Data Science and Engineering (2020) 5:1–11
https://doi.org/10.1007/s41019-020-00117-1

Deep Reinforcement Learning-Based Approach to Tackle Topic-Aware Influence Maximization

Shan Tian[1] · Songsong Mo[1] · Liwei Wang[1] · Zhiyong Peng[1]

Received: 7 November 2019 / Revised: 31 January 2020 / Accepted: 17 February 2020 / Published online: 28 February 2020

Abstract
Motivated by the application of *viral marketing*, the topic-aware influence maximization (TIM) problem has been proposed to identify the most influential users under given topics. In particular, it aims to find k seeds (users) in social network G, such that the seeds can maximize the influence on users under the specific query topics and diffusion model such as independent cascade (IC) or linear threshold (LT). This problem has been proved to be NP-hard, and most of the proposed techniques suffer from the efficiency issue due to the lack of generalization. Even worse, the design of these algorithms requires significant specialized knowledge which is hard to be understood and implemented. To overcome these issues, this paper aims to learn a generalized heuristic framework to solve TIM problems by meta-learning. To this end, we first propose two topic-aware social influence propagation models based on IC and LT model, respectively, which is conducive to better advertising injections. We then encode the feature of each node by a vector and introduce a model, called *deep influence evaluation model* , to evaluate the user influence under different circumstances. Based on this model, we can construct the solution according to the influence evaluations efficiently, rather than spending a high cost to compute the exact influence by considering the complex graph structure. We conducted experiments on generated graph instances and real-world social networks. The results show the superiority in performance and comparable quality of our framework.

Keywords Social network · Influence maximization · Graph embedding · Reinforcement learning

1 Introduction

With the popularity of social networks [26, 33], more and more people tend to share information by the word of mouth in their daily life. As a result, social network has become a new platform for *viral marketing*, which is used to promote products, innovations and opinions. Motivated by these applications, Kempe et al. [15] first formalize the seeds selection in viral marketing as a discrete optimization problem, which is known as influence maximization (IM). However, recent studies [9, 14, 20] have argued that the generic IM is not topic-aware and cannot be directly applied to online advertising. For example, imagine that we are looking for spokesmen for new sneakers, we will pay more attention to sports star than singing star. Thus, there have been many efforts [2, 9, 14, 22, 23] extending generic IM to topic-aware IM (TIM) to support more personalized and accurate advertising.

In general, there are two ways to solve TIM [21]. The first way is *IM for topic-relevant targets*, which considers that the *user* is topic-aware, and wants to maximize the influence on topic-relevant users. These studies mainly focus on differentiating users and compute the influence on activated users by their benefits under specific query topics. The second way is *IM for topic-dependent diffusion*, which formalizes that the *edges* are topic-aware, and wants to maximize the influence under a new diffusion model. These studies [9, 11] focus on proposing new diffusion models which can capture the dynamic probabilities under different query topics. The most commonly used model associates each edge with a probability under each topic and calculates the propagation

✉ Zhiyong Peng
peng@whu.edu.cn

Shan Tian
tianshan14@whu.edu.cn

Songsong Mo
songsong945@whu.edu.cn

Liwei Wang
liwei.wang@whu.edu.cn

[1] School of Computer Science, Wuhan University, Wuhan, Hubei, China

probability under each query as the expected summation of related probabilities.

However, the real scenario is that people have different probabilities to be activated in different topics, while the probability of people influencing each other changes with topics, so neither of these two ways can solve TIM problems solely, not to mention that they are more or less inadequate. For example, the design of all these algorithms requires specialized human knowledge, i.e., the dynamic programming method, and trial-and-error, which can be very hard to be understood. Moreover, they suffer from the efficiency issues. On the one hand, most current state-of-the-art solutions [23, 27] to solve the first branch problems need to generate hundreds of thousands sample sets randomly to select the most influential user online or offline, time-consuming or space-consuming. On the other hand, methods for the second branch [8, 11] struggle with the enormous computation about the potential diffusion probabilities. What's more, regardless of the high cost, problems with same structure but varied only in data require new sample processes performed again and again.

In this paper, we establish a greedy heuristic framework to learn the algorithm instead. This data-driven method avoids the traditional complex algorithm designing processes by adopting a combination of graph embedding and reinforcement learning. On the one hand, the embedding method considers both users' preferences and dynamic influence probabilities, which can cover both two branches of TIM and then encode the graph to a vector space. On the other hand, the framework builds a neural network called *deep influence evaluation model* to estimate the influence of each candidate under different circumstances and adopts *reinforcement learning* method to train it. Contrary to the traditional methods, the main advantage of our *deep influence evaluation model* is that it can be trained offline while stored with very little disk space. Based on the model outputs, the framework uses a greedy policy to construct a feasible solution quickly which avoids the heavy online influence computation. This mechanism assures our framework can be generalized to different problem instances without new training processes.

More specifically, our contributions can be summarized as follows:

- To solve the TIM comprehensively, we propose a data-driven approach to assign the probabilities based on both IC and LT models. By considering both users' interests and relationships, we define three metrics of user similarity, user benefit and inherent contact frequency which can solve the two branches of TIM simultaneously.
- We propose a new *graph embedding* network, called *Diffusion2Vec* which can extract features for each user in social network automatically, capturing the properties of each user according to graph structure and user's own attributes.
- Based on the embedding network, we define an estimation model called *deep influence evaluation model* (*DIEM*) which is used to calculate the influence of candidate users according to their embeddings, the current partial solution and the query topics. Based on the influence, a feasible solution can be constructed by greedy node selection and addition.
- We adopt an algorithm of reinforcement learning, called *double DQN with prioritized experience replay* to train models. The main advantage of this algorithm is that it can deal with overestimating and delayed reward in a data-efficient way. The training process is set up in such a way that the policy will aim to maximize the targeted influence under each query *directly*.
- We conduct experiments on generated graph instance and real-world social network from *Twitter*. The results show the superiority in performance and comparable solution quality of our framework.

