

Knowledge-based Temporal Fusion Network for Interpretable Online Video Popularity Prediction

Shisong Tang[1†], Qing Li[2*], Xiaoteng Ma[1], Ci Gao[3], Dingmin Wang[4]
Yong Jiang[12], Qian Ma[5], Aoyang Zhang[5], Hechang Chen[3]
[1] Shenzhen International Graduate School, Tsinghua University, China
[2]Peng Cheng Laboratory, China
[3]Jilin University, China
[4]University of Oxford, United Kingdom
[5]ByteDance Inc., China
{tangss21,maxt17}@mails.tsinghua.edu.cn,liq@pcl.ac.cn,gaoci21@mails.jlu.edu.cn,chenhc@jlu.edu.cn
dingmin.wang@cs.ox.ac.uk,jiangy@sz.tsinghua.edu.cn,{maqian.zero,zhangaoyang}@bytedance.com

ABSTRACT

Predicting the popularity of online videos has many real-world applications, such as recommendation, precise advertising, and edge caching strategies. Despite many efforts have been dedicated to the online video popularity prediction, there still exist several challenges: (1) The meta-data from online videos is usually sparse and noisy, which makes it difficult to learn a stable and robust representation. (2) The influence of content features and temporal features in different life cycles of online videos is dynamically changing, so it is necessary to build a model that can capture the dynamics. (3) Besides, there is a great need to interpret the predictive behavior of the model to assist administrators of video platforms in the subsequent decision-making.

In this paper, we propose a Knowledge-based Temporal Fusion Network (KTFN) that incorporates knowledge graph representation to address the aforementioned challenges in the task of online video popularity prediction. To be more specific, we design a Tree Attention Network (TAN) to learn the embedding of online video entities in knowledge graphs via selectively aggregating local neighborhood information, thus enabling our model to learn the importance of different entities under the same relation. Besides, an Attention-based Long Short-Term Memory (ALSTM) is utilized to learn the temporal feature representation. Finally, we propose an Adaptively Temporal Feature Fusion (ATFF) scheme to adaptively fuse content features and temporal features, in which a learnable exponential decay function with the global attention mechanism is constructed. We collect two large-scale real-world datasets from the server logs of two popular Chinese online video platforms, and experimental results on the two datasets have demonstrated the superiority and interpretability of KTFN.

∗ Corresponding author: Qing Li (liq@pcl.ac.cn).
† Work done in Bytedance.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WWW '22, April 25–29, 2022, Virtual Event, Lyon, France
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3511934

CCS CONCEPTS

• **Information systems** → **Multimedia information systems**.

KEYWORDS

Video popularity prediction, Knowledge graph, Attention model, Graph neural networks

ACM Reference Format:
Shisong Tang[1†], Qing Li[2*], Xiaoteng Ma[1], Ci Gao[3], Dingmin Wang[4], Yong Jiang[12], Qian Ma[5], Aoyang Zhang[5], Hechang Chen[3]. 2022. Knowledge-based Temporal Fusion Network for Interpretable Online Video Popularity Prediction. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3485447.3511934

1 INTRODUCTION

With the prevalence of Web 2.0 and mobile devices, an increasing number of users are joining online video platforms such as Youtube[1], TikTok[2], and Douyin[3] for sharing and viewing. One of goals of video popularity prediction is to infer the cumulative views of a given video during a certain period in the future. This task can not only help users filter information, but also support many businesses of platform companies, such as precise advertising [6], recommendation [2], and edge caching strategies [33].

Traditional feature-based methods mainly leverage user features [30], content features [24], temporal features [20], and structural features [9] to conduct popularity prediction. However, such approaches heavily depend on well-designed hand-crafted features, which limits the models' scalability. Tang et al. [25] and Rizoiu et al. [23] performed video popularity prediction based on Hawkes process [10]. Although such approaches do not require excessive feature engineering, they usually make strong assumptions on fixed parameters, which limits model expressiveness [5, 33].

Recently, a large number of deep learning-based models have been proposed to improve the performance of popularity prediction [1, 4, 14, 15, 31]. However, for video popularity prediction, few works address the following three challenges simultaneously: (1) The meta-data from online videos is usually sparse and noisy,

[1]https://www.youtube.com/
[2]https://www.tiktok.com/
[3]https://www.douyin.com/

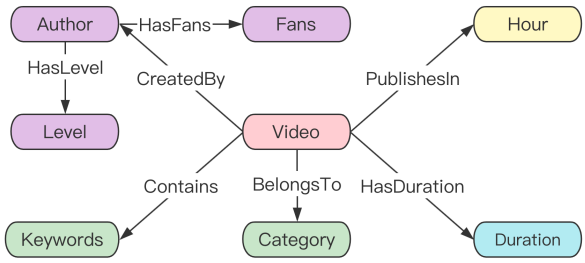


Figure 1: An example of video knowledge graph.

which makes it difficult to learn a stable and robust representation. (2) The influence of content features and temporal features is dynamically changing over the different life cycles of videos, which necessitates designing a flexible feature fusion scheme to capture the dynamics. (3) Apart from the improvement of model performance, it is also highly-demanded to design a model with high interpretability, which is quite beneficial for administrators of video platforms to make strategic decisions and manage their platform resources better.

