Lightweight Influence-Aware News Recommendation in Social Media

Yuting Feng[1], Bogdan Cautis[2]

[1]Department of Electrical and Computer Engineering (ECE), National University of Singapore
[2]Infocomm Technology Cluster, Singapore Institute of Technology

Abstract

News recommendation systems are generally based on the semantic content of news items and user profiles, whereas the underlying recommendation scenario is ignored. With the increasing popularity of social media as pathways to news, making personalized news recommendations in specific social scenarios deserves attention, where the information diffusion patterns and influence mechanisms therein are exploited. We consider in this paper a diffusion and influence-aware perspective on the news recommendation problem, and we propose a *lightweight deep learning* approach for it, called LISM. This approach targets news recommendation in micro-blogging platforms, such as Twitter or Weibo, whose extreme *data velocity* demands a satisfactory trade-off between the model's complexity and its effectiveness. LISM is a content-based deep recommendation model that seeks to represent news and users, while also taking into consideration the observed news diffusions (cascades). We use graph embeddings – node representations that are indicative of news diffusion patterns – leading to valuable social-related information for recommendations. To merge the semantics and social-related representations of news, a specially designed convolutional neural network for joint feature representation (SCNN) is used as the news encoder, while an attention model automatically aggregates the different interests of users. We conduct experiments on two real-world datasets, showing that LISM outperforms the state-of-the-art news recommendation approaches, while exhibiting a good complexity vs. effectiveness tradeoff.

1 Introduction

The tremendous growth of the Web and e-commerce has led to the development of recommender systems, benefiting from a rich research literature in recent years [40]. Many techniques have been proposed in various application scenarios, from traditional collaborative filtering to deep learning-based approaches. In essence, they reason about the relationships between users and items (such as music, books, movies, and news), to identify the items that users might be most interested in and to generate personalized recommendations based on them. Recommender systems are nowadays ubiquitous on the Web, as one of the main engines for its success. One particular scenario that has become highly popular is the one of *online news recommendation*, which raises major specific challenges, pertaining to the highly dynamic, short-lived nature of news.

At the same time, with the advent of social networks, a lot of information – including news – spreads online, generating interest in the analysis of influence and information diffusion, under the formal scope of *influence estimation* [14, 22, 42] and *influence maximization* [25, 30]. While the main application of information diffusion studies in social networks is viral marketing, there are other and quite diverse potential applications, among which recommender systems [27].

In the social media context, it is natural to revisit the "matching" problem by taking into consideration social links and interactions, observing and exploiting how the information that is up for recommendation may also spread in an underlying network through *word-of-mouth* mechanisms. On one hand, such

mechanisms – e.g., likes, shares, reposts / retweets, notifications – enable information to propagate rapidly and touch upon large audiences. On the other hand, social media brings *credibility* to the messages that are being conveyed, as recent studies show that we are more inclined to pay attention to a referral from a *friend* or from a known *influencer* [3].

Naturally, this renders influence and information diffusion important dimensions for any recommendation problems in social media, and even more so in the case of news. Indeed, social media is an increasingly popular platform for news dissemination and consumption, accounting now for about half of the market share [23] (reaching even 65% in some countries). Importantly, both the style of news and the patterns of news consumption have been reshaped by social media.

In the *micro-blogging* context, illustrated by hugely popular platforms such as Twitter and Sina Weibo, one may adopt news items not only based on content and preferences, but also based on the perceived influence from others with respect to news adoption. Such influence may be *local*, i.e., directly exerted by friends or followed users, or *global*, i.e., indirectly exerted by some influential users (a.k.a. "social whales" in the social marketing jargon).

The state-of-the-art ML-based news recommendation approaches (e.g., [13, 46, 51, 63]), which can be labelled as influence or social-agnostic, are generally exploiting the semantic content of news items and the user profiles, with some recent studies adopting a sequential perspective on the users' adoption patterns. Similar to the recent work of [11], we adopt in this paper an influence-aware perspective on news recommendation, with the main goal of *finding a sweet spot between the model's complexity and its effectiveness*. It is well-known that about half a billion tweets are published daily on Twitter [10], leading to billions of daily engagements. Therefore, we believe that only by achieving such a practically relevant complexity vs. effectiveness trade-off we can handle such unprecedented rates at which news posts and engagements thereof occur in micro-blogging applications.

For performance reasons and in order to achieve such an acceptable trade-off, we do not model the problem as a sequential recommendation task, shedding light on the fact that news adoptions in micro-blogging do exhibit temporal diversity, a finding that is similar to the one of the recent study of [54], on online news recommendation.

Starting from a joint history of observed news adoptions and news propagations (cascades) in a micro-blogging network, we propose a deep learning approach which follows a content-based architecture, while also taking into consideration the propagations thereof. Our main contributions are the following:

- We propose a lightweight deep learning-based model, called LISM, designed for news recommendation in social scenarios, where the susceptibility of a target user to news depends on both its semantics and the social impact it carries.
- We use information diffusion patterns to estimate the impact a piece of news may have on the target user, by leveraging the participation of users in news dissemination (the news cascades) and by representing users through graph embeddings.
- We endow news items with diffusion-related information (users involved in diffusion) besides semantics, by a specially designed CNN-based approach (SCNN) to align and fuse multi-source embeddings for joint feature extraction.
- We perform extensive experiments on two real-world datasets (including a publicly available one), confirming that (i) news diffusion patterns can significantly improve the effectiveness of news recommendation and, (ii) the proposed model strikes a balance between complexity and effectiveness, being therefore applicable in the highly dynamic contexts that motivate our work.

2 Related Work

2.1 Deep Learning in Recommender Systems

Early research on recommender systems is mainly about collaborative filtering (CF) [44] or content-based models [35]. The CF models make recommendations based on known user-item pairs, but they usually face cold-start problems, while the content-based mdoels leverage attribute information about users and items. Hybrid models [26] combine the benefits of the two directions.

With the increase in data volumes that recommender systems have to process, traditional models face problems such as computation cost, sparsity, and scalability. To tackle them, deep learning-based models were used recently, leading to many state-of-the-art ideas (e.g., the pioneering work on Restricted Boltzmann Machines [43]), with many studies aiming to capture complex, high-order dependencies between users and items via deep learning models [4, 19].

Among them, some CNN-based approaches are used as feature extractors from unstructured data – text [8], video [24], music [5], etc – while Recurrent Neural Networks (RNNs) are used in sequential modeling to observe the dynamic evolution of user preferences [7] or in session-based recommendation tasks [20].

2.2 Graphs in Recommender Systems

Graph analysis has been systematically explored by researchers to get insights on the relations and interactions between entities. Some of the most common tasks for graph analysis are (i) link prediction / recommendation [31], (ii) node classification [2], (iii) clustering [41], or (iv) visualization [1].

Generally, graphs are either described by an adjacency matrix or are projected (embedded) into a low-dimensional space [16]. The interest of the latter is that it gives an elegant way to transpose discrete data into a differentiable space, where machine learning techniques can apply for downstream analysis, among which recommendation task. With the increasing attention received by graph neural networks (GNNs) in recent research, many works [48, 57, 60] seek to exploit the effectiveness of GNNs in recommender systems. Such models incorporate both the graph structure and node attributes to represent users and items, by fusing neighboring node embeddings.

2.3 Social-Aware Recommender Systems

The underlying idea of recommendation algorithms in social media is to exploit social concepts such as homophily (attraction to similarity) and influence (actions and behavior evolution due to social connectivity). We discuss next such works, among which a few are focusing on news. Some of the pioneering social-aware models incorporate the social links, as indicators of similarity in collaborative filtering [18, 32]. Deep neural networks (and in particular GNNs) have also been used in the social networking scenario, to aggregate social-related information. E.g., the work of [9] builds user-item and user-user graphs, merging user-item interactions and social neighbourhoods for recommendation. In the DICER approach of [12], the authors exploit a dual-side deep context, which leverages social connectivity for modeling both user interests and item attractiveness. Beyond the first-order social neighbourhood interest of a user-user graph, DiffNet++ [56] models also the recursive diffusion process in the graph, under social influence.

Few works have taken steps in the direction of influence-aware news recommendation. The early work of [38] analyzed news in social media for recommendation, but their *Friends-Rank* strategy is to select and directly serve tweets from the user's friends, therefore limiting its scope drastically. More recently, [59] uses Monte Carlo Tree Search to mine high-order connectivity among users, and proposes a multi-armed bandit algorithm for selecting social friends for user representations.

Diffusion patterns or propagation cascades of news are not considered for user modelling in these works. In comparison, going beyond social similarity, we aim to exploit how information items may be diffused and adopted by socially interconnected users. We use a deep *behavior-driven* network to represent users as influencers (posters, i.e., news cascade initiators) or influencees (reposters, i.e., news cascade participants).