2 Problem Definition

In this section, we will define the topic-aware influence maximization (TIM) problem and give a rough sketch of our greedy framework.

2.1 TIM problem

To facilitate our presentation, all frequently used notations are listed in Table 1. TIM introduces *topics* to describe both information characteristics and users' interests and calculated the influence based on not only the seed set S but also the query topics τ. It focuses on maximizing the *targeted influence* over users who are relevant to the query topics under specific diffusion models.

Let τ denote the query topics and $\sigma_G(S|\tau)$ denote the targeted influence spread by the seeds S in an instance of the influence propagation process on graph G. Intuitively, the TIM problem finds a seed set S^* with k users to maximize the targeted influence spread by a seed set S over social network G and can be formally defined as follows:

Definition 1 (*TIM*) Given a graph $G(V, E)$, a positive integer k and a series of targeted topics τ, TIM selects a set S^* of k nodes from V as seed set to maximize the *targeted influence* $\sigma_G(S^*|\tau)$, i.e., $\sigma_G(S^*|\tau) = \arg\max_{S\subseteq V\wedge|S|=k}\sigma_G(S|\tau)$.

2.2 Greedy Framework

We will focus on the popular pattern for designing approximation and heuristic algorithms, namely greedy algorithm.

Table 1 Frequent notations used across the paper

Notation	Meaning
G, V, E	The social network, the vertex set, the edge set
$\tau, k, \widehat{S}, S$	The query topics, the size of seed set, the seed set, current solution set
$u_v^{(t+1)}$	The node embedding of v at $t+1$ iteration
X_v	The properties of node v, including the tag indicted whether it has been selected into S and user profile
p_{uv}^{τ}	The probability of node u successively activate node v under query topics τ
Q	The function to evaluate the influence of each user
$\widehat{Q}$	The estimation model to estimate the influence of each user
$\sigma_G(S^*\|\tau)$	The maximum targeted influence spread among all seed set with size k for social graph G under query topics τ
$\sigma_G(S\|\tau)$	The targeted influence spread by S for social graph G under query topics τ

The greedy framework will construct a solution by sequentially adding one node to the partial solution S, based on some evaluation function Q which measures the influence of each candidate in the context of the current partial solution and the query topics. Specifically:

1. A partial solution is represented as an ordered list $S = (v_1, v_2, \ldots, v_{|s|}), v_i \in V$, and $S = \emptyset$ at the start. $\overline{S} = V \backslash S$ denotes the set of candidate nodes for addition, conditional on S. Furthermore, we use the first element of a vector to represent whether the node is selected into S, i.e., $X_{v[0]} = 1$ if $v \in S$ and $= 0$ otherwise.
2. The quality of a partial solution S can be calculated as the targeted influence $\sigma(S|\tau)$ of S.
3. The generic greedy framework selects a node v to be added according to which v maximized the evaluation function, $Q((v, S)|\tau) \in R$. Then, the partial solution S will be extended as $S := (S, v^*)$, where $v^* := \arg\max_{v \in \overline{S}} Q((v, S)|\tau)$. This step is repeated until the size of solution set achieved k.

Recently, deep learning has promoted the development of computer vision and speech recognition. Based on feeding sufficient data into the deep neural networks, it's possible to learn better representations than hand-crafted features [17]. Thus, we want to design a powerful deep function Q to estimate the influence of each candidate under certain circumstances.

3 Graph Representation

As mentioned above, the *deep influence evaluation model* ($\widehat{Q}$) must evaluate the influence for each candidate under different circumstances. Intuitively, $\widehat{Q}$ should summarize the state of the partial solution and figure out the influence of each candidate in the context of such a solution and query topics. Here, both the state of the solution and the context of a node v can be very complex, hard to be described in closed form. In order to represent such complex context over social networks, we will leverage a deep learning architecture called *Diffusion2Vec* to parameter model $\widehat{Q}$.

In this section, we first introduce our two topic-aware models based on IC and LT, respectively, and then give a thorough explanation about our embedding model: *Diffusion2Vec* and estimation model: $\widehat{Q}$.

3.1 Social Network Mining

As we have discussed in the Introduction, the first branch of TIM ignores the dynamic diffusion probabilities dependent on query topics while the second does not consider associating the probability with regard to users' interests, similarities and contact frequencies at the same time. So neither of these two branches can solve TIM problems completely.

Thus, here we propose three metrics motivated by user-based collaborative filtering algorithm to merge the two branches together. First, to model a social network as an online advertising platform, we extend each node in G to be associated with a user profile represented by a weighted vector. For example, for a user v who is very interested in sports, music but not in politics, his profile may be described as {<music,0.7>, <sport,0.3>, <politics,0.0>}, where 0.7 called his *benefit* under the *music* topic ($B_v^{\tau_1}$). Then, we calculate the sum of benefits related to the query topics to distinguish the targeted users. Next, we define the *similarity* (*sim*) between two users by the *cosine similarity* between their profile vectors. In addition to the user similarity and user benefit, we believe that the initial weight in the network can symbolize the frequency of contact between users. We combine these three metrics with the IC and LT model to obtain topic-aware IC model and topic-aware LT model, respectively. It is worth noting that both our models adopt a data-driven method to calculate the important parameters like $p_{u,v}$ and θ_v rather than assigning random values from [0, 1], which makes more sense to be close to real life.

3.2 Topic-Aware IC Model

Independent cascade (IC) is a classic and well-studied propagation model. It considers a user v is activated by each of its incoming neighbors independently by introducing an influence probability $p_{u,v}$ to each edge $e = (u, v)$. Based on the influence probabilities and given a seed set S at time step 0, a diffusion instance of the IC model unfolds in discrete steps. Each active user u in step t will activate each of its outgoing neighbor v that is inactive in step $t - 1$ with probability $p_{u,v}$. This diffusion instance terminates when no more nodes can be activated.

