Predicting the Popularity of Online Content with Knowledge-enhanced Neural Networks

Hongjian Dou
School of Information, Renmin University of China
hongjiandou@ruc.edu.cn

Wayne Xin Zhao*
School of Information, Renmin University of China
batmanfly@gmail.com

Yuanpei Zhao
School of Information, Renmin University of China
yuanpeizhao@ruc.edu.cn

Daxiang Dong
Baidu Inc., Beijing, China
dongdaxiang@baidu.com

Ji-Rong Wen
Beijing Key Laboratory of Big Data Management and Analysis Methods
jrwen@ruc.edu.cn

Edward Y. Chang
Research & Innovation, HTC
eyuchang@gmail.com

ABSTRACT

Predicting the popularity of online items has been an important task to understand and model online popularity dynamics. Feature-based methods are one of the mainstream approaches to tackle this task. Although many efforts have been made, the solutions to three key issues still remain unclear or need improvement, *i.e.,* what kind of general contexts can be used, how to represent the contexts in a general form, and how to effectively leverage the contexts for prediction. To address these issues, we propose to link online items with existing knowledge base (KB) entities, and leverage KB information as the context for improving popularity prediction. We represent the KB entity by a latent vector, encoding the related KB information in a compact way. We further propose a novel prediction model based on LSTM networks, adaptively incorporating KB embedding of the target entity and popularity dynamics from items with similar entity information. Extensive experiments on three real-world datasets demonstrate the effectiveness of the proposed model.

CCS CONCEPTS

• **Information systems** → **Data mining**;

KEYWORDS

Deep learning, Popularity Prediction, Knowledge Base

1 INTRODUCTION

With the rapid development of Web platforms, various online items (*a.k.a.,* online content), such as Amazon e-books and YouTube videos, are available to users. The increasing of online items has intensified the competition for users' attention [29], since only a small number of items become popular. In order to better understand and model online popularity dynamics, the task of predicting the popularity of web content [20, 28] has become very important and attracted much attention from the research community.

Traditional methods try to build prediction models (*e.g.,* regression models) on time series data of historical popularity statistics [28]. Since web content is usually associated with rich contexts, many studies further leverage different feature information for improving prediction, including content features [23], user features [33], structural features [12] and spatial features [8]. These feature-based approaches utilize both time series and context data in order to learn a better prediction model. However, they usually rely on hand-crafted extraction rules or platform-dependent content characteristics, it is not flexible to apply them to a broad spectrum of data domains. Furthermore, these methods extend traditional machine learning algorithms, and may not be effective to fully utilize context information due to complex data characteristics. For popularity prediction with context information, we have to consider three important issues: (1) what kind of general contexts to be used, (2) how to represent the contexts in a unified and compact way, and (3) how to integrate and utilize the contexts.

To address these difficulties, in this paper, we propose to leverage knowledge bases (KB) for improving the prediction of the popularity of online items. KBs store entity information in triples of the form (head entity, relation, tail entity), typically corresponding to entity attributes. Compared with traditional data forms, KBs provide a general way to flexibly characterize context information of entities from various domains, and emphasize the interconnection of data. Many large-scale KBs have been released for public usage, such as Freebase [7] and Yago [27]. By linking the items from online platforms with the entities from existing KBs, we are able to utilize rich KB information of online items from a variety of domains[1]. For example, by querying Freebase search API with the Isbn number, we can accurately associate

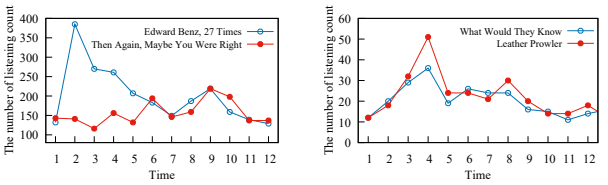
*Corresponding author

KDD'18 Deep Learning Day, August 2018, London, UK

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[1] In addition to reusing KB information, we can also extract new context information, represent it in the form of KB triples, and integrate them

an Amazon online book with a unique entity in Freebase, and then obtain its KB information (*e.g.,* the authors) via reading out all the related KB triples. We hypothesize KBs are of valuable information to improve popularity prediction of online content.



(a) Partially similar trends. (b) Completely similar trends.

Figure 1: Popularity dynamics comparison between two example pairs of items with similar entity information. These four items (*i.e.,* songs) are selected from Last.fm music dataset [Schedl, 2016] and the y-axis corresponds to the number of listening count. Songs in each pair are from the same singer. The two songs of the first pair deviate from each other at the beginning and then becomes similar finally.

The key point of our approach is how to design the prediction model which can effectively integrate popularity time series with KB information. Following [2, 13, 17, 18, 31], we propose to learn vectorized representations (*a.k.a.,* KB embedding) for entities and relations in a latent space. In this way, we encode related KB information of an online item into a compact embedding vector. Then, inspired by recent works on deep learning for popularity prediction [14, 16], we adopt the commonly used Long-Short Term Memory (LSTM) networks as the base architecture to develop our predictor. While, it is not trivial to integrate the KB embedding into the LSTM network. We present an example in Fig. 1. As we can see, two online items with similar entity information are likely to have either similar or dissimilar popularity trends at varying time steps. Hence, an ideal prediction model should adaptively integrate and utilize the KB information, extracting useful data characteristics and removing irrelevant information from KB embedding for the prediction task. To achieve this, our solutions are two folds. First, we propose to use the gate mechanism to integrate time series with KB data, which tries to learn an adaptive combination between the two parts conditioned on the current state. Second, we propose to incorporate the popularity dynamics of the items with similar KB information via the attention mechanism. Our model effectively utilize the KB data and adaptively integrate it into the prediction model.

We summarize the contributions of this paper as follows. First, we propose to link online items with KB entities, and leverage KB information for improving online popularity prediction. To our knowledge, it is the first time KB information is utilized in popularity prediction. Second, we propose a novel prediction model based on the LSTM networks, which effectively incorporates the KB embedding of the target item and the popularity dynamics of its similar KB entities. Third, extensive experiments on three real-world datasets demonstrate the effectiveness of the proposed model compared with several competitive baselines.

into existing KBs. In this paper, we only focus on using existing KB information.

