Prediction of Information Cascades via Content and Structure Integrated Whole Graph Embedding

Xiaodong Feng[1], Qihang Zhao[1], Zhen Liu[2]
[1]School of Public Affairs and Administration,University of Electronic Science and Technology of China
[2]School of Computer Science and Engineering,University of Electronic Science and Technology of China
fengxd1988@hotmail.com,sxg1224@foxmail.com,quake@uestc.edu.cn

Abstract

Understanding the mechanisms governing how one message acquires more popularity than another, modeling how it gains popularity dynamically, and determining the method for predicting its dynamics popularity are of tremendous interest to related decision support systems. The prediction of information cascade begins benefiting from the development of representation learning of graphs using deep learning techniques. However, recent studies are learning the representation of nodes in the graph, which is not suitable of the information cascade prediction problem as the information cascades contain all nodes in the dissemination path as a whole. Inspired by recent successes of deep learning in various data mining tasks, we investigate whether the whole network of information cascade could be embedded in low dimension and whether it could effectively predict the future size of cascades. Besides the learning of each node in the cascade, we design a method to automatically learn the representation of each cascade graph as a whole by building the content-based and structure-based graph whose node refers to each whole cascade. Our results on real datasets show the significant improvement over baselines including feature-based methods, node embedding methods and cascade representation by attention mechanism.

1 Introduction

The ubiquity emergence of online social platforms, e.g., Twitter, Sina Weibo and Facebook, brings unprecedented convenience for us to publish and deliver online content, including tweets, microblogs, videos, and images, which are placing the economy of attention in the center of this era. When users constantly reshare the information posted or shared by others he follows, information cascade will become popular. Among the large number of information cascades, only a small amount will receive most attention from users, while most of these gain few reshares. Thus, being able to accurately predict the future size of information cascades, that is, how many reshares a give online content will acquire, will benefit both users and the owners of platforms a lot.

Though it is challenging to predict the popularity of these contents as social platforms are generally large-scale open system and may be affected by extrinsic factors, many have proved the predictability of cascades and develop many methods to solve it [Cheng *et al.*, 2014; Zhao *et al.*, 2015]. Generally, current methods fall into two categories: generative approaches and feature-based approaches. Generative approaches recognize the popularity of information cascades over time as a dynamic time series fitting problem [Gomez-Rodriguez *et al.*, 2013; Gao *et al.*, 2015], and thus develop certain macroscopic distributions or stochastic processes based on various strong assumptions. Alternatively, feature-driven methods formulate it as a classification or regression tasks and solve it by machine learning techniques that incorporate extensive features. The futures are usually designed based on the knowledge from both social theory and empirical analysis, including content features [Hong *et al.*, 2011], sentiment features [Berger and Milkman, 2012] and network structure [Zhang *et al.*, 2013; Gao *et al.*, 2014] of the cascades. Among these, features extracted from the network structure of the cascade are shown to be powerful by multiple studies [Gao *et al.*, 2014].

Though these features-driven approaches are proven to outperform generative approaches for real prediction task, there are still numerous features to be considered as the performance heavily depends on the hand-crafted features. The features are often based on human's prior domain knowledge and may be specific to particular platform. However, there is no common methodology of designing and measuring the features. To overcome this major deficiency of feature-driven approaches, inspired by the recent success of deep learning for network learning, one state-of-arts study [Li *et al.*, 2017] investigates an end-to-end learning method for cascade prediction. It can automatically learn the most predictive feature representations of the input cascade graphs, under the framework of deep neural network, which is actually mapping these representations to the final cascade size.

While deep learning for graph learning have demonstrated their ability of dealing with nodes in graph for multiple tasks, including recommendation, node classification or clustering, as well as cascade prediction problem. Although, the architecture in [Li *et al.*, 2017] can represent the graph as a whole through an attention component to aggregate the multiple representation of paths containing user nodes in the graph, it is still based on the node-level embedding as the attention mechanism is essentially seen as the weighted average of representation of each node. Thus, how to design a robust system to learn the representation of graph as a whole is still a challenging task, since the future size of an information cascade corresponds to the whole graph that depicts the process of the cascade, not each node in the graph.