Next, we calculate the propagation probability that a user u successfully activate user v with the three metrics: the initial edge weight w_{uv} which is initialized in (0,1] which stands for the contact frequency; the similarity between two users; the sum benefits of user v toward the query topics τ. Formally, the propagation probability between user u and user v under topics τ can be defined as:

$$p^{\tau}_{uv} = (\gamma_1 w_{uv} + \gamma_2 sim(u, v) + \gamma_3 B^{\tau}_{v})/3, \tag{1}$$

where $\gamma_1, \gamma_2, \gamma_3 \in (0, 1)$ and they are used to balance the weights of contact frequency, user similarities and targeted benefits. Since all the value of three factors is in range (0,1], the propagation probability p^{τ}_{uv} is restricted to range (0,1] and can then be used in the standard IC model for computing the influence spread. The targeted influence is finally computed as the summation of benefits of the related users who are activated according to the new diffusion probabilities.

3.3 Topic-Aware LT Model

Linear threshold (LT) is also a seminal diffusion model, which is introduced by Granovetter and Schelling in 1978. The basic idea of LT is that a user can switch its status from inactive to active if a "sufficient" number of its incoming neighbors are active. Formally, in the LT model, each edge $e = (u, v)$ is associated with a weight $b_{u,v}$. Let $N_I(v)$ be the set of incoming neighbors of user v, and it satisfies that $\sum_{u \in N_I(v)} b_{u,v} \leq 1$. Moreover, each user v is also associated with a threshold θ_v. Considering an instance of the diffusion process, the LT model first samples the value of θ_v of each user v uniformly at random from [0, 1]. Then, it proceeds in discrete steps. In step 0, it sets the status of users in S as active and others as inactive. Then, it updates the status of each user iteratively: In step t, all users that were active in step $t - 1$ remain active, and any user v that was inactive in step $t - 1$ switches to active if the total weight of its active neighbors in $N_I(v)$ is at least θ_v. The diffusion instance terminates when no more user is to be activated.

Based on the traditional LT model, we assign the $\theta_v = 1 - \gamma_3 B^{\tau}_{v}$ which considers that users are easier to be activated when they are more interested in the products. Then, we calculate the $b_{u,v}$ as $b_{u,v} = (\gamma_1 w_{uv} + \gamma_2 sim(u, v))$ considering the more similar and closely related two people, one of them has a greater impact on the other. Like in the *topic − aware IC model*, $\gamma_1, \gamma_2, \gamma_3 \in (0, 1)$ and they are also used to balance the weights between the three metrics. Since all the value of three factors are in range (0,1], the propagation probability p^{τ}_{uv} is restricted to range (0,1], and can then be used in the standard LT model for computing the influence spread. In this case, the targeted influence is finally computed as the summation of benefits of the related users who are activated according to the new diffusion model.

3.4 Embedding: *Diffusion2Vec*

Since graph is an important data representation and traditional graph analysis methods suffer from high computation and space overhead, the graph embedding technique has been put forward to solve the graph analysis problem effectively and efficiently. It related to two aspects of researches, i.e., graph analytics [13] and representation learning [4, 5, 7]. It converts graph data into low dimensional vectors while the graph structure and other useful information are preserved.

For example, *Structure2Vec* [16] embeds each node $v \in V$ into a p-dimensional feature space thus each node v can be easily described as a p-dimensional vector u_v and the graph can be represented by the set of these vectors.

However, due to the lack of consideration about node property and information diffusion characteristics, it can not be directly used in our problem settings. So we proposed a new method named *Diffusion2Vec* which considers both the users' attributes and the dynamic propagation probabilities.

Specifically, the *Diffusion2Vec* will initialize the embedding of each node $u^{(0)}_v$=0 and for all $v \in V$ update the embeddings synchronously at each iteration as

$$u^{(t+1)}_v \leftarrow F(X_v, \{u^{(t)}_u\}_{u \in N(v)}, \{p(u, v)\}_{u \in N(v)}; \Theta). \tag{2}$$

X_v denotes a vector which contains useful user information. More specifically, the first element of X_v, $X_{v[0]}$ corresponds to a node-specific tag which means whether the node is selected so far, i.e., $X_{v[0]}$=1 if $v \in S$ and = 0 otherwise. The rest part of X_v, named *benefit vector*, correspond to the profile we mentioned at Sect. 3.1. For example, for a user v with profile {<music,0.7>, <sport,0.3>, <politics,0.0>} and hasn't been selected into S, the X_v will be $\{0, 0.7, 0.3, 0.0\}$. $N_{(v)}$ denotes the neighbors of node v in the graph G. $p(u, v)$ corresponds to probability in the *topic-aware model*. F

Table 2 Instantiation of reinforcement learning components for TIM

State	Action	Reward	Termination
Sequence of selected nodes	Add a node to S	The marginal influence	$\|S\| = k$

denotes a nonlinear mapping such as neural network or kernel function.

Based on the update formula, one can see that the embedding of node v can be seen as a combination of its own features and the aggregated impact of other nodes. In other words, the node features can be propagated to other nodes via the nonlinear propagation function F and the more update iterations were carried on, the farther away the features will be propagated and get aggregated at distant nodes. Finally, the update process will terminate until the number of iterations reaches a predefined parameter: T. At that time, each node embedding $u_v^{(T)}$ will contain information about its T-hop neighborhoods as determined by graph topology, the involved node features. It is worth noting that when the X_v degenerates to a binary scalar, this form of node embedding can be used to solve the generic IM problems.

Next, we will design the powerful *deep influence evaluation model* to evaluate the influence of node with these vector representations.

3.5 Deep Influence Evaluation Model ($\widehat{Q}$)

Specifically, we adopted the neural network as the mapping function in *Diffusion2Vec*; then the embedding can be defined as :

$$u_v^{(t+1)} \leftarrow relu(\theta_1 X_v + \theta_2 \sum_{u \in N(v)} u_u^{(t)} + \theta_3 \sum_{u \in N(v)} relu(\theta_4 p(u, v))),$$

where $\theta_1 \in \Re^{p \times (|\tau|+1)}$, $\theta_2, \theta_3 \in \Re^{p \times p}$ and $\theta_4 \in \Re^p$ are the embedding parameters and relu is the rectified linear unit which is widely used as an activation function in artificial neural networks.