2.4 Personalized News Recommendation

News recommendation is a rather particular task in recommender systems because of the timeliness, diversity, and heterogeneity of news [28]. In addition, the explosion of new media content creators, besides the traditional media ones (newspapers, agencies, etc), makes personalized recommendation even more challenging. Following the success of deep learning in NLP, some of the models derived from information retrieval [33], for comparing query and document similarity, have been transferred to news recommendation, focusing on the highly condensed language in news. In state-of-the-art personalized news recommendation models, researchers generally focus on user representations based on user-item interactions (clicked news), using attention mechanisms to aggregate diverse interests, based on users' history (adopted news titles [49]) and news content [50]. Additional information such as news categories (by either a multi-level attention mechanism to fuse information in [51], or by a multi-head self attention mechanism in [52]), dwell time [50], shares or likes [55] are also used to represent news. In addition, with the development of general-purpose knowledge graphs and knowledge embeddings, discovering news connections based on external knowledge connections has been a recent trend in news recommender systems [46, 63].

In [53], the authors explore the benefits of *pre-trained language models* (PLMs), such as BERT [6], to mine the deep semantic information of news. PLMs are shown to have stronger text modeling ability than shallow models that are built directly from the news corpus. In a similar vein, the work of [62] learns news representations by pre-training with BERT, proving the generalization ability of this approach in cold-start scenarios.

Following the success of GNNs, recent studies also adopt a graph perspective on users and news, building for instance a bipartite graph where users and news are both represented as nodes, and neighboring nodes are respectively aggregated in the representation of news or users [13]. Similarly, [21] builds a user-news-topic graph to represent long-term interests of users.

All these deep learning news recommendation approaches traditionally focus on the news and adoptions thereof, ignoring the underlying social dimension to news.

The very recent work of [11] showed that, when it comes to news recommendation in a social networking scenario, relevant relations among users are revealed not only by common clicked news, but also by implicit influence. They propose a deep learning approach that is tailored for news recommendation in micro-blogging, modeled as a *sequential recommendation task*. Building on the assumption that users tend to adopt similar news successively – a prevalent assumption in the news recommendation literature – their sequential model incorporates temporal aspects, such as attractiveness and timeliness.

We adopt in this paper an influence-aware perspective on news recommendation, similar to [11]. Accordingly, LISM exploits jointly a history of observed news adoption and news propagation (diffusion cascades), bridging the content-based modeling with representations learned from the cascades, capturing local and global influence patterns on news. However, our model follows different architectural choices, which lead to a solution that is arguably on par with the one of [11] in terms of effectiveness, while being much more efficient. We believe that reduced training complexity is paramount in the micro-blogging context, which is one the most dynamic mediums for user-generated online content. By design, our method is not only less computationally intensive, but also less demanding in terms of input data

dimensions for training, and therefore more versatile. Indeed, the temporal / sequential details of news adoptions may not always be (readily) available for the predictive model, therefore requiring approaches that do not depend on them. Finally, our work also provides similar insights and conclusions on online news adoption as [54], which shows that a sequential modeling of the news recommendation problem may be suboptimal.

3 Social Graph Representations

To model the influence exerted among users in social graphs, we build upon prior work on analyzing social interactions and representing users, in the context of a recommendation task. We review here the embedding methods used in our work to extract node representations in a social graph. A graphical illustration of the node representations discussed in this section is provided in Figure 1.

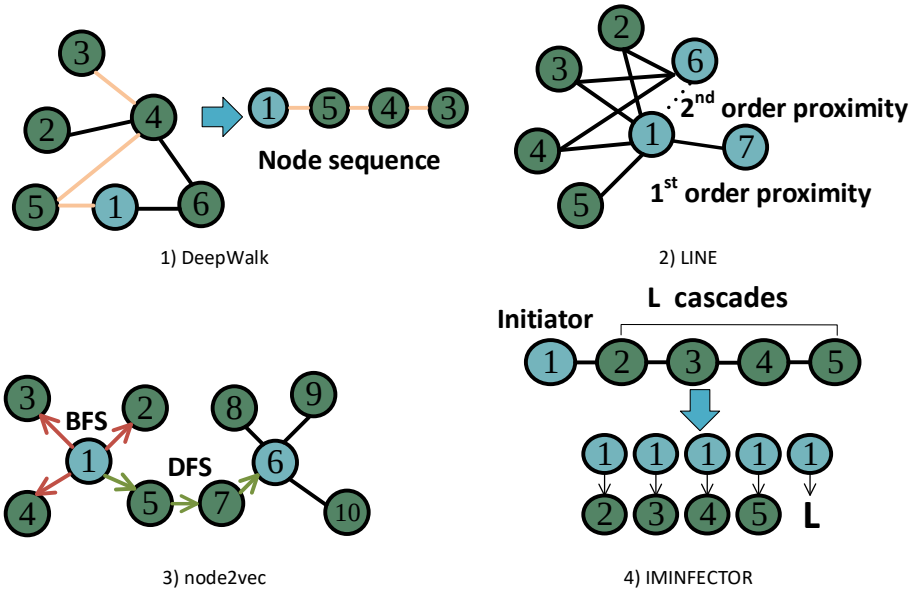

Figure 1: Topology or influence based node embeddings.

For a weighted social graph $G = (V, E, \omega)$, where V are the nodes (users), E are the edges, and ω weights each edge, a graph embedding corresponds to a mapping $\phi : V \to \mathbb{R}^g$, representing each node by a g-dimensional vector that preserves certain graph relationships and similarities. We review the following methods:

- *DeepWalk* [37] generates node embeddings by maximizing the probability of observing the last κ nodes and the next κ nodes in a random walk of length $2\kappa + 1$ centering at node v_i. With the node sequence $S_{v_i} = \{v_{i-\kappa}, ..., v_{i-1}, v_{i+1}, ..., v_{i+\kappa}\}$, the objective is

$$\min_\phi[-\log\Pr(S_{v_i}|\phi(v_i))] \tag{1}$$

  with $\phi(v_i) = \boldsymbol{v}_i$ the g-dimensional embedding of v_i.

- *LINE* [45] uses the adjacency matrix and embeddings to represent jointly probability distributions for each node pair, by the minimized KL-divergence of distributions. For 1st order proximity, the two distributions and objective functions are

$$\hat{p_1}(v_i, v_j) = \omega_{ij}/\Omega \tag{2}$$

$$p_1(v_i, v_j) = 1/[1 + \exp(- < \phi(v_i), \phi(v_j) >] \tag{3}$$

$$O_1 = - \sum_{(i,j) \in E} \omega_{ij} \log p_1(v_i, v_j), \tag{4}$$

where $\Omega = \sum_{(i,j) \in E} \omega_{ij}$ is the sum of all weights in the adjacency matrix $\mathbf{\Omega} \in \mathbb{R}^{|V| \times |V|}$ of G, and $\phi(v_i) = \boldsymbol{v}_i$, $\phi(v_j) = \boldsymbol{v}_j$ are the node embeddings of v_i and v_j. The distributions and objective function for 2nd order proximity are

$$\hat{p_2}(v_j|v_i) = \omega_{ij}/\delta_i \tag{5}$$

$$p_2(v_j|v_i) = \frac{\exp(\phi(v_i), \phi(v_j))}{\sum_{k=1}^{|V|} \exp(\phi(v_k), \phi(v_i))} \tag{6}$$

$$O_2 = - \sum_{(i,j) \in E} \omega_{ij} \log p_2(v_j|v_i), \tag{7}$$

where δ_i is the out-degree of node v_i.

- *node2vec* [17] extends *DeepWalk* to weighted, directed graphs, introducing biased random walks, based on Depth-First Search (DFS) and Breadth-First Search (BFS) strategies. Hyper-parameters μ_1, μ_2 are used to set the search bias β:

$$\beta_{\mu_1 \mu_2}(v_j, v_i) = \begin{cases} \frac{1}{\mu_1}, & \text{if} \quad d_{ji} = 0 \\ 1, & \text{if} \quad d_{ji} = 1 \\ \frac{1}{\mu_2}, & \text{if} \quad d_{ji} = 2 \end{cases} \tag{8}$$

where d_{ji} is the shortest path distance between nodes v_j and v_i. Combined with edge weights $w_{(i-1)i}$, the unnormalized transition probability is

$$\pi_{v_{i-1} v_i} = \beta_{\mu_1 \mu_2}(v_j, v_i) \cdot \omega_{(i-1)i} \tag{9}$$

The ith node in the sequence is generated by

$$P(v_i|v_{i-1}) = \begin{cases} \frac{\pi_{v_{i-1} v_i}}{\Gamma}, & \text{if} \quad (v_i, v_{i-1}) \in E \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

with Γ being a normalizing constant.

Besides the aforementioned *topology-based* graph representation learning, some novel deep learning-based approaches addressing graph-related tasks in an end-to-end way (*GNN models*) have attracted increasing attention recently [58]. As our focus goes beyond structural relationships in the social graph, to the influence a user – albeit distant - may exert on another one and its potential impact on news adoptions, we also consider the *cascade-based* IMINFECTOR approach of [34], which proposes a deep learning model for node representations of *influencer* and *influencee* (susceptible) users.