We present a novel deep learning framework to predict the size of information cascade, which first constructs a new graph (noted as *graph*^{2}) with each cascade graph is analogous to the vertices in *graph*^{2}.To design the edges in *graph*^{2}, we respectively develop content-based (noted as *DeepCon*) and structure-based (noted as *DeepStr*) ways to measure how much each pair of two cascade graphs share the identical influential users and how similar the dynamic process of cascade is. Then the embedding of a cascade graph as a whole is learned by feeding *graph*^{2} into a second order random walk to generate cascade sentence as the input of semi-supervised *Skip-gram*, followed by a Multiple Layer Perception (MLP) to map the learned embeddings to the growth size of information cascades. The proposed method is evaluated on real information cascade datasets by comparing with baselines including features-driven approach, node embedding as well as graph embedding using attention mechanism.

2 Related Work

Our work lies in the cross road of cascade prediction and graph representation, so the recent studies in these fields will be briefly introduced.

2.1 Cascade Prediction

The information cascade prediction methods mainly fall into two groups: feature-driven approaches and generative approaches. Feature-driven approaches treat it as a classification [Gao *et al.*, 2014] or regression [Pinto *et al.*, 2011] task and apply machine learning techniques to model the contribution of the informative features for future popularity. They have empirically revealed the power of temporal features, content features, topological structure of cascade and profile of users engaged in the cascade. Generative approaches typically consider the gain of cascade popularity as a cumulative stochastic process [Yu *et al.*, 2015], modelling it as a parametric model and then estimating the parameters mainly based on the observed time series data. However, these generative methods are directly designed to model the popularity and not to predict it. The models are usually involving strong assumptions, which are oversimplifying the reality and thus they generally underperform in real tasks.

For feature-driven approaches, the performance heavily depends on the hand-crafted features, which may not be directly applied when they are outside particular context and are thus hard to be generalized. *DeepCas* [Li *et al.*, 2017] was proposed to automatically learn the representations of a cascade without manual feature design taking advantage of recent deep network technique as an end-to-end cascade prediction method. *DeepHawkes* [Cao *et al.*, 2017] extended it to consider the time decay effect by means of Hawkes process. Though these two methods have proven magically powerful for cascade prediction, both were actually learning representation of identities of nodes in cascade graph not the graph as a whole, which may limit the performance.

2.2 Graph Representation

Graph representation to learn embedding the nodes in a graph, is instrumental for machine learning tasks that focused on network data, and has gained much attention over the past decades from different communities. It can be traced back to graph embedding for manifold learning such as Laplacian Eigenmaps [Belkin and Niyogi, 2002]. Recently, *Word2Vec* [Mikolov *et al.*, 2013] was proposed to learn distributed dense representation of words from sparse text corpus based on *Skip-Gram*, which can place semantically similar words near each other in space. Inspired byit, *DeepWalk* [Perozzi *et al.*, 2014] was first proposed to learn the language model from a network by using random walks to generate sequences of nodes from a network. In the learned embedding space, nodes close in the network tend to be near. To capture a diversity of network structures, *node2Vec* [Grover and Leskovec, 2016] generated biased second order random walks rather than uniform ones. Different from learning the embedding for identifying local context of nodes, *truc2vec* [Ribeiro *et al.*, 2017] can learn the representations for the structural identity of nodes. Also, some others exploited deep learning frameworks, such as autoencoder [Wang *et al.*, 2016; Gao and Huang, 2018] to learn the embedding of network.

Although there are recently many works focusing on the graph representation learning, they are mostly learning the embedding of each node in the graph. We will try to extent these works to learn the embedding of a whole graph for cascade prediction.

3 Methodology

In this section, we begin with the formal definition of cascade prediction studied in this paper, and then introduced the proposed model.

3.1 Problem Definition

Definition 1 (*global network*).Given a snapshot of a social network as $G = (V, E)$ where V is the set of vertices and $E \subset V \times V$ is the set of edges. A node i represents an actor, e.g., a user in Twitter or an author in the academic paper network, and an edge $(i, j) \in E$ represents a social tie, e.g., retweeting or citation between nodes i and j.

Definition 2 (*cascade graph*). Let C be the set of cascades, containing numbers of cascades as $C=\{c\}$. Each snapshot of cascade c at time t is described by a subgraph $g_c^t =(V_c^t, E_c^t) \in G$, V_c is a subset of nodes in V that have contributed for the cascade c at observed time, an edge $(i_c, j_c) \in E_c^t$ denotes the link between i_c and j_c.