More specifically, we use the embedding $u_v^{(T)}$ for node v and the pooled embedding over the partial solution S, $\sum_{u \in S} u_u^{(T)}$ as the substitution for v and S, respectively. Finally, the estimation model $\widehat{Q}$ can be defined as:

$$\widehat{Q}(S, v; \Theta, \tau) = \theta_5 relu([\theta_6 \sum_{u \in S} u_u^{(T)}, \theta_7 u_v^{(T)}]), \tag{3}$$

where $\theta_5 \in \Re^{2p}, \theta_6, \theta_7 \in \Re^{p \times p}$ and [.,.] is the concatenation operator. The number of iterations T for the graph embedding computation is usually small, such as $T = 4$.

What deserves our attention is the parameter set Θ. Since the model $\widehat{Q}$ is based on the embedding $u_v^{(T)}$, the parameter set Θ is then a collection of 7 parameters actually. That is $\Theta = \{\theta_i\}_{i=1}^7$ and all these parameters must be learned. However, due to the lack of training labels, we will learn these parameters *end-to-end* with reinforcement learning, which will be explained thoroughly in the following section.

4 Parameters Learning

This section shows how to learn the parameters Θ in $\widehat{Q}$. Note that the definition of $\widehat{Q}$ lends itself to an RL formulation and thus can be used as the state-action value in RL directly. Moreover, the state-of-the-art RL algorithms [24, 25] will provide strong supports for learning the parameters.

4.1 RL Formulation

Based on the introduction to RL in [29], we show the instantiations of RL components for TIM in Table 2 and define the states, actions, rewards, policy in the RL framework for TIM as follows:

1. *States*: a state is a sequence of actions (nodes) added into the solution set. In other words, the current state equals to the current solution set S. It is easy to see that the *terminal state* comes when the size of S reaches k.
2. *Actions*: an action corresponds to a node in G which have not been added into the solution set S (the tag of $X_{v[0]} = 0$). Since we have represented each node v as a p-dimensional vector, here the action is a p-dimensional vector too.
3. *Rewards*: the reward (after taking action v) is defined as the marginal targeted influence spread by the user v under the query topics. That is, $r(S, v) = \sigma((S \cup \{v\})|\tau) - \sigma(S|\tau)$. As such, the cumulative rewards of a *terminal state* can be described by the seed set $\widehat{S}$, as $R(\widehat{S}) = \sum_{i=1}^k r(S_i, v_i)$ which equals to $\sigma(\widehat{S}|\tau)$.
4. *Policy*: based on the *deep influence evaluation model* $\widehat{Q}$, we will adopt a deterministic greedy policy $\pi(v|S, \tau) := \arg\max_{v' \in \overline{S}} \widehat{Q}(v', S|\tau)$. Selecting action v corresponds to add node v of $\overline{S}$ to the partial solution S, which results in a new state S' and a reward $r(S, v)$.

4.2 Learning Algorithm

We use *double deep Q-networks* (DDQN) [31] with *prioritized experience replay* [28], which can be generalized to

work with large-scale function approximation to perform end-to-end learning of the parameters Θ in $\widehat{Q}(S, v; \Theta, \tau)$. We use the term *episode* to represent series operations of node addition starting from an empty set $\emptyset$, and until termination; a *step* within an episode is a single action (node addition). This approach will update the parameters with a batch of weighted samples from a dataset E, rather than the single sample being currently experienced. The dataset E is populated during previous episodes with an interval, called *n-step*.

The advantage of *prioritized experience replay* is that the model can be learned more effectively. An experience with high expected learning progress measured by the magnitude of its temporal-difference (TD) error will be replayed more frequently to learn more efficiently. And the DDQN has been shown to reduce the overestimations effectively. Both the two methods lead to much better performance when using a neural network as a function approximation [31], a property that also applies for our model $\widehat{Q}$. The *prioritized experience replay* replaces the sampling methods used by DDQN with stochastic prioritization and importance sampling as illustrated in Algorithm 1.

Algorithm 1 Double DQN with prioritized experience replay

Input:
minibatch b, step-size η, replay period B and capacity N, exponents α and β, budget L.

Initialize replay memory H=$\emptyset$, Δ=0, $p_1 = 1$
Observe S_0 and choose $A_0 \sim \widehat{Q}(S, v; \Theta, \tau)$
for $t = 1$ to L **do**
  Observe $S_t, \mathrm{R}_t, \gamma_t$
  Store transition $(S_{t-1}, A_{t-1}, R_t, \gamma_t, S_t)$ in H with maximal priority $p_t = \max_{i<t} p_i$
  if $t \equiv 0 \bmod B$ **then**
    for $j = 1$ to b **do**
      Sample transition $j \sim P(j) = p_j^\alpha / \sum_i p_i^\alpha$
      Compute importance-sampling weight $w_j = (N \cdot P(j))^{-\beta} / \max_i w_i$
      Compute TD-error $\delta_j = R_j + \gamma_j Q_{t\,\mathrm{arg\,et}}(S_j, \arg\max_a Q(S_j, a)) - Q(S_{j-1}, A_{j-1})$
      Update transition priority $p_j \leftarrow |\delta_j|$
      Accumulate weight-change $\Delta \leftarrow \Delta + w_j \cdot \delta_j \cdot \nabla_\theta Q(S_{j-1}, A_{j-1})$
    end for
    Update weights $\theta \leftarrow \theta + \eta \cdot \Delta$, reset Δ=0
    From time to time copy weights into target network $\theta_{t\,\mathrm{arg\,et}} \leftarrow \theta$
  end if
  Choose action $A_t \sim \pi_\theta(S_t)$
end for

5 Experiment

In this section, we study the performance of *DIEM* on generated instances and real-world dataset with different diffusion models. First, we reveal the details of experimental settings. Then, we give an introduction to the baseline solutions [23]. Finally, we report the experimental results and give an explanation. The methods are implemented with C++ and Python and run on a CentOS server (Intel i7-7700 3.60 GHz CPU with 8 cores GPU with GeForce GTX 1080Ti and 64 GB RAM). It cost 3 and 72 h to train the models for one topic and five topics, respectively.