2 RELATED WORK

In this section, we review the related studies for popularity prediction.

A classic approach to popularity prediction is to build regression or classification prediction models [20, 28] by taking as input the previous popularity statistics. They make the predictions by characterizing temporal dependence or correlation patterns in the time series data. Since simple prediction models may not be effective to capture complex temporal characteristics, follow-up studies have introduced a series of more powerful prediction models, such as reinforced Poisson process [25], multi-dimensional time-series model [21], lifetime-aware regression model [19] and transfer autoregressive model [4]. With rich context data on the Web, many studies propose to leverage these auxiliary features for improving popularity prediction [5], including content features [23], user features [33], structural features [12] and spatial features [8].

Recently, deep learning has become a popular technique to address various complicated tasks. A typical deep learning approach to popularity prediction is to utilize Recurrent Neural Networks (RNN) to capture temporal dependence and build better predictions [14, 22, 26, 30, 34]. They mainly rely on the excellence of RNN in modeling sequence data. Furthermore, several studies also adopt neural networks as a transformer to leverage various features for popularity prediction, including event signal [6], cascade path [3], cascade graph [16] and multi-modality information [32].

Our work is closely related to the above works. While, we have a different focus, *i.e.,* how to leverage KB information for improving popularity prediction. Although context information has be explored to some extent, to our knowledge, no work has utilized KB data for popularity prediction. As will be shown in the model and experiment parts, it is not trivial to integrate and model KB information into the prediction model. We have made the initiative attempt on this direction.

3 PROBLEM DEFINITION

Let $\mathcal{I}$ denote a set of items on an online platform, *e.g.,* ebook on Amazon or music on Last.fm. An observation window $[1, n]$ of n time steps (*a.k.a.,* intervals) is given[2]. At the t-th time step, each individual item i receives a value measuring its popularity within the current step, denoted by v_t^i. Popularity values reflect the received online attention for an item, *e.g.,* the number of reviews or clicks. By sorting these values by

[2]Note we use 1 to n to indicate a relative time span. Not all the items share the same absolute time span (*i.e.,* lifespan).

time ascendingly, we can form a time series of popularity values for item i, namely $\{v_1^i, \cdots v_t^i, \cdots v_n^i\}$, called *popularity time series*. We are often interested in future popularity. Let $v_{n,m}^i$ denote the incremental popularity in the m steps after time n, so we have $v_{n,m}^i = \sum_{t=n+1}^{n+m} v_t^i$.

Besides popularity time series, we assume that a knowledge base (KB) is also available as the input. A KB is defined over an entity set $\mathcal{V}$ and a relation set $\mathcal{R}$, containing a set of KB triples. A KB triple $\langle e_1, r, e_2\rangle$ denotes there exists relation r from $\mathcal{R}$ between two entities e_1 and e_2 from $\mathcal{V}$, stating a fact stored in KB. For example, a KB triple (China, hasCapitalCity, Beijing) describes that *Beijing* is the capital city of *China*. Since we assume it is possible to link online items with KB entities, item set $\mathcal{I}$ can be considered as a subset of KB entity set $\mathcal{V}$, so we have $\mathcal{I} \subset \mathcal{V}$. By linking an item with a KB entity, we can obtain all its related KB information.

Knowledge-based popularity prediction is to predict the incremental popularity value $v_{n,m}^i$ for an item i after m time steps given previous n popularity values and its KB information. Following [3], we predict the incremental popularity to avoid data dependency. Our definition is general in that we parameterize the task setting with two numbers n and m. When $m = 1$, the task becomes the next-step popularity prediction; when $m = +\infty$, the task becomes final popularity prediction. Also, the granularity of time steps (*e.g.,* day, month or year) and the scale of popularity values (*e.g.,* absolute or normalized values) can be set accordingly for different tasks. We will specify the details in Section 5.

4 THE PROPOSED MODEL

In this section, we present the proposed model for the task of knowledge-based popularity prediction. We start with a base model which adopts the standard LSTM architecture, and then extend the model by incorporating KB information in two aspects, namely KB embedding and KB neighbors.

The notations we will use throughout the article are summarized in Table 1.

Table 1: Notations and explanations.

Notation	Explanation
$\mathcal{I}$	item set
$\mathcal{V}$	entity set in a KB
$\mathcal{R}$	relation set in a KB
r	a relation in $\mathcal{R}$
$\boldsymbol{r}$	embedding vector of relation r
i	an item in $\mathcal{D}$
e_i	the corresponding entity of item i in a KB
$\boldsymbol{e}_i$	embedding vector of entity e_i
t	time index
k	KB neighbor index
v_t^i	the popularity value of item i at time t
$v_{n,m}^i$	the $m-$step incremental popularity of item i at time n
$\boldsymbol{h}_n^i$	hidden state of item i using LSTM at time n
$\boldsymbol{h}_n^k$	hidden state of KB neighbor k using LSTM at time n
$\tilde{\boldsymbol{h}}_n^i$	hidden state of item i using LSTM with KB embedding
$\boldsymbol{s}_n^i$	final representation of item i at time n

4.1 A LSTM-based Popularity Prediction Model

Recurrent Neural Networks (RNN) have been shown effective to capture and characterize the temporal dependence in sequence data, especially the Long Short Term Memory (LSTM) networks [11]. Similar to RNN, the LSTM network generates current hidden state vector $\boldsymbol{h}_t$ conditioned on previous hidden state vector $\boldsymbol{h}_{t-1}$ and current input vector $\boldsymbol{x}_t$, so we have

$$\boldsymbol{h}_t = \text{LSTM}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t; \Theta), \tag{1}$$

where LSTM$(\cdot)$ is the LSTM unit and Θ denotes all the related parameters. By adding input, forget and output gates, LSTM can better capture long sequence dependence. Hence, we adopt the LSTM network as the main architecture to build the prediction model.