For the graph embedding task, IMINFECTOR takes a set of cascades – time-order sequence of timestamped adoptions in the form of $[(v_1, t_1), ..., (v_j, t_j), ..., (v_{m_x}, t_{m_x})]$ – where v_1 is the cascade initiator, all other v_j nodes are the influenced users in the propagation process, and m_x denotes the length of the cascade for item x. A time-sensitive sampling is used to select influenced nodes from cascades as training data. The hidden layer output is set as $\boldsymbol{z}_{t,i} = \boldsymbol{O}_i \boldsymbol{T}^T + \boldsymbol{b}_t$, for influencer v_i and target user v_t. $\boldsymbol{O} \in \mathbb{R}^{I \times g}$

provides the influencer (cascade initiator) embeddings, $\boldsymbol{T} \in \mathbb{R}^{|V| \times g}$ provides the target (susceptible) user embeddings, and $\boldsymbol{b}_t$ is the bias. I is the number of cascade initiators in the training data. The output is

$$\hat{\boldsymbol{y}}_t = \frac{e^{\boldsymbol{z}_{t,i}}}{\sum_{v_{i'} \in V} e^{\boldsymbol{z}_{t,i'}}} \tag{11}$$

with $\boldsymbol{y}_t$ being the one-hot representation of target node v_t to minimize the cross-entropy loss function

$$L_t = \boldsymbol{y}_t \log(\hat{\boldsymbol{y}}_t). \tag{12}$$

We will use the trained target node embeddings matrix $\boldsymbol{T}$ to represent users in the news diffusion network.

4 Problem Formulation

In an online, micro-blogging context, news adoptions are no longer observed by clicks, but instead by posting (tweeting) or reposting (retweeting) actions. The input for our news recommendation approach comes in the form of a social network and a cascade history, which also gives us the adoptions.

A news item will in general be first posted by a source, the cascade initiator, and then it will be adopted (posted) by other users involved in that diffusion process, the cascade participants. When a cascade reaches a user, she may become aware of who posted initially and who adopted that news item before, increasing her awareness about it and thus swaying the adoption decision.

With a social graph representation where each user is endowed with an embedding vector, news items can be described as records $x = \{W_x, V_x\}$, where W_x is the semantic information of the news content, and V_x represents the social-related information. Semantic information W_x is composed of a list of words in news titles $W_x = [w_1, w_2, ..., w_{n_x}]$, with n_x the number of words in the news title. (For simplicity and computation efficiency we only take titles into consideration here, but this can be easily generalized to include the news abstract / content.) Let V_x denote the list of involved users $V_x = [v_1, v_2, ..., v_{m_x}]$, obtained from the diffusion cascade of news item x, $L_x = [(v_1, t_1), \ldots, (v_{m_x}, t_{m_x})]$, which is a time-ordered sequence of timestamped adoptions by nodes (users), with $\{v_1, \ldots, v_{m_x}\} \subseteq V$, m_x being the number of users involved in the propagation of x, and v_1 being the cascade initiator.

With the representation for each news in the social scenario, a user i's adoption history in LISM will be represented as a set $\{x_1, x_2, ..., x_{s_i}\}$, where each x_j represents a news item adopted by user i, and s_i the total number of news items adopted by user i. For a user i with a known adoption history, our recommendation task is to predict the probability that i will click on some candidate news x_k.

5 LISM for News Recommendation

We present in this section the overall framework of LISM, followed by details on how we compute node embeddings while preserving the structural relationships and accounting for the observed diffusion patterns, then the design of SCNN for multi-source embeddings of words and nodes, and finally the application of an attention mechanism to aggregate the users' history.

5.1 LISM Framework

LISM takes a candidate news and a target user's history as input, and outputs the corresponding adoption probability. The structure of LISM is shown in Figure 2, and each component is detailed next, passing bottom-up over this framework.

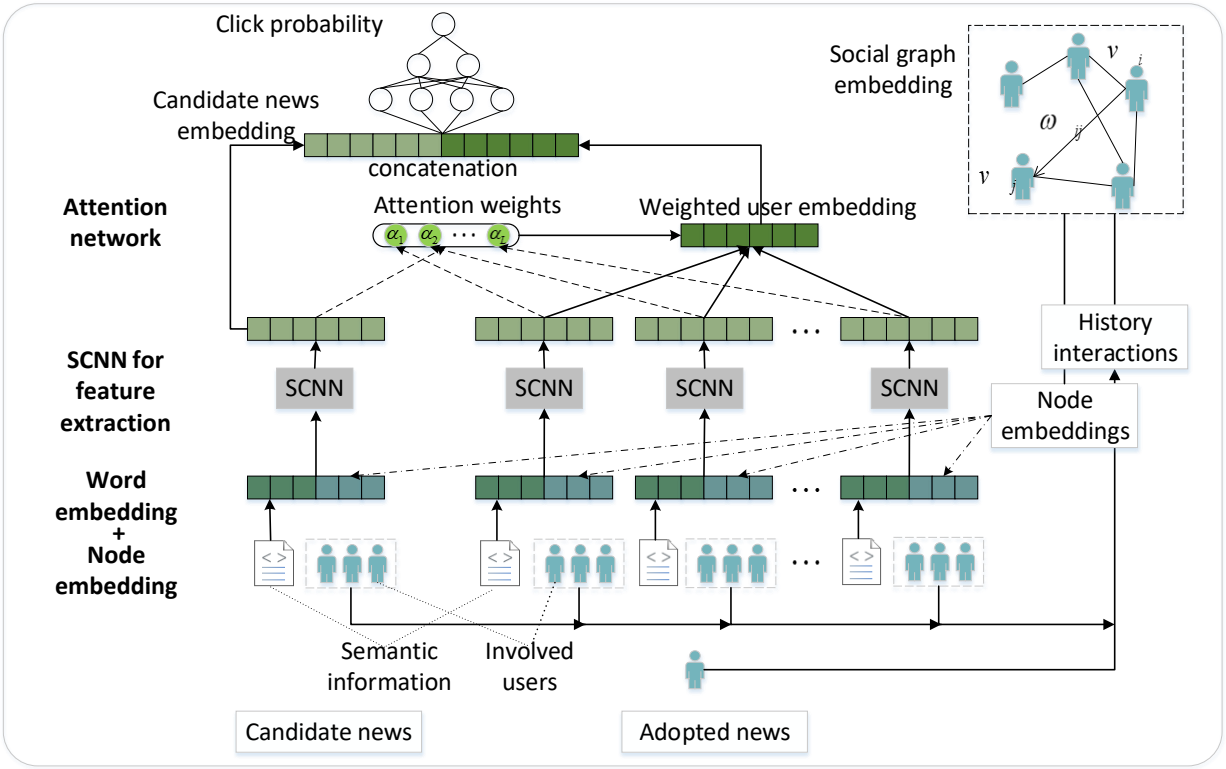

Figure 2: LISM framework.

In LISM a news item is an embedding matrix, composed of the news title represented as a word sequence $\boldsymbol{W}_x = [\boldsymbol{w}_1, \boldsymbol{w}_2, ...\boldsymbol{w}_{n_x}]^T \in \mathbb{R}^{n_x \times f}$, and the users in the cascade sequence represented as $\boldsymbol{V}_x = [\boldsymbol{v}_1, \boldsymbol{v}_2, ...\boldsymbol{v}_{m_x}]^T \in \mathbb{R}^{m_x \times g}$. $\boldsymbol{w}$ is the word embedding vector of w, and $\boldsymbol{v}$ is the node embedding vector of v, f and g being the embedding dimensions for word and node vectors respectively. The word vectors are pre-learned from a large corpus, while the node vectors are the output of a pre-trained node embedding model.

The process of user node embeddings in the social graph was discussed previously. The weights of edges in social graph and the feature extraction with SCNN will be detailed in the following two sections respectively. The representations of the candidate news and of the user's adopted news provided by SCNN are fed into an attention-based network, to allocate weights to each piece of history w.r.t. the candidate news. This weighted user history embedding and candidate news embedding are then concatenated as the input to a Deep Neural Network (DNN), to get the click probability.

5.2 Influence-Aware Social Graph

To endow news items with diffusion-related information (users involved in diffusion), besides semantics, we can either use a cascade-based diffusion network to project V into a vector space, as $\phi : V \to \mathbb{R}^g$, by topology-based embedding techniques designed for weighted, directed graphs, such as node2vec [17], or alternatively, we can directly use the cascades to obtain embeddings that capture the influencer / susceptible representation of nodes, as in [34].

For the former alternative, as in [15], we start by building a *diffusion graph* (V, E, ω) from the initial social graph $G = (V, E)$ and the cascades. For $(v_i, v_j) \in E$, we look at how often tweets from v_i are

retweeted by v_j. More precisely,

$$\omega_{ij} = \frac{R_{i2j}}{\sum_{v_k} R_{k2j}} \tag{13}$$

where ω_{ij} is the influence v_i has on v_j, R_{i2j} is the number of times v_j reposts from v_i, and v_k iterates over all the nodes that v_j has reposted from. In this way, the structural relationship and historical interactions among nodes are both present in (V, E, ω); for simplicity, hereafter G will denote this diffusion graph.

Alternatively, the work of [34] focuses on influence maximization based on cascades solely, using cascade-based node embeddings. Since the focus of our paper is not on influence maximization, but on leveraging the diffusion patterns to capture a user's adoption likelihood, we can pre-train such an embedding model on the news cascades, leading to node representations $\boldsymbol{v}_i, i \in V$.

