded sentences we need to aggregate them to represent the article just like in natural language processing (NLP). The difference is that the content of sentences in an article has already been written, however in our cases, we need to first figure out what are the contents of these sentences, that is, to embedding structural information and the influence of another diffused content into each sequence. The structural information is solved by using random walk, while the influence embedding remains unsolved. Therefore, we utilize the aforementioned MUCA-diffusion graphs and design a novel MUCA-LSTM model to embed the influence information. Finally we feed embedding vector of the cascade graph as input of a MLP network to obtain the final size increase result.

Node Embedding. Each node in a sequence is represented by a unique embedding vector $x \in \mathbb{R}^d$ where d is the feature dimension of the vector. A simple example is to use the one-hot vector to represent each node's identity.

MUCA-LSTM Sequence Embedding. In order to incorporate MUCA-diffusion graphs into the sequence embedding vector, we design a novel MUCA-LSTM model which contains $k+1$ input layers and $k+1$ LSTM layers as shown in Fig. 2, to iteratively calculate embedding vectors for each cascade sequence.

Assume we are embedding the i^{th} cascade sequence c_i^1 in I_1. Firstly, for each of the k sampled sequences of I_2, we apply the standard Long-Short-Term-Memory (LSTM), an improved version of traditional recurrent neural network (RNN) which is known to be effective for modeling sequences, to output the hidden states in each sequence . When applying LSTM recursively to a sequence from left to right, the sequence representation will be more and more enriched by information from later nodes in this sequence, with the gating mechanism deciding the amount of new information to be added and the amount of history to be preserved, which simulates the process of information flow during a diffusion. To be more specific, we denote step j the j^{th} node in each sequence, for each step i, we input the embedding vector of j^{th} node and the previous hidden state h_j^i, so as to compute the hidden state at the current step as shown in Eqn. (1).

$$h_j^i = LSTM(x_j^i, h_{j-1}^i) \tag{1}$$

Then we design a new LSTM layer, namely MUCA-LSTM layer to incorporate the MUCA-diffusion graphs into targeted embedding sequence c_i^1 by revising standard LSTM as the following running example.

Example 3.3. Consider a running example as shown in Fig. 1 where the targeted embedding sequence is $c_i^1 = \{(A, t_1), (B, t_2), (C, t_3), (D, t_4)\}$. Assume for each cascade graph we sample 2 cascade sequences (i.e. k=2) and for I_2 they are $c_1^2 = \{(E, t_0), (B, t_1), (G, t_2)\}$, $c_2^2 = \{(D, t_0), (A, t_1), (G, t_2), (C, t_3)\}$. At timestamp t_1, A is activated by I_1 and we get the MUCA-diffusion graph as shown in Fig. 1(a) . We can know that the potential influencers of A are $PI_{A,t_1} = \{D^2\}$ where the superscript 2 denotes the sequence number where D belongs to. So we take both $A's$ embedding vector x_A and the hidden state output h_{D^2} of D in c_2^2 as input to get the hidden state output of A, i.e. h_A. Next at timestamp t_2, B is activated by I_2 and the MUCA-diffusion graph is shown in Fig. 1(b) . The potential influencers of B are $PI_{A,t_2} = \{E^1, B^1, A^2\}$. Therefore, we first use pooling technique to aggregate $h_{E^1}, h_{B^1}, h_{A^2}$ into a single vector $h_{tmp} = \phi(h_{E^1}, h_{B^1}, h_{A^2})$ and then take both h_{tmp} and $B's$ embedding vector x_B as input to calculate the hidden state output of B, i.e. h_B. The same procedure is operated on the remaining nodes in the targeted embedding cascade c_i^1.

Generally speaking, our MUCA-LSTM differs from the standard LSTM in the input of each cell where we input not only the node embedding vector but also its influencers' hidden states. To incorporate MUCA-diffusion graphs into the sequence embedding vector, we adopt a pooling function to aggregate the influencers' hidden state h_i^j as shown in Eqn. (2), where the aggregation function $\phi(\cdot)$ can be a simple pooling or some more sophisticated attention mechanism. In this paper, we adopt the mean pooling method.

$$h_{tmp}^j = \phi(\{h_v | v \in PI_{v_j,t_j}\}) \tag{2}$$

Our MUCA-LSTM also has a memory cell with three functional gates along with a cell state, similar to the standard LSTM. We set the dimension of the cell state vector as D and illustrate the structure of MUCA-LSTM as follows.