For our task, the input at each time t is the observed popularity value v_t^i. In this case, $\boldsymbol{x}_t$ degenerates into a scalar value. Our task is specified by two numbers n and m. For item i, when LSTM receives n input values, it makes the prediction of m-step incremental value $\hat{v}_{n,m}^i$ using a function $g(\cdot)$ conditioned on the n-th hidden state vector $\boldsymbol{h}_n^i \in \mathbb{R}^L$ of item i. Formally, we have

$$\hat{v}_{n,m}^i = g(\boldsymbol{h}_n^i), \tag{2}$$

where the superscript of i indicates item i and $g(\cdot)$ is set to a linear function.

4.2 Enhancing Prediction with KB Embeddings

The above prediction model mainly captures the temporal correlation or dependence in time series data. In our setting, we also have KB data available, which contains potentially useful information for popularity prediction. Next, we study how to integrate KB information into the prediction model.

4.2.1 Knowledge Base Embedding. Given an online item i, let e_i denote its corresponding entity in KB. Since KB is originally framed as a set of triples, we can obtain a set of related triples where e_i plays the head or tail entity. Using the related triples, the first solution is to represent each e_i by a one-hot relation-based vector. However, such a feature vector has a large dimension size and is usually sparse. For effectively encoding KB information for e_i, we propose to learn a distributed vector $\boldsymbol{e}_i \in \mathbb{R}^D$. To learn KB embedding, we use the commonly used model TransE [2] to minimize the loss of the triples

$$\sum_{\{\langle e_1, r, e_2\rangle\}} \| \boldsymbol{e}_1 + \boldsymbol{r} - \boldsymbol{e}_2 \| . \tag{3}$$

We train the TransE model using all the triples in KB instead of using only those related to linked entities. The learned KB embedding provides a general and compact representation for KB information, which is more flexible to use and integrate.

4.2.2 Gate-based Integration of KB Embeddings. Now, we study how to integrate KB embedding into the LSTM-based prediction model. For popularity prediction, KB embedding is likely to contain both useful and irrelevant information, even noise. It may not work well to directly incorporate the KB embedding into the prediction model. To leverage KB embedding $\boldsymbol{e}_i$, we first transform it into a vector that is more suitable for the current task

$$\tilde{\boldsymbol{e}}_i = \text{MLP}(\boldsymbol{e}_i), \tag{4}$$

where $\text{MLP}(\cdot)$ is a standard Multi-layer Perceptron containing two hidden layers and using *relu* as the activation function in our work.

For item i, we have both the hidden state vector $\boldsymbol{h}_n^i$ learned from time series data and the transformed embedding $\tilde{\boldsymbol{e}}_i$ learned from KB data. We need to consider how to effectively combine these two vectors. Instead of setting a fixed weight, the model should be able to adaptively tune the combination weight based on the current state. To achieve this, we adopt the gate mechanism to combine the transformed KB embedding $\tilde{\boldsymbol{e}}_i$ and hidden state vector $\boldsymbol{h}_n^i$

$$z_n^i = \text{sigmoid}(\boldsymbol{W}^E \tilde{\boldsymbol{e}}_i + \boldsymbol{U}^E \boldsymbol{h}_n^i), \tag{5}$$

$$\tilde{\boldsymbol{h}}_n^i = z_n^i \cdot \tilde{\boldsymbol{e}}_i + (1 - z_n^i) \cdot \boldsymbol{h}_n^i, \tag{6}$$

where $z_n^i \in (0, 1)$ is the adaptive combination weight, $\boldsymbol{W}^E$ and $\boldsymbol{U}^E$ are parameter matrices, and $\tilde{\boldsymbol{h}}_n^i$ is the KB-enhanced representation of item i at the n-th time step.

In our model, we first adopt nonlinear transformation to learn suitable representations of KB embedding for popularity prediction. Then, the gate mechanism tries to balance the two factors conditioned on the current hidden state. A benefit of the gate-based combination method is that even for the same item we can have different combination weights at varying time steps, adaptively integrating KB information.

4.3 Enhancing Prediction with KB Neighbors

Till now, our prediction model only utilizes the information from the target item itself. Figure 1(a) shows that two items with similar KB information are likely to have similar popularity dynamics. Hence, we further propose to incorporate the popularity dynamics of related items with similar entity information to improve popularity prediction. For convenience, we call two items in the same domain with similar KB information *KB neighbors*. Now, our problems become how to identify KB neighbors and integrate the information of KB neighbors for popularity prediction.

4.3.1 KB Neighbor Identification. To measure the relatedness (or similarity) between two items using KB data, an intuitive idea is to compute the path reachability over the KB graph. However, the KB graph is usually very huge, and it is item-consuming to run graph search algorithms for each individual entity. Based on the learned KB embedding, we propose to compute the distance between entity embeddings for measuring item relatedness. Formally, given two entities e_1 and e_2, we compute the KB embedding distance via a distance function $f(\boldsymbol{e}_1, \boldsymbol{e}_2)$, where $f(\cdot)$ can be flexibly set to any distance function for vectors, *e.g.,* cosine and L_1 norm. In this way, we can rank the candidate items ascendingly by their KB embedding distance with the target item. Our idea is similar to that in [20]. They select the neighbors based on historical popularity trends, highly relying on the training set; while we select the neighbors using KB information, independent of historical popularity trends. In order to reduce data dependency, we further remove all the candidate items which has a prolonging lifetime with the target entity. For efficiency consideration, we only keep top K related entities of the entity type as KB neighbors.