5.1 Experimental Setup

Instance Generation We generate graph instances to evaluate the proposed method against existing state-of-the-art solution WRIS [23]. We first generate graphs according to Barabasi-Albert (BA) rule[1] which have been used to model many real-world networks. Specifically, for a given range on the number of nodes, e.g., 1M-10M, we sampled the number of nodes uniformly at random in that range and then generate a graph according to BA. Finally, we associate each node with a regularized vector generated randomly (the length of which equals the total number of query topics) to simulate users' profiles.

Real-World Dataset To test the performance on real scenario, we use a dataset collected from *Twitter* for performance evaluation. This dataset contains 41.6 million users with 476 millions tweets [18]. We extract 50 topics from it and user profile is represented by a weighted term vector in the topic space. The vector is generated by aggregating all the tweets related with the same user into a giant document and using the LDA model[1] to mine users' preferences in the topic space. To test the scalability with increasing number of users, we sampled 10M, 20M, 30M, 40M users from the dataset.

Queries The two factors that make up a query are topics and seed set size. For the topics, we use queries from AOL search engine.[2] Given 200 topics defined in advance, we first filter the topic queries and retain those only containing our topics manually. For the seed set size, we adopt $k = 30$ for initial estimates on generated instances and vary the number of k from 10 to 50 on *twitter* for further evaluation. It is worth noting that the framework does not need to be retrained until the whole topic space is changed.

Parameter Settings The hyperparameters and parameter settings used in our experiments are shown in Table 3. For our method, we simply tune the hyperparameters on small

[1] https://github.com/kenneth-orton/TwitterLDATopicModeling.
[2] https://www.aolsearch.com/.

Table 3 Main configurations used in experiments with default values highlighted

Datasets	Users	Edges	Q topics	k
Generated	1M–2M,...,9M–10M	40–80M,...,360–400M	1	30
Twitter	10M, 20M, 30M, **40M**	0.7B, 1.1B, 1.2B, **1.3B**	1, 2, **3**, 4, 5	10, 15, ..., **30**, ..., 50
γ_1	γ_2	γ_3	Batch size	N-step
0.8	0.8	1	64	3

graphs (i.e., the graphs with less than 50 nodes), and fix them for large graphs.

5.2 Baseline Solutions

To solve the TIM problems, [23] has proposed a weighted sampling technique (WRIS) based on RIS and achieved an approximation ratio of $(1 - 1/e - \varepsilon)$. However, because the method needs to generate hundreds of thousands of random sample sets and requires intensive computation overhead, they further proposed two disk-based solutions, RR and IRR, to support the online query process. The idea is to put the sampling procedure from online to offline and build index on the random sample sets for each topic. During query processing, RR directly loads all the related random sample sets into memory and uses the greedy algorithm to find the top-k seed users. IRR further improves over RR by incrementally loading the most promising RR sets into memory and adopts the top-k aggregation strategy to save computation costs.

Since the baseline methods adopt IC model to simulate the information diffusion process. As for our new topic-aware IC model, we recalculate the activation probabilities and feed these values together with users' properties into the original frameworks. As for the topic-aware LT model, we extend the framework with a mapping table to record the θ_v of each user and modify the code of propagation according to the rules of LT as we mentioned in Sect. 3.1. As for the parameters, we set ϵ to 0.1, K to 100 and the partition size δ for IRR to 100 as they are the most accurate settings adopted in [23] for all experiments.

Generally, we use WRIS as a baseline solution because it uses an online sampling and thus has an effective assurance. Since IRR outperforms RR in the *Twitter* dataset as they reported, we adopt IRR as a baseline to compare the efficiency. That is, we compare the method based on deep learning (*DIEM*) with WRIS and IRR to evaluate the quality by expected targeted influence and efficiency by average running time and disk space, respectively.

In the following experiments, we evaluate the performance on generated graph instances and *Twitter*, respectively.

5.3 Experiments on Generated Instance

Comparison of Effectiveness

The graph embedding framework enables us to train and test on graphs of different sizes, since the same set of model parameters are used. How does the performance of the learned algorithm using small graphs generalize to test graphs of larger sizes? To investigate this, we train *DIEM* on graphs with 0.5M-1M nodes and test its generalization ability on graphs with up to 10M nodes regarding one topic and 30 seeds. The results are shown in Table 4.

We can see that *DIEM* achieves a very good performance compared with *WRIS* under either topic-aware IC model or topic-aware LT model. As the graph size increases to 10M, there are still only subtle differences between the targeted influences they produced, which mainly come from the exploration and utilization in RL. That is to say, our framework can be generalized to larger graph instances without new training processes.

Discussion of Different Diffusion Models As we can see from Table 4, the total influence spread under the topic-aware IC model is a little higher than the topic-aware LT model. This is probably because each node has more chances of random activation in the topic-aware IC model, while each node has only one calculation chance in the topic-aware LT model.