- **Input Gate:** Assume $W_i \in \mathbb{R}^{D \times d}$, $b_i \in \mathbb{R}^D$, $Z_i \in \mathbb{R}^{D \times D}$ as input gate parameters, the input gate activation vector $i_j \in \mathbb{R}^D$ in the j^{th} step is computed as

$$\mathbf{i_j} = W_i \mathbf{x_j} + Z_i \mathbf{h_{tmp}} + b_i \tag{3}$$

- **Forget Gate:** Assume $W_f \in \mathbb{R}^{D \times d}$, $b_f \in \mathbb{R}^D$, $Z_f \in \mathbb{R}^{D \times D}$ as forget gate parameters, the forget gate activation vector $f_j \in \mathbb{R}^D$ in the j^{th} step is computed as

$$\mathbf{f_j} = W_f \mathbf{x_j} + Z_f \mathbf{h_{tmp}} + b_f \tag{4}$$

- **Cell State:** Assume $W_c \in \mathbb{R}^{D\times d}$, $b_c \in \mathbb{R}^D$, $Z_c \in \mathbb{R}^{D\times D}$ as cell state parameters, we first aggregate cell states from the node's influencers' as follows where $\circ$ denotes the element-wise multiplication.

$$c_j^{'} = \phi(\{c_v | v \in PI_{v_j,t_j}\}) \tag{5}$$

Then the cell activation vector $c_j \in \mathbb{R}^D$ in the j^{th} step is computed as

$$\begin{aligned} \hat{c_j} &= tanh(W_c \mathbf{x_j} + Z_c \mathbf{h_{tmp}} + b_c) \\ c_j &= i_j \circ \hat{c_j} + \mathbf{f}_j \circ c_j^{'} \end{aligned} \tag{6}$$

- **Output Gate:** Assume $W_o \in \mathbb{R}^{D\times d}$, $b_o \in \mathbb{R}^D$, $Z_o \in \mathbb{R}^{D\times D}$ as forget gate parameters, the forget gate activation vector $o_j \in \mathbb{R}^D$ in the j^{th} step is computed as

$$o_j = \sigma(W_o \mathbf{x_j} + Z_o \mathbf{h_{tmp}} + b_o) \tag{7}$$

Finally the output hidden state vector $h_j \in \mathbb{R}^D$ is given by

$$h_j = o_j \circ tanh(c_j) \tag{8}$$

Cascade Graph Embedding and Learning. After we get the output hidden state vector h_j for each step in every embedding sequence of I_1, we aggregate these hidden states vector to obtain the final embedding vector for the cascade graph as follows, where h_i^j is j^{th} hidden state vector in i^{th} cascade sequence of I_1.

$$h_{output} = \phi(\sum_i \sum_j h_i^j) \tag{9}$$

Finally , we feed the cascade graph embedding vector h_{output} to a MLP network to obtain the predicted size increase of a cascade graph, i.e. $f(g_t, \delta t) = MLP(h_{output})$ where MLP stands for multi-layer perception.

4 EXPERIMENTS AND RESULTS

To validate our proposal, we adopt real world datasets and present comprehensive experiments to evaluate the performance of MUCA-LSTM. The experiments are conducted on a machine with Intel Core i7 server, 2.70GHZ processor and 8GB RAM.

4.1 Experiment Setup

Datasets. We conduct the experiment on academic networks to predict the cascades of scientific papers. The dataset is collected from Acemap[1], which is an academic website for academic visualization. We first construct the network graph $\mathcal{G} = (V, E)$ within the domain of computer science bacause Acemap has classified the paper into three-level topic hierarchy structures from L_0 to L_2 where computer science domain belongs to L_0. The node set V are authors who have been active in the computer science domain during our chosen experiment period from 2000 to 2007. To identify active authors, we only select authors who has more than 10 citation in computer science domain between 2000 and 2007 and filter those who own less than 10 citations. An directed edge in the edge set E represents the relationship tie between two nodes. We draw an edge from source node A to target node B if author B has cited A's papers for at least 5 times thus filtering those edge with low relationship tie.