4.3.2 Attention-based Integration of KB Neighbors. With the identified KB neighbors, we next describe how to utilize the information of KB neighbors for improving the prediction. Given a target item, its KB neighbors are likely to share similar popularity dynamics. Hence, it is intuitive to incorporate their popularity dynamics into the prediction model. For each neighbor k, we still use the LSTM network to encode their popularity dynamics up to the n-th time step into a hidden vector

$$\boldsymbol{h}_n^k = LSTM(\{v_1^k, \cdots, v_n^k\}; \Theta'), \tag{7}$$

where we use a different configuration Θ' for the LSTM network compared with the one for encoding the target item, because $\boldsymbol{h}_n^k$s are mainly used to improve the prediction for item i instead of item k itself. To integrate multiple hidden vectors of KB neighbors, we adopt the attention mechanism [1] to set the summation weights $\{\alpha_k^i\}$ conditioned on item i. Formally, α_k^i is defined as follows

$$\alpha_k^i = \frac{\exp(w(\boldsymbol{h}_n^i, \tilde{\boldsymbol{e}}_k))}{\sum_{k'=1}^{K}(\exp(w(\boldsymbol{h}_n^i, \tilde{\boldsymbol{e}}_{k'}))}, \tag{8}$$

where $\boldsymbol{h}_n^i$ is the derived hidden state vector of item i using only time series data, $\tilde{\boldsymbol{e}}_k$ is the transformed KB embedding of item k, and $w(\boldsymbol{h}_n^i, \tilde{\boldsymbol{e}_k})$ is set using the following function

$$w(\boldsymbol{h}_n^i, \tilde{\boldsymbol{e}}_k) = \boldsymbol{a}^\top \tanh(\boldsymbol{W}^N \boldsymbol{h}_n^i + \boldsymbol{U}^N \tilde{\boldsymbol{e}}_k), \tag{9}$$

where $\boldsymbol{W}^N$ and $\boldsymbol{U}^N$ are the parameter matrices, and $\boldsymbol{a}$ is the parameter vector. With the obtained attentive weights, we can encode the information from the K KB neighbors into a unique vector $\check{\boldsymbol{h}}_n^i$:

$$\check{\boldsymbol{h}}_n^i = \sum_{k=1}^{K} \alpha_k^i \cdot \boldsymbol{h}_n^k. \tag{10}$$

Finally, our item representation $\boldsymbol{s}_n^i$ for popularity prediction is a vector concatenation of $\tilde{\boldsymbol{h}}_n^i$ and $\check{\boldsymbol{h}}_n^i$,

$$\boldsymbol{s}_n^i = \tilde{\boldsymbol{h}}_n^i \oplus \check{\boldsymbol{h}}_n^i, \tag{11}$$

where $\tilde{\boldsymbol{h}}_n^i$ is the representation learned using only the information from the item itself (including both time series and KB data) defined in Eq. 6 and $\check{\boldsymbol{h}}_n^i$ is the representation learned using K KB neighbors defined in Eq. 10. Similar

to the gate mechanism in Section 4.2, our model adaptively set the attention weights conditioned on the current hidden state. It is able to alleviate the problem in Fig. 1(a), in which items have different popularity dynamics correlation at varying time steps. The attention mechanism can be viewed a key-value retrieval procedure, where the query $\boldsymbol{h}_n^i$ is the time series representation of the target item, and the keys $\{\tilde{\boldsymbol{e}}_k\}_{k=1}^K$ are the transformed KB embeddings of KB neighbors. The derived result is the attentive combination of time series representations of KB neighbors $\{\boldsymbol{h}_n^k\}_{k=1}^K$. By using non-linear query-key matching mechanism in Eq. 15, our model is more capable of inferring the usefulness of each KB neighbor for the target item. Since we filter out neighbors with a prolonging lifetime, our model will not use any information after the observed window. After obtaining $\boldsymbol{s}_n^i$, we still adopt the linear function $g(\boldsymbol{s}_n^i)$ to generate the final prediction.

We present the overall schematic diagram of the proposed model in Fig. 2. It is clear to see that the model consists of two parts: one utilizes the information of the target item itself, and the other utilizes the information of its KB neighbors. KB information is used in both aspects. First, it is transformed as a direct signal to enhance the prediction; second, it is used as the keys of the attention module. We call the proposed model *KB-enhanced Popularity Prediction Network* (**KB-PPN**).

4.4 Model Learning

We define the loss over the training set as follows

$$L = \sum_{i \in \mathcal{I}} \sum_{t \in [1,n]} \ell(v_{t,m}^i, \hat{v}_{t,m}^i), \tag{12}$$

where $\mathcal{D}$ is the item set, $v_{t,m}^i$ and $\hat{v}_{t,m}^i$ are the ground-truth or predicted incremental popularity value for item i in the time span $(t, t+m]$, and $\ell(\cdot)$ is the loss function, which is set to *Mean Absolution Error*. We learn our model parameters by using mini-batch gradient descent with the Adam optimizer.

5 EXPERIMENTS AND ANALYSIS

This section present the experiment setup and result analysis.

5.1 Experimental Setup

5.1.1 Construction of the Datasets. In our task, we need to prepare both KB and popularity time series data. For KB data, we adopt the one-time FREEBASE [7] dump consists of 63 million triples. For popularity time series data, we use three item popularity datasets, namely LAST.FM music [24], MOVIELENS movie [9] and AMAZON book [10]. To measure the popularity value, for the music dataset, we use the listening count, while for the other two datasets, we use the number of received ratings. It is uninteresting to predict the popularity of items with either a small popularity value or a short lifespan. We rank the items by its total popularity, and then select top items covering at least 40% of the entire time span of the dataset. Then, we link online items with FREEBASE entities. With an offline FREEBASE search API, we retrieve book entities with the ISBN number as queries, and retrieve music or movie entities with the item title as queries. It is likely to return multiple matched entities, and we further incorporate other attribute information (*e.g.,* singer or director) to filter the candidates until a single matched entity has been left. For the three datasets, we find 62.5%, 77.1% and 55.7% *selected items* can be linked to FREEBASE entities respectively. To train TRANSE, we start with linked entities as seeds and expand the graph with one-step search. In order to exclude temporal evidence from KB, we remove all the triples related to a temporal relation (*e.g.,* RELEASEDATE) together with the entities in the triples. By inspecting into the original datasets, we have found that a linked item has a larger popularity than an non-linked item on average, which indicates the more popular an item is, the more likely it is to be included in KBs. For the music dataset, we use a month as a time step; while the other datasets are much more sparse, we use a year as a time step. For each linked item, we build the popularity time series by time steps. Following [20], we construct the ten-fold cross validation for evaluation. The final results are averaged from ten runs. We summarize the detailed statistics of the three linked datasets after filtering in Table 2.