A comparison over the impact on the prediction task of the topology-based node representations and the cascade-based ones will be detailed in the experiments section. We stress here that only the latter are time-dependent, but these can be pre-trained on a relevant – yet not necessarily the latest – history of cascades. In fact, the frequency by which the node embedding layer is re-trained can be seen as knob on the efficiency vs. effectiveness tradeoff.

5.3 SCNN for Social-Related Multi-Source Feature Extraction

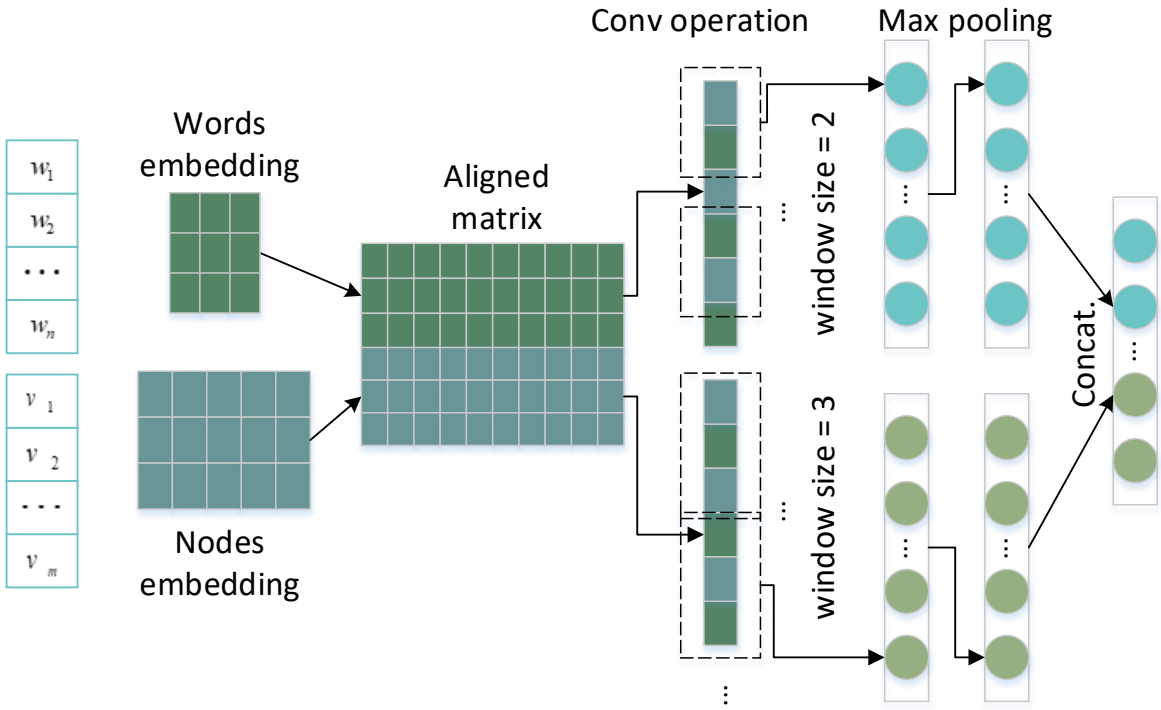

Figure 3: SCNN for word and node feature extraction.

Inspired by the similarity between node embeddings and word embeddings [37], we design a special CNN for news encoding in order to (i) align node embeddings with word embeddings, eliminating the heterogeneity of vector spaces, and (ii) fuse and extract features for joint representation of multi-source embeddings (Figure 3).

Following the notations from Section 4, with pre-trained node embeddings $\boldsymbol{v}_i$ obtained as in Section 5.2, and word embedding pre-learned from a large corpus, the representation embedding of news x for

both semantic and diffusion-related information is

$$\boldsymbol{X} = \{\boldsymbol{W}_x, \boldsymbol{V}_x\} = \{\boldsymbol{w}_1\boldsymbol{w}_2...\boldsymbol{w}_{n_x}\boldsymbol{v}_1\boldsymbol{v}_2...\boldsymbol{v}_{m_x}\} \tag{14}$$

with $\boldsymbol{w} \in \mathbb{R}^f$ and $\boldsymbol{n} \in \mathbb{R}^g$. For a generic x, we simplify notation hereafter writing n and m instead of n_x and m_x. To align the word and node vectors, a transformation function ρ is applied on node vectors to map the node embeddings into the word space, while preserving the original node relationships, leading to

$$\boldsymbol{E} = [\boldsymbol{w}_1\boldsymbol{w}_2...\boldsymbol{w}_n\rho(\boldsymbol{v}_1)\rho(\boldsymbol{v}_2)...\rho(\boldsymbol{v}_m)] \tag{15}$$

where $\boldsymbol{E} \in \mathbb{R}^{(m+n)\times f}$ and the transformation function being either the linear one $\rho(\boldsymbol{v}) = \boldsymbol{M}\boldsymbol{v}$ or the non-linear one $\rho(\boldsymbol{v}) = \sigma(\boldsymbol{M}\boldsymbol{v}) + \boldsymbol{b}$, with $\boldsymbol{M} \in \mathbb{R}^{f\times g}$ the trainable transformation matrix, $\boldsymbol{b} \in \mathbb{R}^f$ the trainable bias, and σ either the sigmoid function or the tanh one.

SCNN with multiple filters extracts features from the multi-source embedding matrix $\boldsymbol{E}$. Each $\boldsymbol{h} \in \mathbb{R}^{f\times l}$ filter, with varying window size l, is applied to a sub-matrix $\boldsymbol{E}_{i+l-1}$ and a new feature is generated as

$$c_i^h = f(\boldsymbol{h} \cdot \boldsymbol{E}_{i:i+l-1} + \boldsymbol{b}) \tag{16}$$

with $i = 1, 2, \ldots, n + m - l + 1$ and the bias $\boldsymbol{b} \in \mathbb{R}^f$. A feature map $[c_1^h, c_2^h, \cdots, c_{n+m-l+1}^h]$ is obtained going through the embedding matrix $\boldsymbol{E}$. After a max-over-time pooling operation on the feature map, the largest feature is chosen as

$$c_{max}^h = \max\{c_1^h, c_2^h, \cdots, c_{n+m-l+1}^h\} \tag{17}$$

For multiple filters, the final representation of news, comprising both word and node / diffusion information, is the concatenation of the largest feature from each filter $\boldsymbol{h}^i$, for $i = 1, \ldots, \gamma$, as follows:

$$\boldsymbol{e} = [c_{max}^{h_1}, c_{max}^{h_2}, \cdots, c_{max}^{h_\gamma}] \tag{18}$$

5.4 Attention Mechanism for User Interests

Assuming a user i with history of adopted news $\{x_1^i, x_2^i, \cdots, x_{s_i}^i\}$, after SCNN for joint feature extraction, the final representation is written as $\{\boldsymbol{e}_1^i, \boldsymbol{e}_2^i, \cdots, \boldsymbol{e}_{s_i}^i\}$. The same SCNN is applied on the candidate news k, leading to representation $\boldsymbol{e}_k$.

Considering that a user may have a wide range of interests, and that previously adopted news pertain to various topics of interest, instead of systematically taking an average of all embeddings $\boldsymbol{e}^i = \frac{1}{s_i}\sum_{j=1}^{s_i}\boldsymbol{e}_j^i$ of adopted news, an attention network is used to characterize the user's preferences w.r.t. different candidate news $\boldsymbol{e}_k$. To illustrate the difference of impact that each adopted news $\boldsymbol{e}_j^i$ has on the candidate one $\boldsymbol{e}_k$, the attention network automatically allocates weights to adopted news to aggregate a user's preferences w.r.t. $\boldsymbol{e}_k$. For a given candidate k represented as $\boldsymbol{e}_k$, the impact weight that clicked news $\boldsymbol{e}_j^i$ in user i's history has on candidate news $\boldsymbol{e}_k$ is

$$\alpha_{k,j}^i = \text{softmax}(\mathcal{N}(\boldsymbol{e}_k, \boldsymbol{e}_j^i)) = \frac{\exp(\mathcal{N}(\boldsymbol{e}_k, \boldsymbol{e}_j^i))}{\sum_{j=1}^{s_i}\exp(\mathcal{N}(\boldsymbol{e}_k, \boldsymbol{e}_j^i))} \tag{19}$$

where $\mathcal{N}$ is a DNN serving as the attention network, which takes the candidate news $\boldsymbol{e}_k$ and one piece of adopted news $\boldsymbol{e}_j^i$ as input and outputs the corresponding impact weight between the two. The softmax function is used for normalization. With impact weights, the entire history can then be summed up as

$$\boldsymbol{e}^i = \sum_{j=1}^{s_i}\alpha_{k,j}^i\boldsymbol{e}_j^i \tag{20}$$

Table 1: Dataset description.