[1]http://acemap.sjtu.edu.cn

As for cascade contents, since the network graph needs to be constructed before a cascade begin, we select computer science papers published in year 2008 and construct a cascade for each of those papers. To be more specific, if a paper is cited by a certain author, we include her as an activated node in the cascade and record her activation time, i.e. the citation year of this paper. Therefore, the cascade format is a set of tuples $c_i^m = \{(V_1, t_1), (V_2, t_2), ..., (V_n, t_n)\}$ where $t_i, i = 1, 2, ..., n$ denotes a specific year and V_i denotes the set of authors who cited this paper in year t_i. We construct the cascade from 2008 to 2013 and use the number of activated authors in 2014 and 2015 as labels.

After constructing cascades, we manually select several two-paper tuples to model two simultaneously diffused contents in the network. According to our design intuition, the two papers in one tuple are supposed to be two highly related papers, otherwise the mutual influence between these two papers affects little about the cascade size prediction problem. Hence, we first classify the paper into several clusters and then match every two papers by the criterion that these two papers must be published in the same year (to guarantee they stay in the same diffusion period) and belong to the same clusters so that they may pretend to be in high relativity. After the above procedures, we finally extracted 5102 matched paper pairs for training, 821 pairs for validation and 866 for test respectively. The statistic of the dataset are illustrated in Tab. 2. Note that the scale growth is measured by $log_2(\Delta s + 1)$ where Δs denotes the increased active nodes per year.

Table 2: The statistic of the dataset

	Set	Acemap
# nodes in $\mathcal{G}$	All	151384
# edges in $\mathcal{G}$	All	2481312
cascades	train	1554
	val	851
	test	845
Avg. nodes per cascade	train	73
	val	75
	test	74
Avg. edges per cascade	train	68
	val	73
	test	72
Avg. scaled growth	train	3.01
	val	3.03
	test	3.02
matched paper	train	5102
	val	821
	test	866

Clustering. To classify the papers into several clusters, we first measure the similarity of two papers by two features, i.e., topic similarity and cascade similarity. Here, we apply the Microsoft Academic Graph, which provides topic information of each paper and the topic hierarchy structure. If the topic hierarchy structure of two papers as shown in Fig. 4 is similar, we consider they tend to

have high relativity in topics. Hence, we define the topic similarity of two papers in Eqn. (10) where w_i is the weight coefficient for each level and n_i is the num of same topic that two paper belong to. As for cascade similarity, it plays a significant role and we measure it by the following three factors.

- Same node num $|V^{'}| = |V_A \cap V_B|$.
- Average node degree of "intersectant graph" $G^{'} = (V^{'}, E^{'})$ where $V^{'} = V_A \cap V_B$ is the same activated author set of two cascade graghs and $E^{'}$ is the set of the edge between these same authors. The average node degree is defined as $\frac{V^{'}}{E^{'}}$.
- Activated tendency: we consider that if the tendency of two papers that paid attention are similar, they may have some interactions during the diffusion. We define the cite tendency vector as $C = [c_{t_1}, c_{t_2}, ..., c_{t_n}]$ where c_{t_i} is the citation percent in year t_i among total citations in the whole time period and we apply cosine distance of two cite tendency vector to measure the similarity.

We combine these aforementioned factors and use normalization to define cascade similarity between paper A and B as denoted in Eqn. (11) where f denotes the cosine distance of two vectors. Finally, we set the distance between two papers in Eqn. (12) where λ is the coefficient.

$$Sim_{topic} = \sum_{i=0}^{2} n_i * w_i \tag{10}$$

$$Sim_{cascade} = \frac{|V_A \cap V_B|}{|V_A \cup V_B|} + \frac{V^{'}/E^{'}}{\frac{1}{2}(V_A/E_A + V_A/E_A)} + f(C_A, C_B) \tag{11}$$

$$Dis(A, B) = 1/(Sim_{topic} + \lambda Sim_{cascade}) \tag{12}$$

With distance definition we can obtain the distance metrics M and we apply DBSCAN, a kind of density-based clustering method, for clustering as illustrated in Alg. 1.

Figure 4: The effect of number of dimension of hidden state vectors

Baselines. We select three state-of-art and representative baselines for comparison with our model.