To better extract content features from the sparse and noise meta-data and learn potential knowledge-level connections, it is a remedy to introduce the knowledge graph into video popularity prediction. A knowledge graph is a directed heterogeneous graph that describes the facts, where nodes correspond to entities and edges correspond to relations. Compared with traditional data formats, knowledge graphs can provide a general and compact context. Recently, researchers have successfully applied knowledge graphs to recommendation systems [28], language representation learning [17], and question answering [3].

Considering the aforementioned challenges of video popularity prediction and inspired by the success of knowledge graphs in different domains, we propose a Knowledge-based Temporal Fusion Network (KTFN) to perform online video popularity prediction. An example of video knowledge graph we constructed is shown in Figure 1. Following [16, 28], we exploit embedding approaches to learn vector representation for entities and relations in the knowledge graph. For a given video, we first search its set of contextual entities in the knowledge graph (i.e., its immediate neighbors in the knowledge graph). Then, we design a Tree Attention Network (TAN), which logically transforms a graph into a tree. The TAN performs information propagation and aggregation with the attention mechanism to learn the content feature vector of the video. For time-series data, we employ an Attention-based LSTM (ALSTM) [29] network to obtain the temporal feature vector of a video. The previous study [15] shows that different features have different impacts in different life stages of the video. To capture this property, we further propose an Adaptively Temporal Feature Fusion (ATFF) method. Specifically, we construct an information valve based on a learnable exponential decay function to filter the feature vector. And then the global attention mechanism is adopted to fuse the filtered content feature vector with the filtered temporal feature vector to obtain the final feature vector of the video. Our contributions are summarized as follows:

- We incorporate knowledge graphs into online video popularity prediction, which provides a compact context to learn the video content features from sparse and noisy meta-data.
- We propose TAN, an architecture based on graph neural networks, which converts graphs into trees to learn local neighborhood information of entities while improving the interpretability of the model.
- We construct a learnable exponential decay function and combine the global attention mechanism to adaptively fuse content features and temporal features.
- Extensive experiments on datasets collected from two large-scale video-sharing platforms demonstrate the superiority and interpretability of our porposed model.

The rest of this paper is organized as follows: Section 2 presents the related work of popularity prediction. The framework of KTFN is described in detail in Section 3. Experimental results and interpretable instances are shown and analyzed in Section 4. Section 5 concludes this paper.

2 RELATED WORK

The traditional popularity prediction methods are mainly divided into two categories, namely, feature-based methods and point process-based methods. Feature-based approaches have verified the predictive effectiveness of features including user features [30], content features [24], temporal features [20], and structural features [9], which can provide us with a relatively preliminary understanding and knowledge of predicting the popularity of items in the future. However, the features involved in such methods are usually extracted by heuristics and the final prediction performance of the model is highly dependent on the quality of these heuristics. The point process-based method regards the information dissemination process as an arrival process of the user's forwarding behavior. Zhao et al. [32] predicted the final number of retweets of a post using a self-exciting point process. Rizoiu et al. [23] proposed a combination of the Hawkes intensity process with exogenous stimuli and endogenous trigger effects from Twitter and YouTube to predict the popularity of videos. The point process-based approaches provide a well-defined generic framework for popularity prediction. However, their reliance on certain specific assumptions limits their generality and model expressiveness [5, 33].

In recent years, the prevalence of neural networks has inspired many deep learning-based prediction models. A typical deep learning approach is to employ Recurrent Neural Networks (RNN) to capture temporal dependencies [14, 19, 21]. Cao et al. [1] combined Hawkes processes with deep learning methods for popularity prediction to overcome the limitations of the simple parametric form on the capability of point process models. Zhang et al. [31] proposed a user-guided hierarchical attention network using the attention mechanism to learn modalities content and user features for image popularity prediction. The deep fusion of temporal processes and content features network was proposed by Liao et al. [15] to model multi-modal data for article popularity prediction. Dou et al. [4] exploited the embedding of knowledge base entities and their neighbors to enhance the popularity prediction based on LSTM networks. However, there currently fails to exist a work that leverages the sparse and noise meta-data of videos for popularity prediction. Moreover, many existing deep learning-based popularity predictions ignore the interpretability of the models.