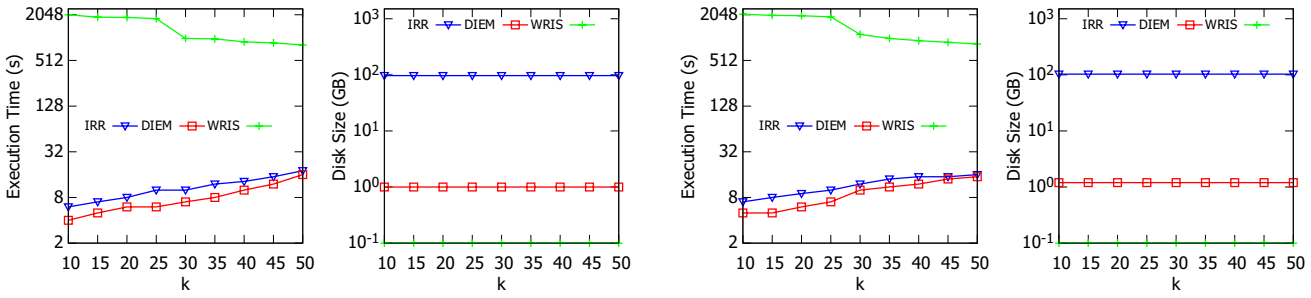
5.4 Experiments on Real-World Dataset

In the following experiments, we evaluate the efficiency and effectiveness of our proposed framework on real-world

Table 4 Targeted influence spread when varying test graph size

Test size	1M–2M	3M–4M	5M–6M	7M–8M	9M–10M
WRIS (IC)	372.1	472.7	548.2	668.4	792.3
DIEM (IC)	375.6	478.3	556.3	673.9	798.6
WRIS (LT)	330.5	422.5	488.7	609.8	692.4
DIEM (LT)	335.0	433.8	496.0	613.4	699.6

Values are average *targeted influence* over 10 test instances



(a) IC model **(b)** LT model

Fig. 1 Varying the seed set size: k

Table 5 Targeted influence spread when varying k under IC and LT model

k	WRIS (IC)	IRR (IC)	DIEM (IC)	WRIS (LT)	IRR (LT)	DIEM (LT)
10	8674	8674	8674	7988	7978	7983
15	10667	10665	10666	10014	10011	10007
20	12089	12099	12093	11238	11250	11233
25	13101	13097	13098	12307	12315	12330
30	13930	13931	13925	13017	13024	13022
35	14619	14616	14617	13987	14003	13998
40	15210	15211	15209	14632	14654	14643
45	15732	15735	15730	15039	15041	15045
50	16212	16214	16215	15823	15854	15833

dataset collected from *Twitter* w.r.t increasing seed users k, graph size $|V|$ and query topics $|\tau|$, respectively.

5.4.1 Vary the Seed Set Size

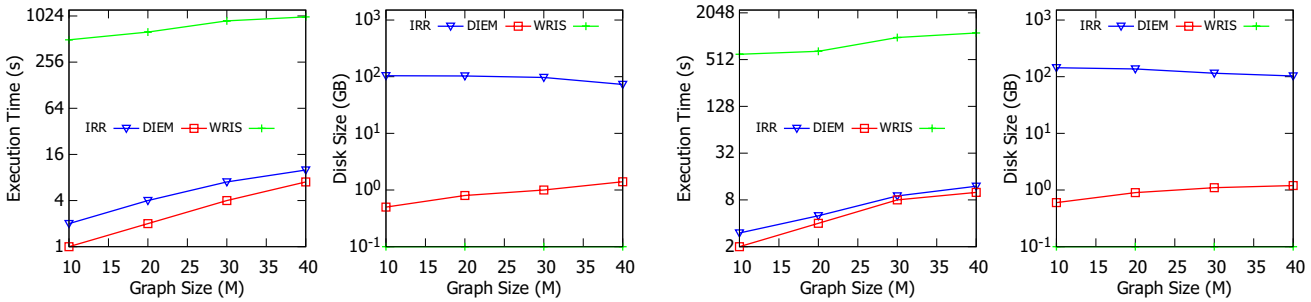
Comparison of Efficiency We first examine the performance with increasing seed set size k. The running time and disk cost are shown in Fig. 1. It is obvious that the methods based on deep learning architecture (*DIEM*) and disk index (*IRR*) are significantly faster than the online sampling method (*WRIS*) under either IC or LT model. The average response time to a query using *DIEM* and *IRR* are 160x and 120x times faster than *WRIS* under IC model. However, *DIEM* takes 100x times less disk space than *IRR*. This is because *IRR* needs to store abundant precomputed sampling sets under each topic while *DIEM* only needs to maintain the model parameters.

As k increases, it takes a slightly longer time for both *DIEM* and *IRR* methods to answer a query under the two models. However, *IRR* still spends a little more time than *DIEM* under both models. This is because both methods need more iterations to find the seed users, causing more computations but *IRR* costs more disk I/O than *DIEM*. To our surprise, the performance of *WRIS* is slightly faster as k increases. The reason is that the performance of *WRIS* is mainly dependent on the number of RR sets generated, which is inversely proportional to the optimal targeted influence. However, the optimal targeted influence is directly proportional to k.

Comparison of Effectiveness In addition to the result that *DIEM* and *IRR* are much faster than *WRIS*, the solutions they generated are also not worse than *WRIS*. We report the *targeted influence* of seed sets returned by all the methods and models in Table 5. There is almost no difference between all the methods. Thus, the *targeted influence* spread for all methods will not be presented in the rest of the experiment section, as the results show similar patterns.

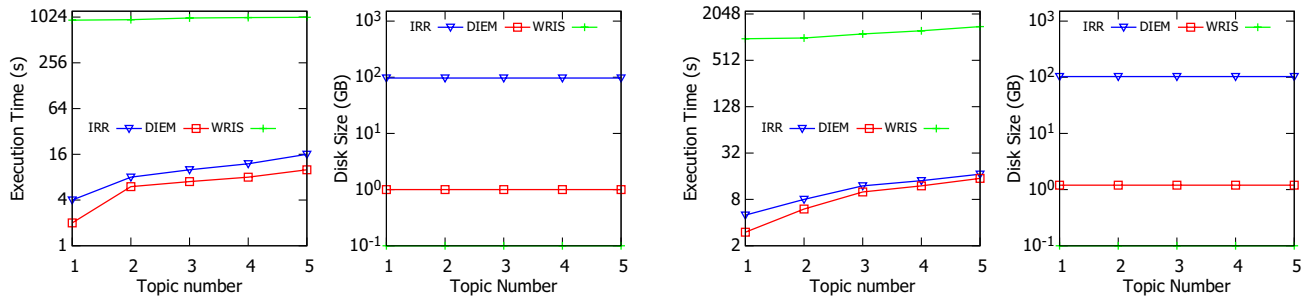
5.4.2 Vary the Graph Size

Comparison of Efficiency We increase the graph size, i.e., $|V|$, to test the scalability of our proposed architecture. The results are shown in Fig. 2. The methods under both IC and LT models need more time to generate a solution when the graph becomes larger because more sampling or computation need to be processed. Nevertheless, *DIEM* and *IRR* clearly outperform *WRIS* by great margins in all scenarios. However, *DIEM* still takes much less disk space than *IRR*. This is because, as the graph size grows, there are more *RR* sets needed to be built for the *IRR* index while *DIEM* only needs to maintain more parameters in neural networks. It