- DeepCas [16] considers only one item propagating in the network to predict the future size of the cascades. It applies random walk process to sample multiple paths to represent the diffusion cascade and learn in an end-to-end manner using RNN. We choose the hyperparameters as 200 walks of length 10 for each cascade, the same setting as in [16].
- Topo-LSTM [29] also considers one item in the network to predict the next activated node. It holds the view that a cascade is not merely a sequence of nodes ordered by their activation time stamps; instead, it has a richer structure indicating the diffusion process over the data graph. Hence, it revise the structure of RNN to represent diffusion topology and the input data is the activation sequence of the nodes. To better compare the with our model, we did two main revisions for Topo-LSTM without changing its core idea. We change the input data as our random walk result and add a MLP to predict the future cascade growth size.
- Feature-based method: we select several structural features that could be generalized across data sets from recent studies of cascade prediction [5, 21]. The selected features reflect density and centrality of a cascade graph including average node degree, number of leaf nodes, edge density.

Algorithm 1: DBSCAN for Clustering

Input: Sample Set $D = \{x_1, x_2, ..., x_m\}$, Distance Metric M^{m*m}, parameter$(\epsilon, MinPts)$;
$\Omega \leftarrow \emptyset$;
for $j = 1, 2, ..., m$ **do**
  $N_\epsilon(x_j) \leftarrow$ `FindNeighbor`(x_j, ϵ);
  if $|N_\epsilon(x_j)| \geq MinPts$ **then** $\Omega \leftarrow \Omega \cup \{x_j\}$;
$k \leftarrow 0$;
Initialize unvisited samples $T \leftarrow D$;
while $\Omega \neq$ **do**
  $T_{old} \leftarrow T$;
  Random select a core object $o \in \Omega$, initialize queue $Q =< o >$;
  $T = T \backslash o$;
  while $Q \neq$ **do**
    $q \leftarrow$ `deQueue`(Q);
    if $|N_\epsilon(x_j)| \geq MinPts$ **then** $\Delta = N_\epsilon(q) \cup T$, `enQueue`$(\Delta)$, $T \leftarrow T\ \Delta$;
  $k \leftarrow k + 1$, $C_k = T_{old} \backslash T$;
  $\Omega = \Omega \backslash C_k$;

Table 3 depicts our experimental parameters, where the default values of these parameters are in bold font.

Table 3: Experiments Setting.

Parameters	Values
Prediction period Δt	**1 year**, 2 year
Dimension D of the hidden state vector h	32, 50, 128, **256**, 512
Number of sampled sequences per cascade graph	10,20,50,70,100,150,**200**
Average length of sampled cascade sequence	1,2,3,**4**,5

Accuracy evaluation metrics. Since the possible value of size increase is infinite, mean squared error (MSE) are applied in our experiments to evaluate the accuracy of predictions, which is used in previous work of cascade prediction [32]. MSE is defined as follow in Eqn. (13). y is the truth value and is defined as $y_i = log_2(\Delta V_i + 1)$ where ΔV_i is the size increase of a cascade graph. $\hat{y}$ denotes the predicted value.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y_i} - \hat{y})^2 \tag{13}$$

4.2 Experiment Results

Effect of various prediction period We first compare the MSE score among those methods under two prediction period as shown in Tab. 4. The results shows that MUCA-LSTM always outperforms other methods under various prediction period. Moreover, deeplearning methods are better at near future size increase prediction while the performance of feature-based method is almost irrelevant to the period length.

Table 4: Effect of prediction period.

	Deepcas	Topo-LSTM	Feature-based	MUCA-LSTM
$\Delta t = 1\ year$	1.643	1.432	1.793	1.324
$\Delta t = 2\ year$	1.682	1.447	1.621	1.378

Running Time Comparison We use the default parameter settings as shown in Table 3 to test the running time of these methods. The result is shown in Tab. 5. Although $MUCA-LSTM$ spends the longest time for prediction, the time is comparable with another two deeplearning method: Deepcas and Topo-LSTM, and the MSE of MUCA-LSTM is smaller than those of the two methods. Feature-based method requires the minimal running time however the MSE is considerably larger than the three deeplearning method.

Table 5: Running Time Comparison.