	Twitter	*Weibo*
# users	248,195	692,833
# retweet records	4,999,535	31,211,347
# original tweets	4,566,942	232,978
# news	441,632	13,153
avg. # words per title	6.94	7.26
median length of diffusion chain	2.74	23
max length of diffusion chain	124	31009

Finally, the concatenation of the candidate news embedding $\boldsymbol{e}_k$ and the user's integrated history embedding $\boldsymbol{e}^i$ is fed into another DNN $\mathcal{D}$ to predict the probability that i adopts the candidate news $\boldsymbol{e}_k$, by

$$P_{k,i} = \mathcal{D}(\boldsymbol{e}_k, \boldsymbol{e}^i). \tag{21}$$

6 Experimental Evaluation

Datasets For our experiments, we use the same datasets as [11], originating from the two most popular micro-blogging platforms in China (*S*ina Weibo) and worldwide (*Twitter*). The news articles have been crawled, starting from the links appearing in tweets.

While the *Twitter* dataset is collected through the Twitter API, the *Weibo* one is a publicly available dataset [61], especially designed for information diffusion studies. Both datasets are diffusion-oriented, consisting of tweets pertaining to news and the follower / followee network of the posting users, where for each tweet of a piece of news (adoption), the information on whether it is a retweet or an original tweet, and the retweet / original user are recorded. The main statistics for these datasets are in Table 1.

6.1 Baselines

The following state-of-the-art methods are used as baselines in our experiments.

- **LibFM** [39] is a classical factorization model in feature engineering to estimate interactions. It has been widely used in recommendation tasks for click-through rate (CTR) prediction based on users' and items' features. Here, the news title embedding and the node embedding of target users are used as input features.

- **DeepWide** [4] is a deep learning-based model, containing a non-linear part (deep) and linear part (wide) to learn feature interactions. News title embeddings and node embeddings of users involved in the news diffusion process are concatenated as input.

- **DeepFM** [19] is an end-to-end factorization machine-based deep neural model, having a shared input with "wide" and "deep" parts compared to DeepWide; it uses the same input as DeepWide.

- **DCN** [47] introduces a cross network that is more efficient in learning bounded-degree feature interactions, while keeping the benefit of DNN in high dimensional nonlinear feature learning. In the experiments, we use the same input as DeepWide.

- **DKN** [46] recommends news by exploiting an external knowledge graph to capture relationships between news, and links the candidate news with the target user's history of adopted news both at knowledge-level and at semantic-level. To compare with DKN, experiments are carried out by also incorporating the knowledge entity embeddings into LISM (denoted as LISM(+)).
- **DAN** [63] enhances the framework of DKN with knowledge-entity type information, and uses an LSTM mechanism to model the sequential evolution of users' interests. Accordingly, for this baseline method, we use knowledge entities, entity types, as well as an LSTM to aggregate the users' history.
- **GERL** [13] builds a bipartite graph where users and news are both represented as nodes, and neighboring nodes are respectively aggregated in the representation of news or users, along with the semantic information for the recommendation. For this baseline method, we build the same kind of graph.
- **NAML** [51] exploits semantics and news categories, by a multi-level attention mechanism encoding news and users. For this baseline method, we use the topic distribution vector for categories and the news' titles for the semantics dimension.
- **IGNiteR** [11] is the extremely recent and most related approach, studying the same problem we consider in this paper, influence-aware news recommendation in micro-blogging scenarios, modeled as a sequential recommendation task. They follow the assumption that users tend to adopt similar news successively, and their sequential model incorporates temporal aspects, such as attractiveness and timeliness.
- **DICER** [12] is a social-aware recommendation model, which leverages the social connections among users to build a user-user graph, complementing the user-item and item-item graphs constructed from the click history. To compensate for the semantic dimension, we replace the ID embedding matrix of news items with the averaged news title embedding.

Note: we did not compare with the NRMS model [52], as it bears significant similarities with NAML on the attention mechanism. Also, we leave as a future extension the use of PLMs in both our model and in NAML (basically corresponding to [53]'s NAML-BERT), but also in the UNBERT approach of [62]. Finally, other news recommendation models – such as CPRS [50] or FedRec [55] – were not included in our comparison, as they pertain to aspects that are orthogonal to our framework and at the same time available only in proprietary data from commercial platforms like MSN News. Finally, we stress that for completeness purposes we include in our comparison DAN and IGNiteR, even though they exploit as additional information the adoption timestamps.

6.2 Experimental Setup

Our algorithm was implemented and evaluated using Tensorflow. As the two datasets record users' behaviour over a span of 3 years, for each user in our predictive framework an observation window of 3 months of clicked history is used; in this way, a user from the application may be split into several ones in our framework. We randomly sampled as negative samples from the news released during the given period and not clicked by the target user. We split the train / test set by the timeline, so that around 85% of data is used for training and 15% is used for testing; we randomly sample 10% from the training set for validation. To remove outliers, we also used other filters as follows: the maximum number of clicked news per user is set to 20, the maximum title length is set to 10 for *Twitter* and to 15 for *Weibo*, and the maximum number of users involved in a diffusion is set to 30. In the training phase, the ratio between

negative samples and positive samples is 4, and in the test phase the maximum number of negative samples is set to 10. One year worth of diffusion cascades has been used in the training phase, to obtain the node embedding of users. We used node embeddings with dimensionality $g = \{50, 100, 200, 300\}$. As to semantic embeddings, we chose the Chinese word vectors pre-trained on Weibo [29] corpus with word2vec of the Chinese words for *Weibo* dataset, and Glove [36] vectors pre-trained on Twitter corpus for the *Twitter* dataset. The word embedding dimensionality is $f = \{50, 100, 200, 300\}$. In the SCNN part, the window size is set from the combination of $l = \{1, 2, 3, 4\}$, and in each window size, the filter number varies in $\{32, 64, 128, 256\}$. Further discussion on the variants and hyper-parameters will be given in Section 6.3.

The best set of parameters is determined by the performance in the validation phase, with the average AUC, F1, MRR, NDCG@5, NDCG@10 scores as the evaluation metrics. We did our best to tune the baseline methods. In DKN, DAN, GERL, DICER and NAML, the datasets are pre-processed in the same way (sampling ratio, user history, title length), while the knowledge-entity length is set to 10 for DKN and DAN, and the number of neighboring user is set to 20 in GERL. IGNiteR uses the same one year worth of diffusion cascades to obtain the node embeddings. In DICER, the number user / item links is set to 10 per item, and the number of friends in DICER is set to 30. The attention hidden layer size is set to 200 for NAML and GERL, 256 for DKN and DAN, and $[128, 64]$ for DICER. The hidden layer structure of DeepWide model is set as $[512, 512]$, and for DeepFM and DCN as $[256, 256]$. For LibFM, the number of factors is set as 50. The dimensions of word embeddings and node embeddings are set as $f = 100$ and $g = 100$. Each experiment is repeated for 10 times and the average performance is reported in the results.

6.3 Results Analysis

In this section, we present the comparison of LISM with the baselines, the comparison among variants of LISM, as well as the impact of hyper-parameters on its performance.

6.3.1 Comparison with Baselines

Table 2: Comparison with baseline methods on *Weibo*.

Models	AUC	F1	MRR	NDCG@5	NDCG@10
LibFM	65.70 ± 0.45	63.70 ± 1.50	15.84 ± 1.09	55.37 ± 0.59	57.95 ± 0.21
DeepFM	68.16 ± 0.46	65.29 ± 0.98	16.13 ± 1.80	59.47 ± 1.23	61.15 ± 1.01
DeepWide	68.30 ± 2.03	66.96 ± 1.48	16.26 ± 2.12	61.36 ± 0.78	62.08 ± 0.61
DCN	69.57 ± 2.021	67.65 ± 0.50	16.90 ± 0.91	61.87 ± 0.56	63.14 ± 0.49
DKN	71.82 ± 1.38	69.97 ± 2.26	18.56 ± 1.09	64.38 ± 1.23	66.15 ± 1.58
DAN	72.69 ± 0.95	71.87 ± 0.53	19.79 ± 0.63	65.13 ± 0.77	67.49 ± 1.22
DICER	73.53 ± 0.65	71.28 ± 0.93	20.06 ± 0.28	66.03 ± 1.27	68.59 ± 1.53
GERL	73.71 ± 1.28	71.54 ± 0.68	20.59 ± 0.84	66.32 ± 0.59	68.98 ± 0.52
NAML	74.85 ± 0.78	72.15 ± 1.02	21.63 ± 1.06	67.44 ± 1.22	69.38 ± 1.28
IGNiteR	76.92 ± 0.58	73.85 ± 1.01	23.63 ± 0.36	69.59 ± 0.91	72.01 ± 0.78
LISM	75.94 ± 1.34	72.56 ± 1.05	23.44 ± 0.67	68.35 ± 1.78	71.08 ± 0.51
LISM(+)	$\mathbf{76.37 \pm 1.86}$	$\mathbf{72.89 \pm 1.12}$	$\mathbf{23.75 \pm 0.26}$	$\mathbf{68.78 \pm 1.2}$	$\mathbf{71.59 \pm 1.48}$

Table 3: Comparison with baseline methods on *T*witter.