(a) IC model **(b)** LT model

Fig. 2 Varying the graph size: $|V|$



(a) IC model **(b)** LT model

Fig. 3 Varying the query topics number: $|\tau|$

shows that *DIEM* is more effective for larger graphs without compromising its performance superiority.

5.4.3 Vary the Number of Query Topics

Comparison of Efficiency When we vary the number of query topics $|\tau|$ from 1 to 5, as shown in Fig. 3, the results demonstrate similar pattern: *DIEM* and *IRR* are at least two orders of magnitude better than *WRIS* in a social network with millions of users under both models. However, the solution found by *IRR* takes similar or longer time than our *DIEM*, but is still of worse disk cost.

6 Related Work

IM problem is an NP-hard problem and has been extensively studied. Approaches to tackling the generic IM problem have mainly two flavors: approximation algorithm and heuristics. Kempe et al. [15] proposed the first simple greedy algorithm with an approximation ratio of $(1 - 1/e - \varepsilon)$. Since then, there has been a lot of research devoted to improving the efficiency while keeping the theoretical bound [6, 12, 19]. Although CELF [19] and CELF++ [12] have significantly improved the running time, the methods were only examined in small graphs with thousands of vertexes. RIS [6] uses random sampling technique and is the first method scalable enough to handle graphs with millions of vertexes. The heuristics on IM improved the efficiency by discarding the theoretical bound. For example, Chen et al. [10] used vertex degree as a selection criterion. They propose that fine-tuned heuristics may provide truly scalable solutions to IM with satisfying influence spread and blazingly fast running time.

The above methods cannot be directly applied in online advertising because the same seeds are returned for different advertisements. To solve this problem, the topic-aware IM (TIM) was proposed in [2, 23]. Inflex [2] first precomputes a number of top-k seed sets offline and process the online query by finding nearest neighbors among the neighbor approximation. Li et al. [23] adopted a weighted sampling technique based on RIS to make sure the targeted users relevant to the advertisement have a higher probability to be sampled. To meet the real-time processing requirement, they further devise two disk-based index structures to push the sampling procedure from online to offline. However, the solution requires nearly 100 GB disk size to store the sampling sets on handling 5 topics and a graph with 40 million users. Moreover, new sampling RR sets need to be generated when the method runs on a new graph. Tian at al. [30] proposed a learning approach

to tackle the topic-aware influence maximization, but they only consider the probabilities under IC models.

Besides, there has been some seminal work on using deep architectures to learn heuristics for combinatorial problems on graphs recently [3, 32]. However, the architectures used in these works are generic, not yet effectively reflecting the combinatorial structure of graph problems. In [16], the authors proposed a unique combination of reinforcement learning and graph embedding to incrementally learn heuristic algorithms for traditional problems over graphs like the minimum vertex cover, maximum cut and traveling salesman problems. Since there are more complex structures in the topic-aware influence maximization problem settings, such as user profiles and dynamic diffusion probabilities, we extend the vector representation by incorporating useful information and adopt a more advanced algorithm, namely *double DQN with prioritized experience replay* to train the larger models.

Compared with existing topic-aware IM solutions, our greedy heuristic framework behaves like a meta-algorithm which is driven by data instead of designing the complex algorithm. Our solution not only retrieves seed users in a few seconds with graph with millions of nodes while costs very little disk size, but also can be generalized to solve different graph instances under the same problem structure.

7 Conclusion

We established a machine learning framework for automatically designing greedy heuristics for topic-aware influence maximization (TIM) problems. The point of our approach is the combination of deep graph embedding with reinforcement learning. Besides, we proposed two diffusion models to capture the characteristics in advertisements spread. Through extensive experiments on generated graph instances and real-world social network, we demonstrate the effectiveness of the proposed framework as compared to manually-designed TIM algorithms. Moreover, the learned heuristics can be generalized to solve different TIM problems, i.e., TIM of larger graph size, TIM on different graph instances under the same topics with all excellent performance.

Acknowledgements This work is supported by the National Key Research and Development Program of China (Project Number: 2018YFB1003400), and the Fundamental Research Funds for the Central Universities (Project Number: 2042017kf1017).