	Deepcas	Topo-LSTM	Feature-based	MUCA-LSTM
Running time	207 mins	214 mins	154 mins	232 mins

Effect of Dimension of Hidden State Vectors. We study how the value of dimension D of hidden state vectors affect the performance of the MUCA-LSTM. As shown in Fig. 5, with the increase of dimension number, the MSE decreases. However, when the dimension number continues to get larger, the decrease degree gets smaller and finally reaches an almost stable level. This is because when the dimension is very small, the hidden state vectors may not fully reflected the information of a cascade graph. However when dimension reaches a certain level, it is capable to reveal enough information about each cascade graph so the following increase of dimension number has little effect for performance improvement.
Effect of average length of a cascade sequence. Since we represent a cascade graph by a set of sampled cascade sequences, we study how the average length of these cascade sequences can affect the performance of those deeplearning methods. The result is shown in . Because we limited the length of sampled cascade sequences by filtering those unqualified sequences, we change the default setting of sequence number $k = 200$ to a smaller one $k = 50$. We can observe that with the increase of average cascade sequence, all those deeplearning methods achieves a better MSE. This is because a longer sequence can be used to trace a longer history about how the content is diffused in the network. Moreover, MUCA-LSTM

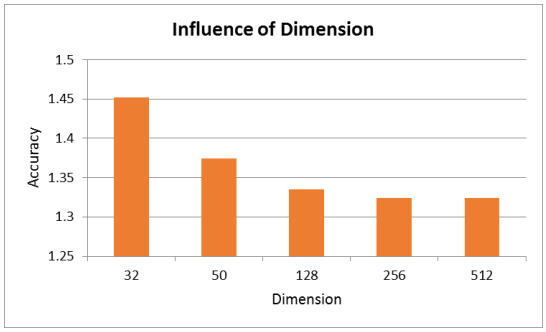

Figure 5: The effect of number of dimension of hidden state vectors

performs better compared with other methods when k gets larger while perform almost the same with baselines when the average length of sequences is too small. This is because in MUCA-LSTM, we incorporate each node's influencers' hidden state vectors into their output hidden state vectors and when the sequence is too short, there is little chance for a node to find lots of influencers so MUCA-LSTM becomes almost the same with single-content diffusion models as Deepcas and Topo-LSTM.
Effect of the number of sampled cascade sequences. At last we study the effect of number of sampled cascade sequences in each cascade graph and the result is shown in Fig. With the increase of sequence number, the MSE achieves a better result while the performance improvement is smaller when the number has reached a relatively large value. This is the more sequences, the better structural information of the cascade graph we can get. However when the number of sequence continues to increase, most of the information has been captured so the improvement is limited.

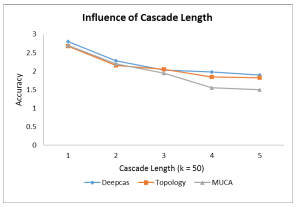

Figure 6: The effect of average sequence length

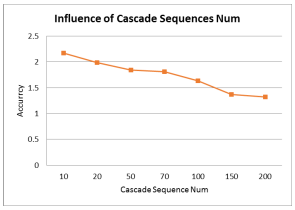

Figure 7: The effect of sequence number

5 VISUALIZATION

In order to visualize our work and facilitate the usage of experiment result, we deploy an on-line visualization system, named Paper Map on Acemap Website, to show the information and prediction result of each paper.

An overview of Paper Map is shown in Fig. 8. In the paper map, nodes represent paper, of which color distinguishes publication year and radius distinguishes the citation count of paper. Simultaneously, the line between the nodes represent the existence of the reference relationship between paper. As for layout, nodes are clustered by publication year, which helps visualize the citation trend of the papers more intuitively.

Through the system, user can search a specific paper they interested in by the search box, and the paper's position will show on the Paper Map. Users can also filter the paper according to topic to generate submap by option box.

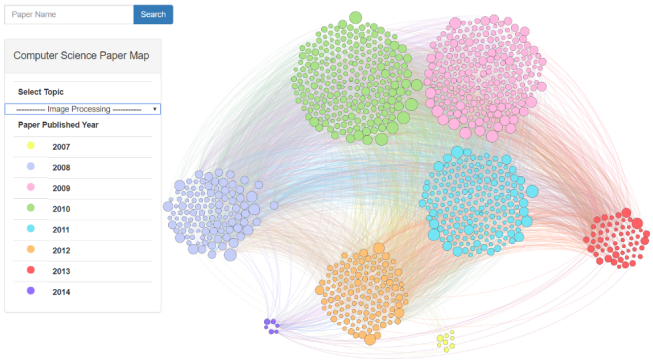

Figure 8: Overview of paper map system

More information of each paper can be available to users by clicking the paper nodes in Paper Map system. As illustrated in Fig. 9. By click effect, papers with reference relationship will be highlighted. The information panel will appears to show the essential information as well as trend and prediction result of citation count.