Definition 3 (*growth size*). It can be described as the increment of the size of cascade c after a given time interval

Δt, denoted as $\Delta V_c = |V_c^{t+\Delta t}| - |V_c^t|$. The growth size of a cascade is exactly what we are concerned about and to predict given global network G and cascade graph g.

The framework of our model takes the cascade graph as input and outputs growth size (shown as figure 1). It first constructs a new graph to describe how cascades are virtually connected with each other. Then we generate the cascade context by means of a second-order random walk, and learn the embeddings of cascades using Skip-gram. Finally, the embeddings are fed into a Multiple Layer Perception to fit the growth size of cascades.

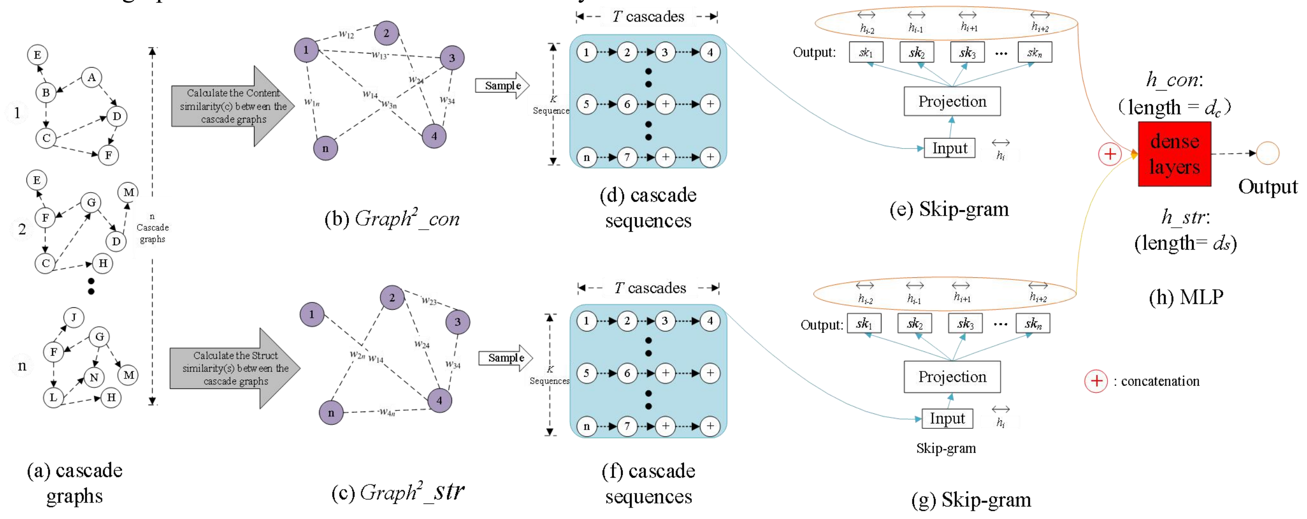


Figure 1: The framework of proposed *DeepCon* and *DeepStr*.

3.2 Constructing High order Graph containing Cascades

To learn the embedding of cascade graph as a whole using language model, the context of cascade graphs has to be first constructed. Thus, we construct a high order weighed graph containing all the cascades as vertices.

Definition 4 (High order cascade graph: Graph²). Given a cascade set C, we define the high order cascade graph to encode the similarity between cascades as $Graph^2 = (C, E^2, W)$, where E^2 is the extracted similarity-based links between cascades and W is the corresponding weight.

Content-based *Graph²*

The cascades are typically passed through nodes and different nodes have different social influence for cascades, e.g. the role of opinion leaders in information dissemination. Naturally, the growth size of a cascade highly depends on nodes who propagates the cascades (e.g. retweeting tweets and citing academic papers), which are the nodes in the cascade graph at observation time. Thus, we assert that cascades sharing more identical nodes in the propagation path will be more likely to have similar growth size in future. Accordingly, we first propose to measure the similarly between cascades based on how similar the nodes in them are close, as *content similarity*.