Table 2: Statistics of our datasets. APPS denotes the average popularity value per step.

Datasets	#selcted	#linked	Time span (yr.)	APPS
Music	37,000	23,120	2006-2014	577
Movie	12,000	9,260	1995-2015	277
Book	4,000	2,228	1997-2014	41

5.1.2 Evaluation Metrics. Following [20], we adopt three standard measurements as evaluation metrics:

- Mean Absolute Percentage Error (*MAPE*) measures the average derivation between the predicted and observed popularity over all items, defined as

$$MAPE = \frac{1}{N} \sum_{i=1}^{N} |\frac{\hat{v}_{n,m}^i - v_{n,m}^i}{v_{n,m}^i}|. \tag{13}$$

- *Accuracy* measures the fraction of items correctly predicted for a given error tolerance ϵ, defined as

$$ACC = \frac{1}{N} \sum_{i=1}^{N} |\{|\frac{\hat{v}_{n,m}^i - v_{n,m}^i}{v_{n,m}^i}| < \epsilon\}|. \tag{14}$$

  We set the threshold $\epsilon = 0.15$ in this paper.
- Mean Relative Squared Error (*mRSE*) measures the relative error between the predicted and observed popularity, defined as

$$mRSE = \frac{1}{N} \sum_{i=1}^{N} \big(\frac{\hat{v}_{n,m}^i}{v_{n,m}^i} - 1\big)^2. \tag{15}$$

5.1.3 Comparison Methods. The comparison methods are as follows.

- Multivariate Linear Regression (MLR) [20]: it predicts the popularity of an item using a linear combination of previous popularity values.

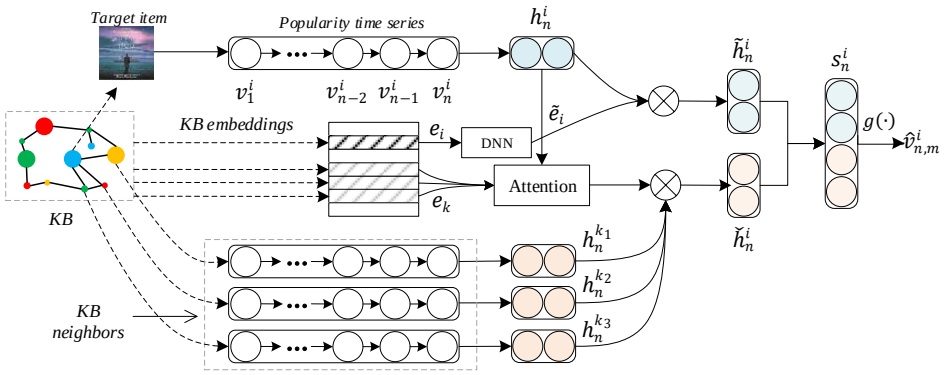


Figure 2: The overall schematic diagram of the proposed model.

- MRBF [20]: it is an extension of the basic MLR model by considering the similarity between the item and known examples from training set.
- Support Vector Regression (SVR) [15]: Khosla *et.al.* adopt SVR model using linear kernel to predict popularity with time series data as features.
- Random Forest (RF): We use RF to predict the popularity of online articles. RF adopt a tree-based ensemble approaching to capture complex data characteristics.
- Long Short-Term Memory (LSTM) [11]: LSTM improves RNN with a better capacity of encoding long sequences.
- The State Frequency Memory (SFM) [34]: it is the extension of the basic LSTM model by capturing multi-frequency time series patterns, which is a recently published work on dynamics prediction.
- Our model: we prepare three variants of our model: (1) only using KB embedding in Eq. 6, denoted by KB-PPN$_{+E}$, (2) using both KB embedding in Eq. 6 and KB neighbors in Eq. 7 without attention, denoted by KB-PPN$_{+E+N}$, and (3) our full model in Eq. 11, denoted by KB-PPN$_{full}$.

For MLR, SVR and LSTM, we also implement the corresponding variants by directly integrating KB embedding as features, denoted by MLR$_{+E}$, SVR$_{+E}$ and LSTM$_{+E}$ (using simple concatenation).

5.1.4 Parameter Setting. All the models have some parameters to tune. We either follow the reported optimal parameter settings or optimize each model separately using 10-fold cross validation. For all the neural network models, following [16], we vary the hidden layer size L in $\{32, 64, \cdots, 512\}$, the number of hidden layers in $\{1, 2, 3\}$, the activation function in {relu, tanh, sigmoid}, the batch size in $\{64, 128, \cdots, 1024\}$, and the initial learning rate in $\{0.02, 0.01, \cdots, 10^{-4}\}$. The embedding size of TransE D is selected from $\{50, 100, \cdots, 300\}$, and the number of neighbors K is selected from $\{1, 2, 3, 4, 5\}$. In order to avoid over-fitting, the dropout rate is chosen from $\{0.2, 0.3, ..., 0.8\}$.

5.2 Results and Analysis

We present the main comparison results of different methods for popularity prediction in Table 3. Since different datasets have varying time spans and popularity scales, we set different values for n and m[3]. For ease of result analysis, we categorize the comparison methods into two groups, namely traditional methods and neural network methods.

Among all the traditional baselines, SVR and RF perform better than the others, since they adopt more powerful modeling mechanisms (*i.e.,* margin-based optimization or tree-based non-linear transformation) and are likely to yield better performance. Another interesting observation is that the improvement of KB embedding using MLR is smaller than that of using SVR and RF. It indicates that the learned embedding may not be directly useful in linear prediction models, which confirms to our intuition.