Models	AUC	F1	MRR	NDCG@5	NDCG@10
LibFM	60.72 ± 1.17	59.80 ± 1.32	13.30 ± 0.59	57.76 ± 0.67	59.59 ± 0.41
DeepFM	65.23 ± 1.12	64.30 ± 1.24	13.89 ± 1.31	61.56 ± 0.96	63.21 ± 0.97
DeepWide	65.78 ± 1.25	64.96 ± 0.92	14.07 ± 0.52	61.96 ± 0.62	63.88 ± 0.84
DCN	66.40 ± 1.13	65.35 ± 1.06	14.58 ± 0.31	62.47 ± 0.76	64.02 ± 2.01
DKN	68.72 ± 0.96	67.02 ± 0.86	15.22 ± 0.28	64.17 ± 1.32	66.35 ± 1.45
DAN	69.83 ± 0.34	69.28 ± 0.59	15.62 ± 0.54	64.83 ± 1.34	67.02 ± 1.33
DICER	70.38 ± 1.72	69.08 ± 1.95	15.79 ± 0.21	65.31 ± 1.72	67.70 ± 1.56
GERL	70.39 ± 1.38	69.48 ± 1.68	16.35 ± 0.32	65.54 ± 2.59	68.44 ± 2.52
NAML	71.05 ± 2.18	70.15 ± 1.92	16.54 ± 0.55	67.19 ± 0.72	70.58 ± 1.23
IGNiteR	74.82 ± 0.58	72.25 ± 0.53	17.97 ± 0.63	69.57 ± 1.22	72.75 ± 0.46
LISM	72.87 ± 1.05	71.68 ± 1.80	17.47 ± 0.18	69.01 ± 0.95	72.08 ± 1.25
LISM(+)	$\mathbf{73.41 \pm 1.83}$	$\mathbf{72.53 \pm 1.70}$	$\mathbf{17.93 \pm 0.41}$	$\mathbf{69.43 \pm 1.36}$	$\mathbf{72.77 \pm 1.24}$

In the comparison with baselines, in LISM the dimensions of node and word embeddings are both 100 and the number of filters is 200. As to the graph embedding method, the best result of LISM on both *Weibo* and *Twitter* are obtained with the node representations generated by the IMINFECTOR cascade-based graph embedding method, trained on the diffusion cascades. Both the Chinese and English knowledge entity embeddings of *Weibo* and *Twitter* respectively are generated by the TransE model for knowledge graph embedding. We can see from the results of Table 2 and Table 3 that LISM outperforms all the baselines on news data from social media, with the exception of our direct competitor (IGNiteR), which validates once more the motivation to exploit diffusion patterns in news recommendation.

In terms of effectiveness, we can observe that LISM is on par with IGNiteR. The latter performs slightly better both in the Twitter case (on four out of the five metrics) and in the Weibo case (on three out of the five metrics). This comes as a pleasant surprise, since, as discussed in the following section, LISM is significantly lighter computationally, and does not require knowledge of the time dimension (adoption timestamps) and the sequential user adoption history this dimension enables.

Within the class of generic DL-based models, LibFM performs worse than other models, which indicates that the factorization model fails to capture nonlinear interactions in news and the social graph. DeepFM makes up for this with its deep part for learning high-dimensional interaction. DCN and DeepWide outperform DeepFM in both datasets. Considering the performance of LibFM, it is possible that the factorization model is less effective in learning graph information.

We can note that, with the exception of DICER, the performance of the neural news recommendation models overpasses that of the generic deep recommendation models. Although DICER is not designed for news recommendation, with the social-aware information integrated in the user-user graph, it performs quite well under in our social scenario. NAML performs the best among the baseline methods, showcasing the efficiency of the multi-view attention mechanism that aggregates news features. GERL and DICER perform almost comparably well on both datasets, with GERL slightly outperforming the other. Our intuition is that these models share common ideas in the building of graph structures encompassing users and news, with DICER leveraging the social relations and a well-designed GNN to construct such graphs, while GERL balances the results with a dedicated news transformer for semantic information.

We can also see that although knowledge-level information brings improvements in both datasets,

the ones in Twitter are larger. A possible explanation may be that knowledge-level information weighs more for the recommendation in Twitter than in Weibo.

6.3.2 Comparison among LISM variants

To give insights into LISM, experiments are carried out to compare the variants of the model. From Table 5, we can see that with node embeddings, LISM performs better on all metrics, which confirms that the influence-aware user representations improve news recommendation on social platforms. Among the graph embedding methods, the best results on *Weibo* and *Twitter* are obtained with the influence-base embedding, which generates node representations based on cascades, while node2vec performs best among the topology-based methods.

Table 4: Comparison among LISM variants in *Weibo*.

Variants	Weibo				
	AUC	F1	MRR	NDCG@5	NDCG@10
Node embedding	**75.94 ± 1.34**	**72.56 ± 1.05**	**23.44 ± 0.67**	**68.35 ± 1.78**	**71.08 ± 0.51**
Without node embedding	72.08 ± 1.78	70.13 ± 2.15	17.87 ± 0.78	64.65 ± 1.57	67.21 ± 1.67
IMINFECTOR	**75.94 ± 1.34**	**72.56 ± 1.05**	**23.44 ± 0.67**	**68.35 ± 1.78**	**71.08 ± 0.51**
Node2vec	73.55 ± 0.89	71.39 ± 1.21	19.43 ± 0.35	66.07 ± 1.02	68.63 ± 1.17
LINE	73.03 ± 1.74	70.97 ± 0.81	18.63 ± 0.22	65.67 ± 1.32	68.14 ± 1.01
DeepWalk	72.68 ± 1.57	70.78 ± 1.31	18.03 ± 0.45	65.12 ± 1.24	67.77 ± 1.06
Attention	**75.94 ± 1.34**	**72.56 ± 1.05**	**23.44 ± 0.67**	**68.35 ± 1.78**	**71.08 ± 0.51**
Without attention	70.18 ± 1.05	68.54 ± 2.37	16.77 ± 0.86	62.78 ± 1.02	64.19 ± 1.97
Knowledge	**76.37 ± 1.86**	**72.89 ± 1.12**	**23.75 ± 0.26**	**68.78 ± 1.2**	**71.59 ± 1.48**
Without knowledge	75.94 ± 1.34	72.56 ± 1.05	23.44 ± 0.67	68.35 ± 1.78	71.08 ± 0.51
Linear transformation	75.60 ± 1.32	72.53 ± 1.03	23.42 ± 0.85	**68.37 ± 1.73**	71.04 ± 0.30
Non-linear transformation	**75.94 ± 1.34**	72.56 ± 1.05	**23.44 ± 0.67**	68.35 ± 1.78	**71.08 ± 0.51**
Zero-padding	75.61 ± 1.04	**72.58 ± 1.15**	23.36 ± 0.81	68.12 ± 1.88	70.97 ± 0.98

In the two datasets, the use of an attention mechanism greatly improves performance. The main reason is that the attention network can automatically aggregate a user's diverse interests with respect to the candidate news. Performance is improved on both datasets when knowledge entities are incorporated in CTR prediction, which corroborates with the DKN [46] conclusions about the importance of knowledge-level connections in news recommendation. In terms of transformation function to align node and word vectors, there is no option that is significantly better than the other.

6.3.3 Training Complexity

Our training complexity analysis is related to the various hyper-parameters of the methods we consider. Since each model has its own set of parameters to be tuned, we cannot guarantee here absolute fairness in the comparison. When using computation time as the metric for efficiency / data complexity, a common phenomenon in deep learning is that with the complexity of the neural network going up (number of layers, number of cells, etc.), the predictive performance may be improved by a small margin, while the computation time increases by a large margin. However, we strive to avoid biases as much as possible by running the various models based on the same parameter setting for their common stages or components. Note that the parameter setting may now be different from the one used in the previous effectiveness comparison, where we used the best tuned hyper-parameters for each model.

Table 5: Comparison among LISM variants in *Twitter*.