Let $g_1 = (V_1, E_1)$, $g_2 = (V_2, E_2)$ denote two cascade graphs with V_1, V_2 denoting the vertex sets and E_1, E_1 denoting the edge sets correspondingly. The content similarity between them is determined by how many common nodes shared in V_1 and V_2 weighted by the influence of nodes, formulated as:

$$w\_con(g_1, g_2) = \sum_{v \in V_1 \cap V_2} deg(v) \quad (1)$$

where $deg(v)$ denotes the degree of node v in global network G. Then we can link two cascades with similarity between them exceeding the average similarity of all pair cascades and set corresponding similarity be weight as $Graph^2\_con$.

Structure-based *Graph²*

For all cascades, the graph between nodes can be abstracted into different topological structures, such as tree, star and network. It has been empirically examined by hand-crafted features that different topological structures may result in different potential dissemination trend of cascades, and thus make difference for growth size. Therefore, we then propose to measure the structural similarity between different cascade graphs to automatically learn the structural features from the *High order cascade graph* ($Graph^2\_str$) based on it. To measure the structural similarity of two cascade graph g_1 and g_2, we refer to [Ribeiro *et al.*, 2017] to compute the structural similarity, defined as follows.

Let $R_k(u)$ denote the set of nodes at distance (hop count) exactly $k \geqslant 0$ from u in cascade graph g_c. Note that $R_1(u)$ is exactly the set of neighbors of u and $R_0(u)$ is u itself. Let $s(S)$ denote the ordered degree sequence of a set $S \subset V$ of nodes. Let r_1 and r_2 be the root nodes (e.g. the users who publish the original information or the author of the academic paper to be cited) of cascades g_1 and g_2, respectively. The structural similarity of these two cascades can be measured by summing the distances between the degree sequence of nodes at all possible distance k to root nodes as:

$$w\_str(g_1, g_2) = e^{-D(r_1, r_2)}$$
$$D(r_1, r_2) = \sum_{k=0:K} l(s(R_k(r_1)), s(R_k(r_2)))) , \quad (2)$$

where $l(s_1, s_2)$ measures the distance between sequences s_1 and s_2. Note that $R_k(r_1)$ and $R_k(r_2)$ are only defined when there both exists nodes at distance k to r_1 and r_2, so the K should be the minimum value of two maximum distances K_1 and K_2 over all nodes to the root nodes. The weights are inversely proportional to structural distance, and assume values equal to 1 only if $l(R_k(r_1), R_k(r_2)) = 0$ at all k.

As $s(R_k(r_1))$ and $s(R_k(r_2))$ could be of different sizes, we adopt Dynamic Time Warping (*DTW*) [Salvador and Chan, 2007] to measure the distance $l(s_1, s_2)$ between two ordered degree sequences, a technique that can cope better with sequences of different sizes and loosely compares sequence patterns. *DTW* aims to find the optimal alignment between two given sequences of different lengths, so that the total distances between two sequences is the smallest given a distance function $d(a, b)$ for two scalars. Here, we adopt the same function as in [Ribeiro *et al.*, 2017]: $d(a, b)= max(a, b)/ min(a, b)$-1. Obviously, $d(a, b)$=0 when a=b, implying that distance of two identical ordered sequences will be 0.

Based on the definition of weight function, we connect two cascades $<g_1, g_2>$ in *Graph*^{2}*_con* if $w\_str\ (g_1, g_2)$ exceeds the average similarity over all pair cascades and set corresponding similarity corresponding weight.

3.3 Learning a Semi-supervised Language Model

Making an analogy between cascade in *Graph*^{2} and words in document, we can sample cascade sequences from *Graph*^{2} as to be analogous to sentences. Existing ways of generating paths from network are mostly based on random walk. The biased random walk in *Node2Vec* that considers both breadth-first and depth-first sampling strategies is applied to generate context for cascades from *Graph*^{2}.