We also visualize the cascades of papers input to neural network topology by dynamic graph, where nodes representing the infected author, with color illustrating the infected year and lines illustrating the reference relationships between authors.

We remark that the goal of designed Paper Map system is to help researchers to get a more intuitive understanding of the experimental data structure and prediction results, which helps to get inspiration from visual effects of paper information and connection.

6 RELATED WORK

Information prediction are empirically proved to be predictable to some extent, including Tweets/microblogs [11, 13, 30], photos [5], videos [2] and academic papers [23]. In literature, it can be classified into two fashions.

Model-based Method. Early work mainly focus on model-based method with certain macroscopic distributions [15, 33] or stochastic processes that explain the microscopic actions of passing along the information. The basic models such as Independent Cascade (IC) Model [8, 22] and Linear Threshold (LT) Model [4, 9] are widely studied. Recent researches often take the detailed features into consideration like network structure and temporal information [5], user nodes' social roles and preferences [12, 27, 28], user edges' relationships [10, 14] as well as diffusion content or topic awareness [1, 26, 31].

In [32], Yang et al held that users may play multiple roles with respect to different social communities and proposed a Role-Aware information diffusion model (RAIN) which integrates social role recognition and diffusion modeling. A Gibbssampling based algorithm are then developed to learn the parameters using the historical diffusion dataset. As for [10], different types of relationships between of users were seen as indispensable to considered in heterogeneous networks and two variations of the linear threshold model are discussed. [12] took both users' preferences over topics and topic distributions of cascading messages into account and formulated the problem as non-smooth convex optimization problems with coordinate proximal gradient descent to solve while in [17] textual documents, social influences, and topic evolution are all unified and the whole diffusion process is regularized through a Gaussian Markov Random Field.

In these model-based method, these features usually makes strong assupmtions and the distribution are often assumed to be known with parametric formulations. However, they will no doubt oversimplify the reality, as a result, they generally underperform in real prediction tasks. Moreover, this requires much manual effort and extensive domain knowledge and seems hard to generalized to other applications.

Learning-based Method. With the development of machine learning techniques, recent work starts to exploit deep learning in prediction tasks to avoid explicit feature engineering for diffusion modeling. For instance, Embedded-IC [3] embedded users in a latent space to extract more robust diffusion probabilities. It differentiates two kinds of roles for the nodes; i.e., an active node serves as a "sender", and an inactive node serves as a "receiver", which each learns a vector as a node's embedding. Then, it models an activation based on the closeness between an inactive node's receiver embedding vector and the active nodes' sender embedding vectors. DeepCas [16] applied Gated Recurrent Unit (GRU) [6] to predict the cascade size in the future. The input data of RNN are extracted from the cascades and represents as a sequence of nodes. It firstly models the cascade at each time step with an induced subgraph over the active nodes and then sampled the subgraph into some random walk paths. However, [29] pointed out that a cascade has a richer structure indicating the diffusion process instead of merely a sequence of nodes ordered by their activation time stamps. Hence, they revised the structure of RNN and introduced Topo-LSTM, to represent the diffusion topology.

Standard RNN are designed for sequences, frequently applied in music and speech domains. Nowadays, more variants of RNN structures are studied to adapt to various applications. Tree structured RNN are exploited especially in natural language processing. For example, [25], not limited in linear chains, introduced a generalization of LSTMs to tree-structured network topologies to predict semantic relatedness and sentiment classification. In [7], a RNNG is developed to encode a parse tree. There are also some RNN variants for directed acyclic graphs (DAGs). In [24], DAG-RNNs are proposed to deal with DAG-structured images for Scene Labeling. It modeled long-range contextual dependencies among image units and eventually enhanced the discriminative power of local representations tremendously.

Our work differs from this literature as we take the influence of other contents' diffusion in the social networks into account to predict the future cascade size. We focus on the feature extraction of multi-cascade and LSTM model design to incorporate multi contents for better prediction.