Variants	Twitter				
	AUC	F1	MRR	NDCG@5	NDCG@10
Node embedding	**72.87 ± 1.05**	**71.68 ± 1.80**	**17.77 ± 0.18**	**69.01 ± 0.95**	**72.08 ± 1.25**
Without node embedding	68.74 ± 1.07	65.78 ± 1.53	14.49 ± 0.70	63.85 ± 0.92	65.95 ± 1.73
IMINFECTOR	**72.87 ± 1.05**	**71.68 ± 1.80**	**17.47 ± 0.18**	**69.01 ± 0.95**	**72.08 ± 1.25**
Node2vec	70.16 ± 0.85	69.68 ± 1.56	15.70 ± 0.73	65.03 ± 1.25	67.38 ± 0.97
LINE	69.56 ± 1.27	66.28 ± 1.09	15.13 ± 0.49	64.73 ± 1.68	66.87 ± 0.85
DeepWalk	69.12 ± 2.27	65.98 ± 2.23	14.79 ± 0.66	64.21 ± 1.79	66.37 ± 1.90
Attention	**72.87 ± 1.05**	**71.68 ± 1.80**	**17.77 ± 0.18**	**69.01 ± 0.95**	**72.08 ± 1.25**
Without attention	67.94 ± 2.57	65.18 ± 2.93	13.98 ± 1.69	63.35 ± 1.74	65.75 ± 0.89
Knowledge	**73.41 ± 1.83**	72.53 ± 1.70	**17.93 ± 0.21**	**69.43 ± 1.36**	**72.77 ± 1.24**
Without knowledge	72.87 ± 1.05	71.68 ± 1.80	17.77 ± 0.18	69.01 ± 0.95	72.08 ± 1.25
Linear transformation	72.80 ± 1.15	71.58 ± 1.69	**17.82 ± 0.15**	**69.02 ± 1.55**	72.04 ± 2.25
Non-linear transformation	**72.87 ± 1.05**	**71.68 ± 1.80**	17.77 ± 0.18	69.01 ± 0.95	**72.08 ± 1.25**
Zero-padding	72.83 ± 1.95	71.58 ± 1.93	17.69 ± 0.25	68.97 ± 1.59	72.03 ± 1.57

For all models, we sampled from 700 users in *Weibo*, and we generated (i) around 20000 samples for LibFM, DeepWide, DeepFM, DCN, DKN, DAN and NAML, (ii) around 11000 samples for GERL and DICER, and (iii) 7500 samples for LISM and IGNiteR; for GERL, DICER (resp. LISM), we made sure that enough graph neighbours (resp. cascade nodes) are available for training. From the diffusion cascades, we sampled the diffusion information from a 6-months period to generate the cascade-based node embeddings, for both LISM and IGNiteR[1]. For the CNN-based models, the number of filters is set to 200, and the kernel size is set to 3. For the models using an attention mechanism, the number of attention layer units is 200. The other settings are as mentioned in the previous section. The batch size is 200, and the number of epochs is 10. For the performance vs. complexity tradeoff, we use the computation time (order of minutes) for each epoch as the complexity indicator, while NDCG@5 is used as the performance indicator.

We can see from Figure 4 that although LibFM, DeepWide, DeepFM and DCN are relatively fast, they perform significantly worse than the neural news recommendation models.

In contrast to Sec. 6.3.1, where all the models are fine-tuned for optimal performance, IGNiteR shows a clearer dependence on the number of training cascades and on the news adoption timespan. With the diffusion cascades used for training spanning a period of 6 months, although IGNiteR outperforms LISM, the sequential analysis of users' behavior and of news adoption lifecycles increases its complexity. LISM achieves by far the best complexity vs. effectiveness tradeoff, being runner-up and close to IGNiteR's performance, while working with significantly fewer samples and thus having a reduced training time, compared to all the other neural news recommendation models. Within this class, DKN seems to be the most costly method.

Regarding data scalability, we found in our experiments that the training time of LISM evolves linearly with the size of the training data (further details are omitted here).

[1]The training process for the graph neural network takes around 4 minutes for each epoch, with 10 epochs in total. The number of involved nodes was 1.17M.

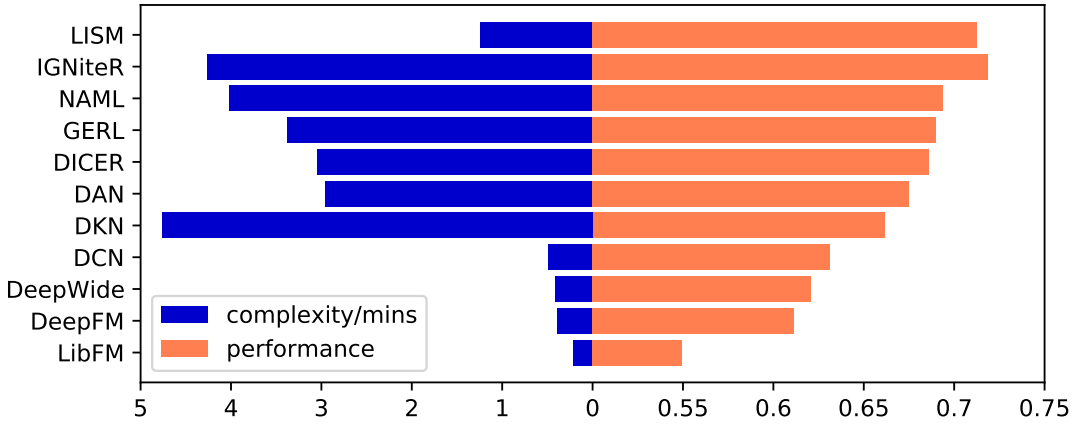

Figure 4: Performance vs. complexity tradeoff.

6.3.4 Analysis on Hyper-Parameters

Parameter setting is important for the performance of LISM. In Figure 5, we analyze the sensitivity of the AUC and F1 metrics to the embedding dimension of words and nodes, and to the CNN's hyper-parameters.

1. *Word embedding dimension and node embedding dimension.* We select from all the combinations of dimensions f and g from $\{50, 100, 200, 300\}$. For a given node embedding dimension g, the performance of LISM improves as the word embedding dimension f increases, but it decreases when the f continues going up, since a large dimension of word embedding may introduce noise and lead to the inability to retain effective information. However, there is an exception for node embedding dimension $g = 100$, which maintains the performance even with large word embedding dimension.

2. *CNN hyper-parameters.* The best performance is obtained with a filter at 128, in all combinations between window size and filter size. LISM's performance drops when the window size gets complicated, perhaps due to over-fitting. Compared to the AUC metric, the results on F1 seem to be more sensitive to the variation of window size, while for AUC the changes are smoother.

7 Conclusion

We propose in this paper LISM, a lightweight content-based deep learning model for news recommendation in micro-blogging scenarios. To incorporate the awareness about news due to social influence, we represent users by embeddings obtained through methods leveraging the diffusion history (cascades), in such a way that news is endowed with social-related information. A special designed CNN model is applied to deal with the joint representation of news and an attention mechanism allows us to aggregate users' diverse interests with respect to candidate news. Extensive experiments are conducted on two real-world datasets, including a publicly available one. They show that news diffusion patterns can significantly improve the effectiveness of news recommendation, compared to influence and social-agnostic models, even when the problem is not (or cannot be) modeled as a sequential recommendation one. Furthermore, they show that the proposed model strikes a balance between complexity and effectiveness, being

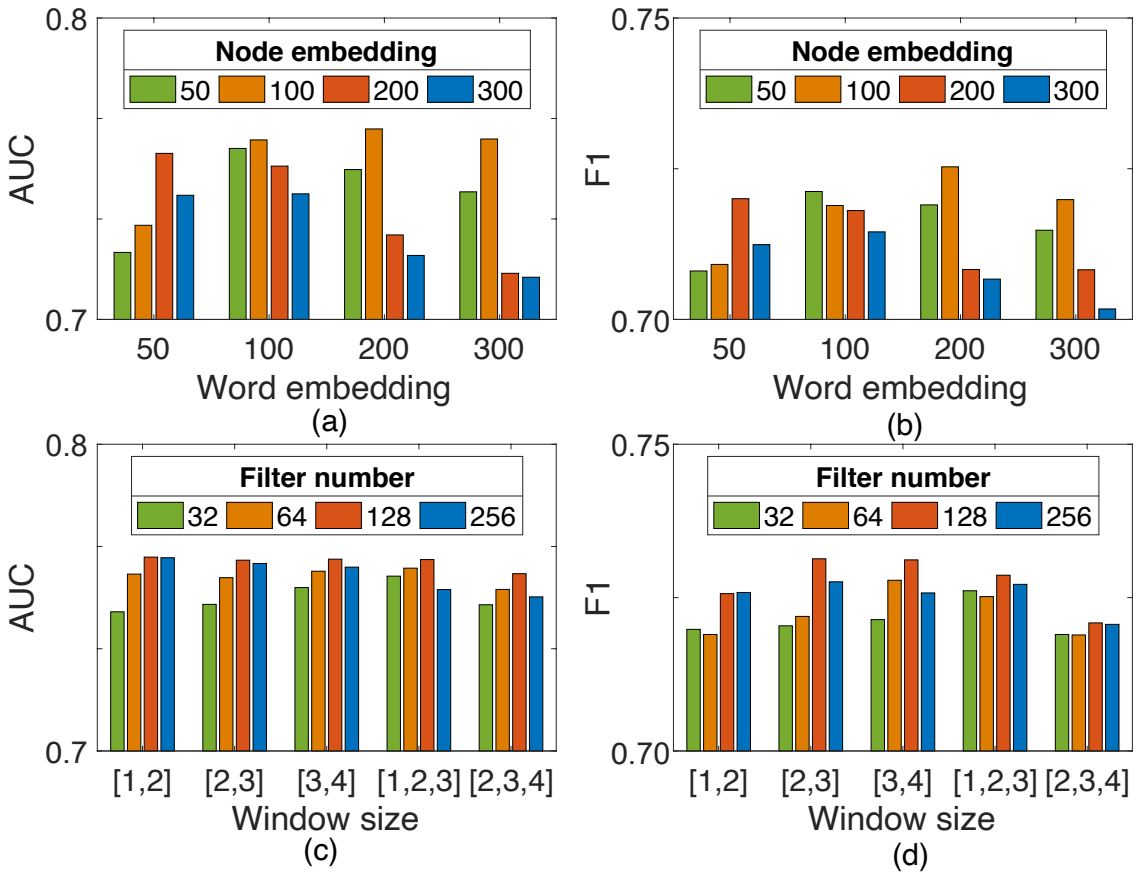

Figure 5: Parameter analysis in LISM.

therefore applicable in the highly dynamic (high data velocity) context of micro-blogging platforms. We believe that our model is generic enough to be applied in any social media setting, when adoptions and diffusion traces are available, including those where adoption timestamps may not be known. A future extension to our work is the use of PLMs for a deeper semantic understanding of news.