Let $N(g)$ be a neighborhood list of cascade c generated through a neighborhood sampling strategy. Supposing the embedding representation is denoted as $H=\{h_1, h_2,\dots h_n\}$, where n is the number of cascades and $h_g \in R^d$ denotes the embedding of cascade graph g. By extending the Skip-gram framework to cascades, the following objective can be optimized to maximizes the log-probability of observing the cascade neighborhood $N(g)$ for all $g \in$ C, as:

$$\max_{\{h_1,h_2,\dots h_n\}} O^{skip-gram} = \sum_{g\in C} (-\log Z_g + \sum_{p\in N(g)} (h_p \cdot h_g)). \quad (3)$$

The per-cascade normalization factor, $Zg = \Sigma_{p\in C}\, exp(h_p \cdot h_g)$ is expensive to compute for *Graph*^{2} with large number of cascades and we approximate it using negative sampling as in *Node2Vec*. Besides the optimization of (3) to encode the similarity between cascades, to predict the growth size accurately, the labeled growth size of cascades in training set can regularize the learning process to generative discriminative embeddings, as a semi-supervised machine learning method. Accordingly, we integrate to minimize the square loss between predicted growth size and ground truth into Eq. (3), where we exploit a multi-layer perception (MLP) as an end-to-end prediction as:

$$\min_{\{h_1,h_2,\dots h_{n_t}\}} O^{loss} = \sum_{c\in C^t} (\hat{y}_c - y_c)^2$$
$$\hat{y}_c = MLP(h_c) \quad , \quad (4)$$
$$y_c = \log_2(\Delta V_c + 1)$$

where C^t denotes the set of cascades in training set. We take log-transformation for the ground truth growth size as the original square loss could be easily affected by outliers [Li *et al.*, 2017].

Thus, the semi-supervised language model is reached by optimizing the following objective as:

$$\min_{\{h_1,h_2,\dots h_{n_t}\}} -O^{skip-gram} + \lambda O^{loss} , \quad (5)$$

where λ>=0 is the weight to balance the weight between them. Eq. (5) can be optimized using stochastic gradient ascent over the model parameters defined in features H.

4 Experiments

In order to evaluate the performance of proposed method, we conducted a comprehensive experiment using data sets from the real world.

4.1 Data sets and Evaluation Metric

We adopted two data sets altogether. In the first scenario, we evaluate the prediction of the cascades of scientific papers. We refer to the AMINER data set that [Li *et al.*, 2017] collated and use the simplified version downloaded from https://github.com/chengli-um/DeepCas. The data set consists of a global network and three cascade network sets. There are 9860 nodes and 560 cascade graphs in the data set, each node representing the author of a paper, simultaneously, a cascade graph composes by the author and the citer of a paper. The global network G based on citations between 1992 and 2002. The training set consists of papers published from 2003 to 2007. Papers published in 2008 and 2009 are used for validation and testing.

The second data set is from Weibo, which is organized from [Cao *et al.*, 2017] in June 1, 2016. In order to fit our needs, we reorganize the Weibo data set. We have sorted out and selected 1565 cascade graphs, which contain 108839 nodes. Each node represents a Weibo user. Then, according to the selected cascade graphs, we construct a global social network G, which contains the relevant information how all nodes are connected. Next, we randomly selected 1200 diagrams from 1565 cascade diagrams as training set, 195 diagrams as validation and 170 diagrams as testing.

We use the mean square error (MSE) of the test data set to evaluate the accuracy of the prediction, which is a common choice for regression tasks and was used in previous cascade prediction works. Denote $\hat{y}$ a prediction value, and y is the true value, the *MSE* is:

$$MSE = \frac{1}{n} \sum_{c\in C^s} (\hat{y}_c - y_c)^2 , \quad (6)$$

where C^s is the cascade set of testing set. Clearly, lower MSE implies better performance.

4.2 Baseline Methods and Parameter Settings

In order to compare with the above methods, we select 3 baselines, including methods that based on nodes embeddings, features of graph used for cascade prediction.

Feature-linear. We select several features of cascade graph and using linear regression with L_2 regularization to predict. The features include: *Number of leaf nodes* (leaf nodes refer to nodes whose degree is equal to 1 in a graph); *Node number* (the number of nodes in a graph); *Edge number* (the number of edges in a graph); *average degree* (the ratio of the degree of all nodes to the number of nodes in the graph); *Density of edge* (the ratio of the number of edges to possible edges as $n*(n-1)/2$. Table 1 shows the statistics of these features of two datasets.