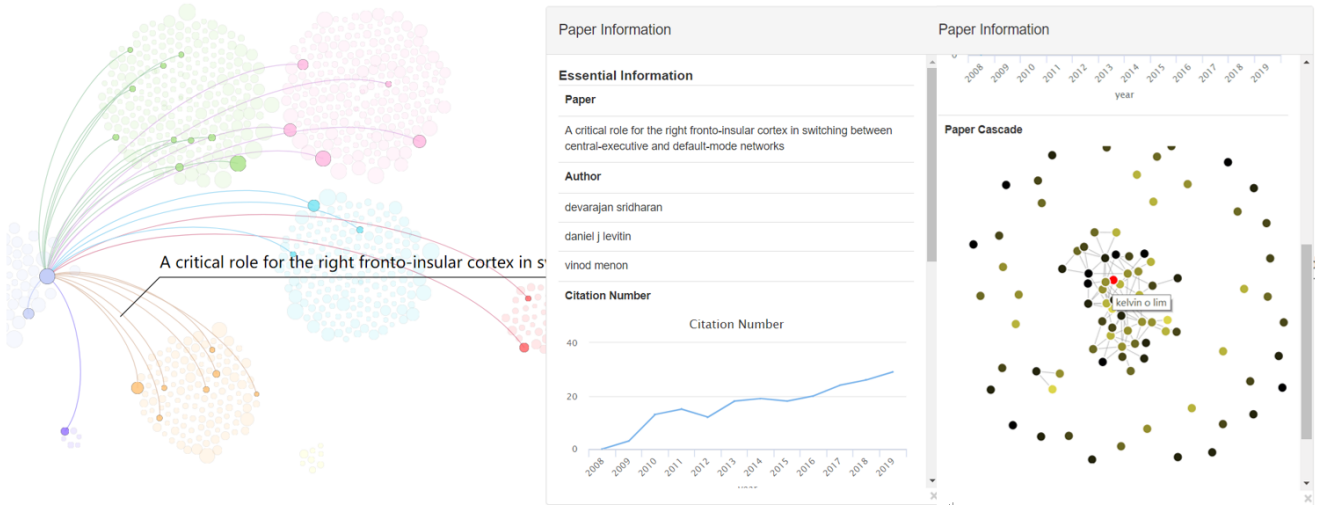

Figure 9: Paper Information.User can search the prediction results of citation count and cascade sequence sampling of papers by click effect.

7 CONCLUSION

In this paper, we tackle the problem of cascade prediction which aims at predicting the number of newly activated nodes in a given future period. We take the influence of another simultaneously diffused content into consideration which largely improves the performance of prediction compared with some state-of-art baselines. We introduce MUCA: a multi-content-aware RNN network for cascade prediction which contains 3 steps: cascade sequence sampling, MUCA-LSTM and cascade graph representation. Our extensive experiments on both real and synthetic datasets verifies the effectiveness of our proposed method. In the future, we will focus on the generalization of model design that incorporates more than two contents diffused in the networks for better predition.

REFERENCES

[1] Nicola Barbieri, Francesco Bonchi, and Giuseppe Manco. 2012. Topic-aware social influence propagation models. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*. IEEE, 81–90.

[2] Christian Bauckhage, Fabian Hadiji, and Kristian Kersting. 2015. How Viral Are Viral Videos?. In *ICWSM*. 22–30.

[3] Simon Bourigault, Sylvain Lamprier, and Patrick Gallinari. 2016. Representation learning for information diffusion through social networks: an embedded cascade model. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 573–582.

[4] Wei Chen, Yifei Yuan, and Li Zhang. 2010. Scalable influence maximization in social networks under the linear threshold model. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 88–97.

[5] Justin Cheng, Lada Adamic, P Alex Dow, Jon Michael Kleinberg, and Jure Leskovec. 2014. Can cascades be predicted?. In *Proceedings of the 23rd international conference on World wide web*. ACM, 925–936.

[6] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[7] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. *arXiv preprint arXiv:1602.07776* (2016).

[8] Kiran Garimella, Aristides Gionis, Nikos Parotsidis, and Nikolaj Tatti. 2017. Balancing information exposure in social networks. In *Advances in Neural Information Processing Systems*. 4666–4674.

[9] Mark Granovetter. 1978. Threshold models of collective behavior. *American journal of sociology* 83, 6 (1978), 1420–1443.

[10] Huan Gui, Yizhou Sun, Jiawei Han, and George Brova. 2014. Modeling topic diffusion in multi-relational bibliographic information networks. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 649–658.