For neural network models, SFM achieves a better performance than LSTM, since it is able to capture multi-frequency time series patterns. Next, we examine the the effect of different ways to integrate KB embeddings of the target item. It is clear to see that KB-PPN$_{+E}$ is substantially better than LSTM$_{+E}$. The major difference is that our model KB-PPN$_{+E}$ adopts a gate mechanism to integrate KB embedding, while LSTM$_{+E}$ simply adopts a vector concatenation, which is less effective to utilize the information of KB embedding. By additionally integrating the information of KB neighbors, KB-PPN$_{+E+N}$ and KB-PPN$_{full}$ perform better than all the other comparison methods, which indicates the usefulness of KB neighbors. While, KB-PPN$_{full}$ further improves over KB-PPN$_{+E+N}$ due to the use of attention mechanisms. The above findings show that the KB information is useful to improve popularity prediction. In particular, the integration way of these information is important to the final performance.

Recall our task is parameterized with two additional numbers of n and m. A good prediction model should be able to work well in various cases of n and m. To examine the performance stability of our model, we vary n and m alternatively, and compare our method with baselines. Due to space limit,

[3]Recall n is the number of previous steps (seen) and m is the number of future steps (predicted).

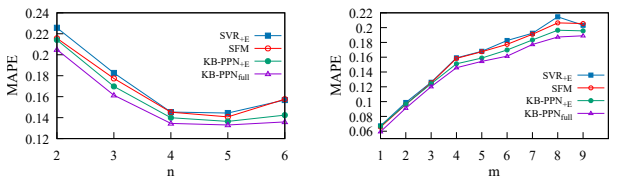
Table 3: Performance comparisons of different methods on popularity prediction. "↓ / ↑" indicate smaller is better or worse. "†" ,"#" indicates the improvement of $KB-PPN_{+E}$ and $KB-PPN_{full}$ over the baseline is significant at the level of 0.01. We also report the improvement ratios over the comparison baselines from $KB-PPN_{+E}$ and $KB-PPN_{full}$ in parentheses.