Features	sets	AMINER	Weibo
Avg. Number of leaf nodes per graph	train	0.7	70.9
	val.	0.2	87.0
	test	1.6	87.3
Avg. Node number per graph	train	19.7	75.5
	val.	17.8	92.2
	test	20.3	92.2
Avg. Edge number per graph	train	73.6	75.1
	val.	57.6	92.1
	test	65.6	91.6
Avg. Average degree per graph	train	6.7	1.9
	val.	6.2	1.9
	test	5.8	1.9
Avg. Density of edge per graph	train	0.4	0.1
	val.	0.4	0.1
	test	0.4	0.1

Table 1: Statistics of extracted futures of two datasets.

Node2vec [Grover and Leskovec, 2016] is selected as a representative of node embedding methods. We learn two embedding vectors from global graph and cascade graph, and join them together to form a new embedding for each node. The average of embeddings of all nodes in a cascade graph is fed into MLP to make the prediction.

DeepCas [Li *et al.*, 2017] is an end-to-end neural network framework that takes as input of the cascade graph and predicts the growth size.

We use **DeepCon**, **DeepStr** and **DeepCon+DeepStr** to denote the three variants of our proposed methods, which take the embedding from *Graph*^{2}*_con*, *Graph*^{2}*_str* and both two as the input of MLP.

We sample K = 200 paths each with length T = 10 from the cascade graph for DeepCas as suggested. For proposed DeepCon and DeepStr, we set T=14. The iteration number is 1000. For *DeepCas*, consistent with [Li *et al.*, 2017], the mini batch size is set to 32 and the smoother α is set to 0.01. The node2vec hyper parameters p and q are set to 1. The final sizes d of embedding when fed into a 4-layer MLP for the both data set are set to 64. The corresponding number of neurons of different layers in MLP is a 64*32*16*1.

4.3 Overall Performance

The overall performance of all competing methods across data sets is displayed in Table 2. Note that the values in Table 2 are logarithmically processed, and if they are converted to normal values, they will be larger than the values in the Table 2. '*DeepCas+DeepCon*' denotes the method to take the embedding from both *deepCas* and *DeeCon* as the input of MLP, the same way for "*DeepCas+DeepStr*" and "*DeepCas+DeepCon+DeepStr*". It is demonstrate that our prosed DeepCon and DeepStr outperform embedding-based approaches Node2Vec and DeepCas, showing the power of learning the embedding of the whole cascade graph as a whole for prediction.

methods	AMINER	Weibo
Feature-linear	1.34	2.46
Node2vec	2.00	5.90
DeepCas	2.08	17.08
DeepCon	1.68	3.3
DeepStr	1.59	2.33
DeepCon+DeepStr	1.38	2.30
DeepCas+DeepCon	1.42	1.39
DeepCas+DeepStr	1.33	1.28
DeepCas+DeepCon+DeepStr	1.27	1.30

Table 2: Compared performance measured by MSE.

Among all competitors, DeepCas + Con + Str is the best one, which is what we expected since it makes use of the node embedding of cascade graphs, the structural similarity and content similarity between cascade graphs.

Feature-linear, as a traditional and elementary prediction method, it works well and outperforms Node2Vec, DeepCas and DeepCon. This shows that if we can find several better features of cascade graph, we can use this method to predict and obtain higher accuracy. At the same time this method is obviously uncertain and unstable. If we cannot find suitable features, the final prediction results will be very inaccurate. Carefully designed features from the literature have already been powerful in capturing the critical properties of cascade graphs. The benefit of deep learning comes from the end-to-end procedure, which is likely to learn high-quality features that can better represent these network properties.

Node2vec, as a node embedding technology, doesn't work well on both of AMINER and Weibo data set. It only takes the average value of node embedding vectors, that is, only the nodes in the graph are used to represent the network, while ignoring other structural and content information in cascades.

DeepCas, an end-to-end neural network framework that takes as input of node sequence of the cascade graph, performs worse than proposed DeepCon and DeepStr. Also, Node2Vec and DeepCas work so poor in Weibo Data. Comparing the structure feature statistics in Table 1, we find cascade in Weibo dataset is much shallower (*edge number is close to number of leaf nodes*) and sparser (*density of edge is* 0.1) than AMINER. It reveals that node embedding can not perform well for sparse and shallow graph as the context generated from the node sequence will be limited.

DeepCon and DeepStr have achieved good results on both datasets, and are superior to existed deep neural network-based approaches in accuracy and robustness on both spare and dense graph from these two datasets. DeepCon works somehow poor on Weibo, which can be explained by the powerful effect of structure similarity of smaller number of shared nodes between cascades in this dataset, which would lead to sparse content similarity-based *Graph^2_con* and insufficient cascade context.