[11] Adrien Guille and Hakim Hacid. 2012. A predictive model for the temporal dynamics of information diffusion in online social networks. In *Proceedings of the 21st international conference on World Wide Web*. ACM, 1145–1152.

[12] Qingbo Hu, Sihong Xie, Shuyang Lin, Wei Fan, and Philip S Yu. 2015. Frameworks to encode user preferences for inferring topic-sensitive information networks. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, 442–450.

[13] Maximilian Jenders, Gjergji Kasneci, and Felix Naumann. 2013. Analyzing and predicting viral tweets. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 657–664.

[14] Jong-Ryul Lee and Chin-Wan Chung. 2014. A new correlation-based information diffusion prediction. In *Proceedings of the 23rd International Conference on World Wide Web*. ACM, 793–798.

[15] Kristina Lerman and Rumi Ghosh. 2010. Information contagion: An empirical study of the spread of news on Digg and Twitter social networks. *Icwsm* 10 (2010), 90–97.

[16] Cheng Li, Jiaqi Ma, Xiaoxiao Guo, and Qiaozhu Mei. 2017. DeepCas: An end-to-end predictor of information cascades. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 577–586.

[17] Cindy Xide Lin, Qiaozhu Mei, Jiawei Han, Yunliang Jiang, and Marina Danilevsky. 2011. The joint inference of topic diffusion and evolution in social communities. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE, 378–387.

[18] Su-Chen Lin, Shou-De Lin, and Ming-Syan Chen. 2015. A learning-based framework to handle multi-round multi-party influence maximization on social networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 695–704.

[19] Wei Lu, Wei Chen, and Laks VS Lakshmanan. 2015. From competition to complementarity: comparative influence diffusion and maximization. *Proceedings of the VLDB Endowment* 9, 2 (2015), 60–71.

[20] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[21] A. Bhatnagar R. Guo, E. Shaabani and P. Shakarian. 2015. Toward order-of-magnitude cascade prediction.. In *ASONAM*.

[22] Kazumi Saito, Ryohei Nakano, and Masahiro Kimura. 2008. Prediction of information diffusion probabilities for independent cascade model. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, 67–75.

[23] Hua-Wei Shen, Dashun Wang, Chaoming Song, and Albert-László Barabási. 2014. Modeling and Predicting Popularity Dynamics via Reinforced Poisson Processes.. In *AAAI*, Vol. 14. 291–297.

[24] Bing Shuai, Zhen Zuo, Bing Wang, and Gang Wang. 2016. Dag-recurrent neural networks for scene labeling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3620–3629.

[25] Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* (2015).
[26] Oren Tsur and Ari Rappoport. 2012. What's in a hashtag?: content based prediction of the spread of ideas in microblogging communities. In *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 643–652.
[27] Farman Ullah and Sungchang Lee. 2017. Identification of influential nodes based on temporal-aware modeling of multi-hop neighbor interactions for influence spread maximization. *Physica A: Statistical Mechanics and its Applications* 486 (2017), 968–985.
[28] CX Wang, XH Guan, Tao Qin, and YD Zhou. 2015. Modeling on opinion leaderâĂŹs influence in microblog message propagation and its application. Ruan Jian Xue Bao. *Journal of Software* 26, 6 (2015), 1473–1485.
[29] Jia Wang, Vincent W Zheng, Zemin Liu, and Kevin Chen-Chuan Chang. 2017. Topological recurrent neural network for diffusion prediction. *arXiv preprint arXiv:1711.10162* (2017).
[30] Lilian Weng, Filippo Menczer, and Yong-Yeol Ahn. 2014. Predicting Successful Memes Using Network and Community Structure.. In *ICWSM*.
[31] Tianbao Yang, Rong Jin, Yun Chi, and Shenghuo Zhu. 2009. Combining link and content for community detection: a discriminative approach. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 927–936.
[32] Yang Yang, Jie Tang, Cane Wing-ki Leung, Yizhou Sun, Qicong Chen, Juanzi Li, and Qiang Yang. 2015. RAIN: Social Role-Aware Information Diffusion.. In *AAAI*. 367–373.
[33] Linyun Yu, Peng Cui, Fei Wang, Chaoming Song, and Shiqiang Yang. 2015. From micro to macro: Uncovering and predicting information cascading process with behavioral dynamics. In *Data mining (ICDM), 2015 IEEE international conference on*. IEEE, 559–568.