Dataset	Models	MAPE (↓)	ACC (↑)	MRSE (↓)
Music ($n=3, m=9$)	MLR	$0.212^{\dagger,\#}$ (7.55%, 10.8%)	$0.450^{\dagger,\#}$ (8.44%, 11.3%)	$0.079^{\dagger,\#}$ (13.9%, 21.5%)
	MLR_{+E}	$0.211^{\dagger,\#}$ (7.11%, 10.4%)	$0.456^{\dagger,\#}$ (7.02%, 9.87%)	$0.078^{\dagger,\#}$ (12.8%, 20.5%)
	MRBF	$0.210^{\dagger,\#}$ (6.67%, 10.0%)	$0.461^{\dagger,\#}$ (5.86%, 8.68%)	$0.079^{\dagger,\#}$ (13.9%, 21.5%)
	SVR	$0.209^{\dagger,\#}$ (6.22%, 9.57%)	$0.461^{\dagger,\#}$ (5.86%.8.68%)	$0.078^{\dagger,\#}$ (12.8%, 20.5%)
	SVR_{+E}	$0.204^{\dagger,\#}$ (3.92%, 7.35%)	$0.471^{\dagger,\#}$ (3.61%, 6.37%)	$0.074^{\dagger,\#}$ (8.10%, 16.2%)
	RF	$0.206^{\dagger,\#}$ (4.85%, 8.25%)	$0.466^{\dagger,\#}$ (4.72%, 7.51%)	$0.076^{\dagger,\#}$ (10.5%, 18.4%)
	RF_{+E}	$0.206^{\dagger,\#}$ (4.85%, 8.25%)	$0.468^{\dagger,\#}$ (4.27%, 7.05%)	$0.075^{\dagger,\#}$ (9.33%, 17.3%)
	LSTM	$0.208^{\dagger,\#}$ (5.77%, 9.13%)	$0.458^{\dagger,\#}$ (6.55%, 9.39%)	$0.076^{\dagger,\#}$ (10.5%, 18.4%)
	SFM	$0.206^{\dagger,\#}$ (4.85%, 8.25%)	$0.469^{\dagger,\#}$ (4.05%, 6.82%)	$0.075^{\dagger,\#}$ (9.33%, 17.3%)
	$LSTM_{+E}$	$0.205^{\dagger,\#}$ (4.39%, 7.80%)	$0.469^{\dagger,\#}$ (4.05%, 6.82%)	$0.075^{\dagger,\#}$ (9.33%, 17.3%)
	$KB\text{-}PPN_{+E}$	0.196	0.488	0.068
	$KB\text{-}PPN_{+E+N}$	0.193	0.494	0.066
	$KB\text{-}PPN_{full}$	**0.189**	**0.501**	**0.062**
Movie ($n=2, m=4$)	MLR	$0.222^{\dagger,\#}$ (15.3%, 18.0%)	$0.427^{\dagger,\#}$ (16.9%, 18.3%)	$0.080^{\dagger,\#}$ (27.5%, 32.5%)
	MLR_{+E}	$0.222^{\dagger,\#}$ (15.3%, 18.0%)	$0.427^{\dagger,\#}$ (16.9%, 18.3%)	$0.080^{\dagger,\#}$ (27.5%, 32.5%)
	MRBF	$0.212^{\dagger,\#}$ (11.3%, 14.2%)	$0.452^{\dagger,\#}$ (10.4%, 11.7%)	$0.075^{\dagger,\#}$ (22.7%, 28.0%)
	SVR	$0.206^{\dagger,\#}$ (8.74%, 11.7%)	$0.460^{\dagger,\#}$ (8.48%, 9.78%)	$0.070^{\dagger,\#}$ (17.1%, 22.9%)
	SVR_{+E}	$0.204^{\dagger,\#}$ (7.84%, 10.8%)	$0.463^{\dagger,\#}$ (7.78%, 9.07%)	$0.068^{\dagger,\#}$ (14.7%, 20.6%)
	RF	$0.211^{\dagger,\#}$ (10.9%, 13.7%)	$0.460^{\dagger,\#}$ (8.48%, 9.78%)	$0.078^{\dagger,\#}$ (25.6%, 30.1%)
	RF_{+E}	$0.209^{\dagger,\#}$ (10.1%, 12.9%)	$0.460^{\dagger,\#}$ (8.48%, 9.78%)	$0.077^{\dagger,\#}$ (24.7%, 29.9%)
	LSTM	$0.196^{\dagger,\#}$ (4.08%, 7.14%)	$0.488^{\dagger,\#}$ (2.25%, 3.48%)	$0.064^{\dagger,\#}$ (9.37%, 15.6%)
	SFM	$0.195^{\dagger,\#}$ (3.59%, 6.67%)	$0.482^{\dagger,\#}$ (3.53%, 4.77%)	$0.063^{\dagger,\#}$ (7.94%, 14.3%)
	$LSTM_{+E}$	$0.192^{\dagger,\#}$ (2.08%, 5.21%)	$0.495^{\dagger,\#}$ (0.81%, 2.02%)	$0.062^{\dagger,\#}$ (6.45%, 12.9%)
	$KB\text{-}PPN_{+E}$	0.188	0.499	0.058
	$KB\text{-}PPN_{+E+N}$	0.189	0.491	0.059
	$KB\text{-}PPN_{full}$	**0.182**	**0.505**	**0.054**
Book ($n=2, m=4$)	MLR	$0.288^{\dagger,\#}$ (16.0%, 19.4%)	$0.324^{\dagger,\#}$ (19.1%, 22.5%)	$0.130^{\dagger,\#}$ (27.7%, 34.6%)
	MLR_{+E}	$0.273^{\dagger,\#}$ (11.4%, 15.0%)	$0.344^{\dagger,\#}$ (12.2%, 15.4%)	$0.119^{\dagger,\#}$ (21.0%, 28.6%)
	MRBF	$0.272^{\dagger,\#}$ (11.0%, 14.7%)	$0.341^{\dagger,\#}$ (13.2%, 16.4%)	$0.115^{\dagger,\#}$ (18.3%, 26.1%)
	SVR	$0.269^{\dagger,\#}$ (10.0%, 13.8%)	$0.337^{\dagger,\#}$ (14.5%, 17.8%)	$0.111^{\dagger,\#}$ (15.3%, 23.4%)
	SVR_{+E}	$0.257^{\dagger,\#}$ (5.84%, 9.73%)	$0.361^{\dagger,\#}$ (6.93%, 9.97%)	$0.105^{\dagger,\#}$ (10.5%, 19.1%)
	RF	$0.261^{\dagger,\#}$ (7.28%, 11.1%)	$0.367^{\dagger,\#}$ (5.18%, 8.17%)	$0.112^{\dagger,\#}$ (16.1%, 24.1%)
	RF_{+E}	$0.257^{\dagger,\#}$ (5.84%, 9.73%)	$0.360^{\dagger,\#}$ (7.22%, 10.3%)	$0.106^{\dagger,\#}$ (11.3%, 19.8%)
	LSTM	$0.261^{\dagger,\#}$ (7.28%, 11.1%)	$0.348^{\dagger,\#}$ (10.9%, 14.1%)	$0.104^{\dagger,\#}$ (9.62%, 18.3%)
	SFM	$0.256^{\dagger,\#}$ (5.47%, 9.37%)	$0.355^{\dagger,\#}$ (8.73%, 11.8%)	$0.101^{\dagger,\#}$ (6.93%, 15.8%)
	$LSTM_{+E}$	$0.246^{\dagger,\#}$ (1.63%, 5.69%)	$0.374^{\dagger,\#}$ (3.21%, 6.15%)	$0.096^{\dagger,\#}$ (2.08%, 11.5%)
	$KB\text{-}PPN_{+E}$	0.242	0.386	0.094
	$KB\text{-}PPN_{+E+N}$	0.238	0.391	0.090
	$KB\text{-}PPN_{full}$	**0.232**	**0.397**	**0.085**

we only report the tuning results on the music dataset. As shown in Fig. 3, we can see that our model is consistently better than the selected baselines in various cases, indicating the robustness of the proposed model. Our model $KB\text{-}PPN_{full}$ have several parameters to tune, including the number of KB neighbors K, the KB embedding size D and the hidden layer size L. We find the number of KB neighbors should be set to a small value. In our experiments $K=3$ yields the best performance. Next, we tune D and L alternatively. In Fig. 4, we can see $D=100$ and $L=128$ gives the best performance. Overall, the performance of our model is relatively stable by varying L and D, consistently better than the baselines.

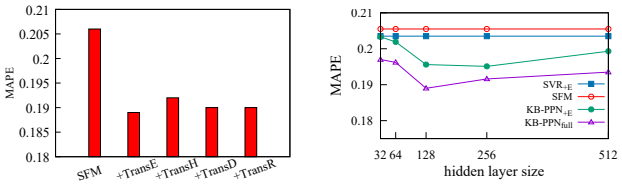
6 CONCLUSION

In this paper, we proposed to heuristically link online items with existing KB entities, and leverage KB information for improving popularity prediction. Our experiment results showed that both KB embedding of the target item and popularity dynamics of its KB neighbors are useful to enhance the prediction. As future work, we will test the proposed approach in more domains. Since not all the entities can find corresponding KB entries, it will be interesting to study how to enhance the prediction performance of non-linked items with KBs. We will also extend the base LSTM network to more complicated network such as SFM [34].



(a) Varying n ($m = 6$). (b) Varying m ($n = 3$).

Figure 3: Performance tuning by varying the number of previous steps n and the number of future steps m.



(a) Varying D ($L = 128$). (b) Varying L ($D = 100$).