4.4 Computational Cost

Table 3 shows the computation time for each method to train the model.

methods	AMINER	Weibo
Node2vec	21.83	275.63
DeepCas	1417.6	3675.66
DeepCon	33.2	309.59
DeepStr	33.69	238.81
DeepCon+Str	28.18	228.48
DeepCas+Con	1498.63	3915.15
DeepCas+Str	1526.85	3720.43
DeepCas+Con+Str	1584.57	3781.61

Table 3: Computation time measured by seconds.

It is shown that DeepCas takes much longer time than other methods. Our DeepCon and DeepStr improve the accuracy and stability while greatly reducing the computation time.

4.5 Convergence Analysis

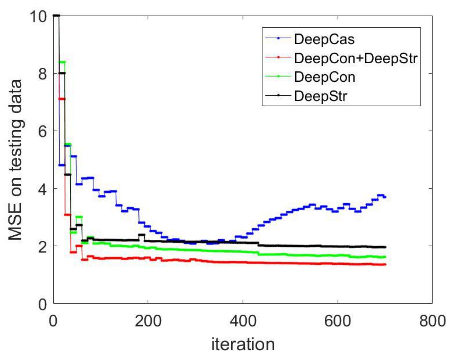


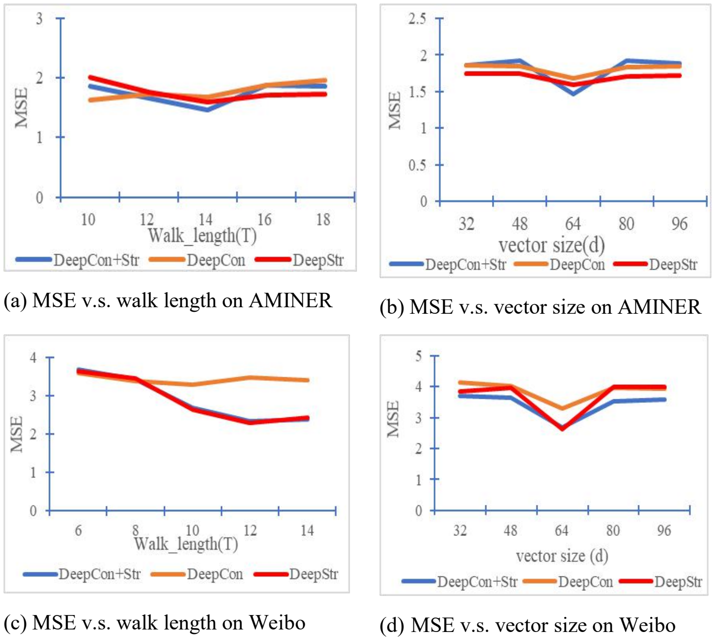
Figure 2: Comparison of convergence of different methods.

We compare the convergence (how the MSE on tested cascades changes over each iteration) of proposed methods with another end-to-end method DeepCas, as in Figure 2.

It is shown that proposed methods (DeepCon\ DeepStr \ DeepCon+DeepStr) will gain a low MSE very fast after about 80 iterations. However, the MSE of DeepCas will reach low level after about 200 iterations. Furthermore, the MSE is not robust as it will be raised after about 400 iterations due to overfitting on training set, which can be magically avoided in the iteration of our proposed methods.

4.6 Parameter Sensitivity

In order to evaluate how performance changes to the parameterization of our methods, we examined the parameter sensitivity by change the size of embedding vector (*d*) and the length of cascades in each random walk sequence (*T*) when fixing other parameters, as shown in Figure 3. It is found that with the increase of T, the prediction declines at first and keep stable. This was can be explained that as the number of cascades in the sequence increase, the effect of each walk would capture more context within cascades and will reach a boundary. For *d* the optimal size is 64 as smaller vector will ignore some potential features and larger vector contains some redundant information.



(a) MSE v.s. walk length on AMINER

(b) MSE v.s. vector size on AMINER

(c) MSE v.s. walk length on Weibo

(d) MSE v.s. vector size on Weibo

Figure 3: Parameter sensitivity analysis.