References

1. Albert R, Barabási A (2001) Statistical mechanics of complex networks. CoRR arXiv:cond-mat/0106096
2. Aslay Ç, Barbieri N, Bonchi F, Baeza-Yates RA (2014) Online topic-aware influence maximization queries. In: Proceedings of the 17th international conference on extending database technology, EDBT 2014, Athens, Greece, March 24–28, 2014, pp 295–306
3. Bello I, Pham H, Le QV, Norouzi M, Bengio S (2017) Neural combinatorial optimization with reinforcement learning. In: 5th international conference on learning representations, ICLR 2017, Toulon, France, April 24–26, 2017, workshop track proceedings
4. Bengio Y, Courville AC, Vincent P (2013) Representation learning: a review and new perspectives. IEEE Trans Pattern Anal Mach Intell 35(8):1798–1828
5. Bordes A, Usunier N, García-Durán A, Weston J, Yakhnenko O (2013) Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems 26: 27th annual conference on neural information processing systems 2013. Proceedings of a meeting held December 5–8, 2013, Lake Tahoe, Nevada, United States, pp 2787–2795
6. Borgs C, Brautbar M, Chayes JT, Lucier B (2012) Influence maximization in social networks: towards an optimal algorithmic solution. CoRR arxiv:1212.0884
7. Cai H, Zheng VW, Chang KC (2018) A comprehensive survey of graph embedding: problems, techniques, and applications. IEEE Trans Knowl Data Eng 30(9):1616–1637
8. Chen S, Fan J, Li G, Feng J, Tan K, Tang J (2015) Online topic-aware influence maximization. PVLDB 8(6):666–677
9. Chen W, Lin T, Yang C (2015) Real-time topic-aware influence maximization using preprocessing. In: Proceedings of 4th international conference computational social networks, CSoNet 2015, Beijing, China, August 4–6, 2015, pp 1–13
10. Chen W, Wang Y, Yang S (2009) Efficient influence maximization in social networks. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, Paris, France, June 28–July 1, 2009, pp 199–208
11. Fan J, Qiu J, Li Y, Meng Q, Zhang D, Li G, Tan K, Du X (2018) OCTOPUS: an online topic-aware influence analysis system for social networks. In: 34th IEEE international conference on data engineering, ICDE 2018, Paris, France, April 16–19, 2018, pp 1569–1572
12. Goyal A, Lu W, Lakshmanan LVS (2011) CELF++: optimizing the greedy algorithm for influence maximization in social networks. In: Proceedings of the 20th international conference on world wide web, WWW 2011, Hyderabad, India, March 28–April 1, 2011 (companion volume), pp 47–48
13. Grover A, Leskovec J (2016) node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, San Francisco, CA, USA, August 13–17, 2016, pp 855–864
14. Guo J, Zhang P, Zhou C, Cao Y, Guo L (2013) Personalized influence maximization on social networks. In: 22nd ACM international conference on information and knowledge management, CIKM'13, San Francisco, CA, USA, October 27–November 1, 2013, pp 199–208
15. Kempe D, Kleinberg JM, Tardos É (2003) Maximizing the spread of influence through a social network. In: Proceedings of the ninth

ACM SIGKDD international conference on knowledge discovery and data mining, Washington, DC, USA, August 24–27, 2003, pp 137–146
16. Khalil EB, Dai H, Zhang Y, Dilkina B, Song L (2017) Learning combinatorial optimization algorithms over graphs. In: Advances in neural information processing systems 30: annual conference on neural information processing systems 2017, 4–9 December 2017, Long Beach, CA, USA, pp 6351–6361
17. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems 25: 26th annual conference on neural information processing systems 2012. Proceedings of a meeting held December 3–6, 2012, Lake Tahoe, Nevada, United States, pp 1106–1114
18. Kwak H, Lee C, Park H, Moon SB (2010) What is twitter, a social network or a news media? In: Proceedings of the 19th international conference on world wide web, WWW 2010, Raleigh, North Carolina, USA, April 26–30, 2010, pp 591–600
19. Leskovec J, Krause A, Guestrin C, Faloutsos C, VanBriesen JM, Glance NS (2007) Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining, San Jose, California, USA, August 12–15, 2007, pp 420–429
20. Li G, Chen S, Feng J, Tan K, Li W (2014) Efficient location-aware influence maximization. In: International conference on management of data, SIGMOD 2014, Snowbird, UT, USA, June 22–27, 2014, pp 87–98
21. Li Y, Fan J, Wang Y, Tan K (2018) Influence maximization on social graphs: a survey. IEEE Trans Knowl Data Eng 30(10):1852–1872
22. Li Y, Fan J, Zhang D, Tan K (2017) Discovering your selling points: personalized social influential tags exploration. In: Proceedings of the 2017 ACM international conference on management of data, SIGMOD conference 2017, Chicago, IL, USA, May 14–19, 2017, pp 619–634
23. Li Y, Zhang D, Tan K (2015) Real-time targeted influence maximization for online advertisements. PVLDB 8(10):1070–1081
24. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller MA (2013) Playing atari with deep reinforcement learning. CoRR arxiv:1312.5602
25. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller MA, Fidjeland A, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
26. Mo S, Tian S, Wang L, Peng Z (2019) Minimizing the spread of rumor within budget constraint in online network. In: Sun X, He K, Chen X (eds) Theoretical computer science. Springer, Singapore, pp 131–149
27. Nguyen HT, Dinh TN, Thai MT (2016) Cost-aware targeted viral marketing in billion-scale networks. In: 35th annual IEEE international conference on computer communications, INFOCOM 2016, San Francisco, CA, USA, April 10–14, 2016, pp 1–9
28. Schaul T, Quan J, Antonoglou I, Silver D (2015) Prioritized experience replay. CoRR arxiv:1511.05952
29. Sutton RS, Barto AG (1998) Reinforcement learning—an introduction. Adaptive computation and machine learning. MIT Press, Cambridge
30. Tian S, Zhang P, Mo S, Wang L, Peng Z (2019) A learning approach for topic-aware influence maximization. In: Web and big data—third international joint conference, APWeb-WAIM 2019, Chengdu, China, August 1–3, 2019, Proceedings, Part I, pp 125–140
31. van Hasselt H, Guez A, Silver D (2016) Deep reinforcement learning with double q-learning. In: Proceedings of the thirtieth AAAI conference on artificial intelligence, February 12–17, 2016, Phoenix, Arizona, USA, pp 2094–2100
32. Vinyals O, Fortunato M, Jaitly N (2015) Pointer networks. In: Advances in neural information processing systems 28: annual conference on neural information processing systems 2015, December 7–12, 2015, Montreal, Quebec, Canada, pp 2692–2700
33. Zhang P, Bao Z, Niu Y, Zhang Y, Mo S, Geng F, Peng Z (2019) Proactive rumor control in online networks. World Wide Web 22(4):1799–1818