Figure 4: Performance tuning by varying the KB embedding size D and the hidden layer size L.

ACKNOWLEDGEMENT

The work was partially supported by National Natural Science Foundation of China under the Grant Number 61502502, Beijing Natural Science Foundation under the Grant Number 4162032, the National Key Basic Research Program (973 Program) of China under Grant No. 2014CB340403, and the Fundamental Research Funds for the Central Universities and the Research Funds of Renmin University of China under Grant 18XNLG22.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR* abs/1409.0473 (2014).
[2] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*. 2787–2795.
[3] Qi Cao, Huawei Shen, Keting Cen, Wentao Ouyang, and Xueqi Cheng. 2017. DeepHawkes: Bridging the Gap between Prediction and Understanding of Information Cascades. In *CIKM*. 1149–1158.
[4] Biao Chang, Hengshu Zhu, Yong Ge, Enhong Chen, Hui Xiong, and Chang Tan. 2014. Predicting the Popularity of Online Serials with Autoregressive Models. In *CIKM*. 1339–1348.
[5] Justin Cheng, Lada Adamic, P. Alex Dow, Jon Michael Kleinberg, and Jure Leskovec. 2014. Can cascades be predicted?. In *WWW*. 925–936.
[6] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep learning for event-driven stock prediction. In *IJCAI*. 2327–2333.
[7] Google. 2016. Freebase Data Dumps. https://developers.google.com/freebase/data. (2016).
[8] Aditya Grover, Ashish Kapoor, and Eric Horvitz. 2015. A Deep Hybrid Model for Weather Forecasting. In *SIGKDD*. 379–386.
[9] F. Maxwell Harper and Joseph A. Konstan. 2016. The MovieLens Datasets. *TiiS* 5, 4 (2016), 1–19.
[10] Ruining He and Julian Mcauley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *WWW*. 507–517.
[11] Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[12] Daichi Imamori and Keishi Tajima. 2016. Predicting Popularity of Twitter Accounts through the Discovery of Link-Propagating Early Adopters. In *CIKM*. 639–648.
[13] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge Graph Embedding via Dynamic Mapping Matrix. In *ACL*. 687–696.
[14] Xiaowei Jia, Ankush Khandelwal, Guruprasad Nayak, James Gerber, Kimberly Carlson, Paul West, and Vipin Kumar. 2017. Incremental Dual-memory LSTM in Land Cover Prediction. In *SIGKDD*.
[15] Aditya Khosla, Atish Das Sarma, and Raffay Hamid. 2014. What makes an image popular?. In *WWW*. 867–876.
[16] Cheng Li, Jiaqi Ma, Xiaoxiao Guo, and Qiaozhu Mei. 2016. DeepCas: An End-to-end Predictor of Information Cascades. (2016), 577–586.
[17] Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2016. Knowledge Representation Learning with Entities, Attributes and Relations. In *IJCAI*. 2866–2872.
[18] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *AAAI*. 2181–2187.
[19] Changsha Ma, Zhisheng Yan, and Chang Wen Chen. 2017. LARM: A Lifetime Aware Regression Model for Predicting YouTube Video Popularity. In *CIKM*. 467–476.
[20] Henrique Pinto and Jussara M. Almeida. 2013. Using early view patterns to predict the popularity of youtube videos. In *WSDM*. 365–374.
[21] Julia Proskurnia, Przemyslaw A. Grabowicz, Ryota Kobayashi, Carlos Castillo, Philippe Cudré-Mauroux, and Karl Aberer. 2017. Predicting the Success of Online Petitions Leveraging Multidimensional Time-Series. In *WWW*. 755–764.
[22] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. 2017. A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. In *IJCAI*. 2627–2633.
[23] Suman Deb Roy, Tao Mei, Wenjun Zeng, and Shipeng Li. 2013. Towards Cross-Domain Learning for Social Video Popularity Prediction. *IEEE Transactions on Multimedia* 15, 6 (2013), 1255–1267.
[24] Markus Schedl. 2016. The LFM-1b Dataset for Music Retrieval and Recommendation. In *ICMR*.
[25] Hua Wei Shen, Dashun Wang, Chaoming Song, and Albertlszl Barabsi. 2014. Modeling and Predicting Popularity Dynamics via Reinforced Poisson Processes. In *AAAI*. 291–297.
[26] Xingjian Shi, Zhihan Gao, Leonard Lausen, Hao Wang, Dit Yan Yeung, Waikin Wong, and Wangchun Woo. 2017. Deep Learning for Precipitation Nowcasting: A Benchmark and A New Model. (2017).
[27] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW*. 697–706.
[28] Gabor Szabo and Bernardo A Huberman. 2008. Predicting the popularity of online content. *Communications of the Acm* 53, 8 (2008), 80–88.
[29] Alexandru-Florin Tatar, Marcelo Dias de Amorim, Serge Fdida, and Panayotis Antoniadis. 2014. A survey on predicting the popularity of web content. *J. Internet Services and Applications* 5, 1 (2014), 8:1–8:20.
[30] Yongqing Wang, Shenghua Liu, Huawei Shen, Jinhua Gao, and Xueqi Cheng. 2017. Marked Temporal Dynamics Modeling Based on Recurrent Neural Network. (2017).
[31] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *AAAI*. 1112–1119.
[32] Bo Wu, Wen-Huang Cheng, Yongdong Zhang, Qiushi Huang, Jintao Li, and Tao Mei. 2017. Sequential Prediction of Social Media Popularity with Deep Temporal Context Networks. In *IJCAI*. 3062–3068.
[33] Bo Wu, Tao Mei, Wen Huang Cheng, and Yongdong Zhang. 2016. Unfolding temporal dynamics: predicting social media popularity using multi-scale temporal decomposition. In *AAAI*. 272–278.
[34] Liheng Zhang, Charu Aggarwal, and Guo Jun Qi. 2017. Stock Price Prediction via Discovering Multi-Frequency Trading Patterns. In *SIGKDD*.