References

[1] Batagelj, V., Mrvar, A.: Pajek—analysis and visualization of large networks. In: Graph drawing software (2004)

[2] Bhagat, S., Cormode, G., Muthukrishnan, S.: Node classification in social networks. In: Social network data analytics (2011)

[3] Bughin, J.: Getting a sharper picture of social media's influence. McKinsey Quarterly (2015)

[4] Cheng, H.T., et al.: Wide & deep learning for recommender systems. In: Workshop on deep learning for recommender systems (2016)

[5] Choi, K., Fazekas, G., Sandler, M., Cho, K.: Convolutional recurrent neural networks for music classification. In: ICASSP (2017)

[6] Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT (2019)

[7] Donkers, T., Loepp, B., Ziegler, J.: Sequential user-based recurrent neural network recommendations. In: RecSys (2017)

[8] Dos Santos, C., Gatti, M.: Deep convolutional neural networks for sentiment analysis of short texts. In: COLING (2014)
[9] Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., Yin, D.: Graph neural networks for social recommendation. In: The World Wide Web Conference. pp. 417–426 (2019)
[10] Fedoryszak, M., Frederick, B., Rajaram, V., Zhong, C.: Real-time event detection on social data streams. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019 (2019)
[11] Feng, Y., Cautis, B.: Igniter: News recommendation in microblogging applications (extended version). arXiv preprint arXiv:2210.01942 (2022)
[12] Fu, B., Zhang, W., Hu, G., Dai, X., Huang, S., Chen, J.: Dual side deep context-aware modulation for social recommendation (2021)
[13] Ge, S., Wu, C., Wu, F., Qi, T., Huang, Y.: Graph enhanced representation learning for news recommendation. In: Proceedings of The Web Conference 2020. pp. 2863–2869 (2020)
[14] Gomez-Rodriguez, M., Leskovec, J., Krause, A.: Inferring networks of diffusion and influence. TKDD (2012)
[15] Goyal, A., Bonchi, F., Lakshmanan, L.V.S.: A data-based approach to social influence maximization. Proc. VLDB Endowement (2011)
[16] Goyal, P., Ferrara, E.: Graph embedding techniques, applications, and performance: A survey. Knowledge-Based Systems (2018)
[17] Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: SIGKDD (2016)
[18] Guo, G., Zhang, J., Yorke-Smith, N.: Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In: Proceedings of the AAAI conference on artificial intelligence. vol. 29 (2015)
[19] Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: DeepFM: a factorization-machine based neural network for CTR prediction. arXiv preprint arXiv:1703.04247 (2017)
[20] Hidasi, B., Quadrana, M., Karatzoglou, A., Tikk, D.: Parallel recurrent neural network architectures for feature-rich session-based recommendations. In: RecSys (2016)
[21] Hu, L., Li, C., Shi, C., Yang, C., Shao, C.: Graph neural news recommendation with long-term and short-term interest modeling. Information Processing & Management **57**(2), 102142 (2020)
[22] Huang, H., Yan, Q., Chen, L., Gao, Y., Jensen, C.S.: Statistical inference of diffusion networks. IEEE TKDE (2019)
[23] Institute, R.: Digital News Report 2022 (2020), `https://reutersinstitute.politics.ox.ac.uk/digital-news-report/2022`
[24] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: CVPR (2014)
[25] Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: SIGKDD (2003)
[26] Kouki, P., Fakhraei, S., Foulds, J., Eirinaki, M., Getoor, L.: HyPER: A flexible and extensible probabilistic framework for hybrid recommender systems. In: RecSys (2015)
[27] Leskovec, J., Singh, A., Kleinberg, J.: Patterns of influence in a recommendation network. In: PAKDD (2006)
[28] Li, M., Wang, L.: A survey on personalized news recommendation technology. IEEE Access (2019)
[29] Li, S., Zhao, Z., Hu, R., Li, W., Liu, T., Du, X.: Analogical reasoning on chinese morphological and semantic relations. arXiv:1805.06504 (2018)
[30] Li, Y., Fan, J., Wang, Y., Tan, K.: Influence maximization on social graphs. TKDE (2018)
[31] Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of the American society for information science and technology (2007)
[32] Ma, H., Zhou, D., Liu, C., Lyu, M.R., King, I.: Recommender systems with social regularization. In: WSDM (2011)
[33] Neculoiu, P., Versteegh, M., Rotaru, M.: Learning text similarity with siamese recurrent networks. In: Workshop on Representation Learning for NLP (2016)
[34] Panagopoulos, G., Malliaros, F., Vazirgiannis, M.: Multi-task learning for influence estimation and maximization. IEEE TKDE (2020)

[35] Pazzani, M.J., Billsus, D.: Content-based recommendation systems. In: Adaptive web (2007)
[36] Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: EMNLP (2014)
[37] Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: KDD (2014)
[38] Phelan, O., McCarthy, K., Smyth, B.: Using twitter to recommend real-time topical news. In: RecSyS (2009)
[39] Rendle, S.: Factorization machines with LibFM. ACM TIST (2012)
[40] Ricci, F.: Recommender systems: Models and techniques. (2014)
[41] Saerens, M., Fouss, F., Yen, L., Dupont, P.: The principal components analysis of a graph, and its relationships to spectral clustering. In: ECML (2004)
[42] Saito, K., Nakano, R., Kimura, M.: Prediction of information diffusion probabilities for independent cascade model. In: KES (2008)
[43] Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted boltzmann machines for collaborative filtering. In: ICML (2007)
[44] Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative filtering recommender systems. In: Adaptive web (2007)
[45] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: Large-scale information network embedding. In: WWW (2015)
[46] Wang, H., Zhang, F., Xie, X., Guo, M.: DKN: Deep knowledge-aware network for news recommendation. In: WWW (2018)
[47] Wang, R., Shivanna, R., Cheng, D., Jain, S., Lin, D., Hong, L., Chi, E.: DCN-M: Improved deep & cross network for feature cross learning in web-scale learning to rank systems. arXiv:2008.13535 (2020)
[48] Wang, X., He, X., Wang, M., Feng, F., Chua, T.S.: Neural graph collaborative filtering. In: Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval. pp. 165–174 (2019)
[49] Wu, C., Wu, F., An, M., Huang, J., Huang, Y., Xie, X.: NPA: neural news recommendation with personalized attention. In: SIGKDD (2019)
[50] Wu, C., Wu, F., Qi, T., Huang, Y.: User modeling with click preference and reading satisfaction for news recommendation. In: IJCAI (2020)
[51] Wu, C., Wu, F., An, M., Huang, J., Huang, Y., Xie, X.: Neural news recommendation with attentive multi-view learning. arXiv preprint arXiv:1907.05576 (2019)
[52] Wu, C., Wu, F., Ge, S., Qi, T., Huang, Y., Xie, X.: Neural news recommendation with multi-head self-attention. In: EMNLP-IJCNLP'19 (2019)
[53] Wu, C., Wu, F., Qi, T., Huang, Y.: Empowering news recommendation with pre-trained language models. In: SIGIR (2021)
[54] Wu, C., Wu, F., Qi, T., Li, C., Huang, Y.: Is news recommendation a sequential recommendation task? In: SIGIR '22: The 45th International ACM SIGIR (2022)
[55] Wu, C., Wu, F., Qi, T., Liu, Q., Tian, X., Li, J., He, W., Huang, Y., Xie, X.: Feedrec: News feed recommendation with various user feedbacks. In: WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022 (2022)
[56] Wu, L., Li, J., Sun, P., Hong, R., Ge, Y., Wang, M.: Diffnet++: A neural influence and interest diffusion network for social recommendation. IEEE Transactions on Knowledge and Data Engineering (2020)
[57] Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., Tan, T.: Session-based recommendation with graph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 346–353 (2019)
[58] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems (2020)
[59] Xiao, W., Zhao, H., Pan, H., Song, Y., Zheng, V.W., Yang, Q.: Beyond personalization: Social content recommendation for creator equality and consumer satisfaction. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 235–245 (2019)
[60] Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 974–983 (2018)
[61] Zhang, J., Liu, B., Tang, J., Chen, T., Li, J.: Social influence locality for modeling retweeting behaviors. In:

IJCAI (2013)
[62] Zhang, Q., Li, J., Jia, Q., Wang, C., Zhu, J., Wang, Z., He, X.: Unbert: User-news matching bert for news recommendation. In: IJCAI (2021)
[63] Zhu, Q., Zhou, X., Song, Z., Tan, J., Guo, L.: DAN: Deep attention neural network for news recommendation. In: AAAI (2019)