5 Conclusion

We present an end-to-end deep neural network model for cascade prediction by learning the embedding of the cascade as a whole rather than aggregating embeddings of each nodes in cascade. A high-order graph is firstly constructed to capture the node-content and structure similarity, followed by a semi-supervised skip-gram model by integrating minimization of the estimated loss of training cascades with MLP predictor. The proposed method can perform better than feature-based and node embedding-based methods, with lower computation cost and robust results over iterations.

One important future work would be how to encode other rich information if available in addition to the graph structure and node identities of cascades, such as text, time series, and how to exploit other deep neural network, e.g.

autoencoder, recurrent neural network, to improve the prediction accuracy.

References

[Balasubramanian and Schwartz, 2002] Mukund Balasubramanian and Eric L Schwartz. The ISOMAP algorithm and topological stability. *Science*, 295(5552):7–7, 2002.

[Belkin and Niyogi, 2002] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.

[Berger and Milkman, 2012] Jonah Berger and Katherine L. Milkman. What makes online content viral? *Journal of marketing research*, 49(2):192–205, 2012.

[Cao *et al.*, 2017] Qi Cao, Huawei Shen, Keting Cen, Wentao Ouyang, and Xueqi Cheng. Deephawkes: bridging the gap between prediction and understanding of information cascades. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1149–1158. ACM, 2017.

[Cheng *et al.*, 2014] Justin Cheng, Lada Adamic, P. Alex Dow, Jon Michael Kleinberg, and Jure Leskovec. Can cascades be predicted? In *International Conference on World Wide Web*, pages 925–936. ACM, 2014.

[Gao and Huang, 2018] Hongchang Gao and Heng Huang. Deep attributed network embedding. In *International Joint Conference on Artificial Intelligence*, pages 3364–3370, 2018.

[Gao *et al.*, 2014] Shuai Gao, Jun Ma, and Zhumin Chen. Effective and effortless features for popularity prediction in microblogging network. In *International Conference on World Wide Web*, pages 269–270. ACM, 2014.

[Gao *et al.*, 2015] Shuai Gao, Jun Ma, and Zhumin Chen. Modeling and predicting retweeting dynamics on microblogging platforms. In *ACM International Conference on Web Search and Data Mining*, pages 107–116, 2015.

[Gomez-Rodriguez *et al.*, 2013] Manuel Gomez-Rodriguez, Jure Leskovec, and Bernhard Sch¨olkopf. Modeling information propagation with survival theory. In *International Conference on Machine Learning*, pages 666–674. IMLS, 2013.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[He and Niyogi, 2004] Xiaofei He and Partha Niyogi. Locality preserving projections. In *Advances in neural information processing systems*, pages 153–160, 2004.

[Hong *et al.*, 2011] Liangjie Hong, Ovidiu Dan, and Brian D. Davison. Predicting popular messages in twitter. In *International Conference on World Wide Web*, pages 57–58, 2011.

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

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

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

[Pinto *et al.*, 2013] Henrique Pinto, Jussara M. Almeida, and Marcos A. Goncalves. Using early view patterns to predictthe popularity of youtube videos. In *ACM International Conference on Web Search and Data Mining*, pages 365– 374, 2013.

[Ribeiro et al., 2017] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R. Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 385–394. ACM, 2017.

[Salvador and Chan, 2007] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[Wang *et al.*, 2016] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225– 1234. ACM, 2016.

[Yu *et al.*, 2015] Linyun Yu, Peng Cui, Fei Wang, Chaoming Song, and Shiqiang Yang. From micro to macro: Uncovering and predicting information cascading process with behavioral dynamics. In *2015 IEEE International Conference on Data Mining*, pages 559–568. IEEE, 2015.

[Zhang *et al.*, 2013] Jing Zhang, Biao Liu, Jie Tang, Ting Chen, and Juanzi Li. Social influence locality for modeling retweeting behaviors. In *International Joint Conference on Artificial Intelligence*, pages 2761–2767, 2013.

[Zhao *et al.*, 2015] Qingyuan Zhao, Murat A. Erdogdu, Hera Y. He, Anand Rajaraman, and Jure Leskovec. SEISMIC: a self-exciting point process model for predicting tweet popularity. *Computer Science*, pages 1513–1522,2015.