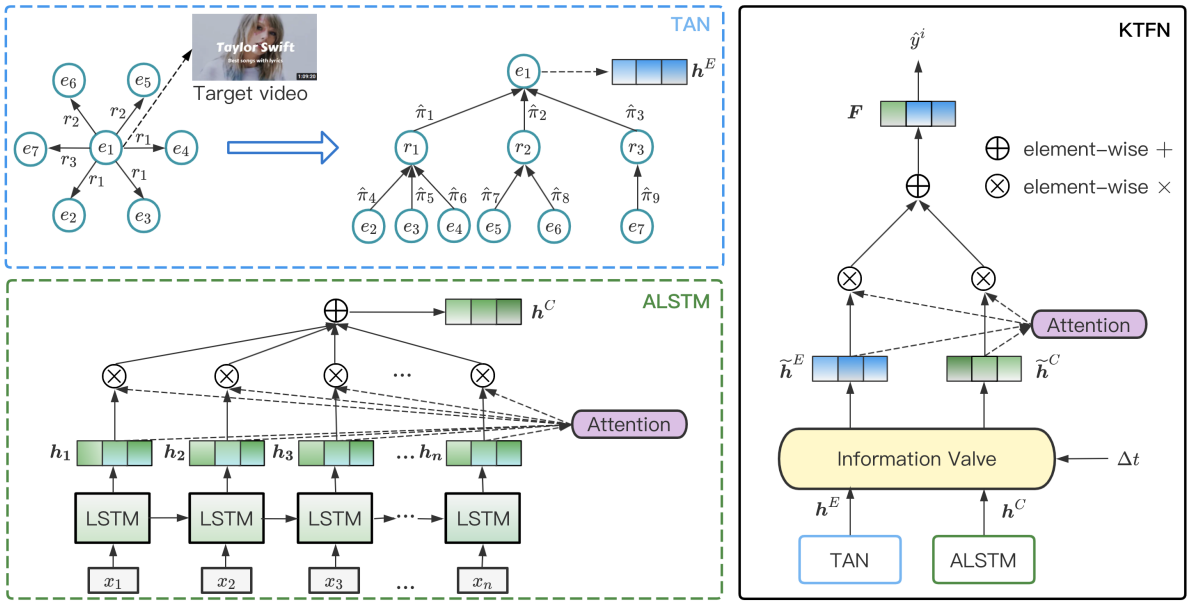


Figure 2: Illustration of the proposed KTFN model.

3 METHOD

In this section, we present the proposed KTFN model, whose framework is shown in Figure 2. We first formulate the knowledge-graph-based popularity prediction problem. Then we introduce the embedding layer, the Tree Attention Network, the Attention-based LSTM, and the structure of ATFF, respectively.

3.1 Problem Definition

We regard the knowledge-graph-based online video popularity prediction task as a regression problem. We discretize continuous time into time steps. For a given video i on the online video platform, its popularity sequence in n time steps is $X^i = (x_1^i, x_2^i, ..., x_n^i)$, where x_j^i represents the number of views of video i at the j-th time step. The prediction target $y^i = \sum_{t=n+1}^{n+m} x_t^i$ is the cumulative popularity of video i in m time steps after time n. In addition, we organize the meta-data of videos into knowledge graph $\mathcal{G}$, which is a heterogeneous graph composed of entity-relation-entity triples. Formally, $\mathcal{G}$ is presented as $\{(h, r, t)|h, t \in \mathcal{E}, r \in \mathcal{R}\}$, where $\mathcal{E}$ and $\mathcal{R}$ separately denote the set of entities and relations in the knowledge graph. For example, the triples $(Fearless, album.artist, TaylorSwift)$ states the fact that Taylor Swift writes the album "Fearless". In particular, a given video i is represented as one entity $e \in \mathcal{E}$.

3.2 Embedding Layer

Knowledge graph embedding maps entities and relations into low-dimensional representation vectors, in which the original graph structure and semantic information is encoded. To train knowledge graph embeddings, we use TransR model [16], which introduces a projection matrix $\boldsymbol{M}_r$ for each relation to map an entity from its own entity space to the corresponding relation space. For each triplet (h, r, t) in the knowledge, whose representation vectors are $\boldsymbol{h}$, $\boldsymbol{r}$ and $\boldsymbol{t}$, respectively, the embedding layer learns embeddings for entities and relations by optimizing translation principle $\boldsymbol{h_r} + \boldsymbol{r} \approx \boldsymbol{t_r}$, where $\boldsymbol{h_r} = \boldsymbol{M}_r\boldsymbol{h}$ and $\boldsymbol{t_r} = \boldsymbol{M}_r\boldsymbol{t}$. Hence, for a triple (h, r, t), its plausibility score is formulated as follows:

$$f_r(\boldsymbol{h}, \boldsymbol{t}) = ||\boldsymbol{M}_r\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{M}_r\boldsymbol{t}||_2^2. \quad (1)$$

The training of TransR considers both correct triples and incorrect triples, and encourages their discrimination through the following margin-based ranking loss:

$$\mathcal{L}_{KG} = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'} max(0, f_r(\boldsymbol{h}, \boldsymbol{t}) + \gamma - f_r(\boldsymbol{h'}, \boldsymbol{t'})), \quad (2)$$

where γ is the margin, S is the set of correct triples and S' is the set of incorrect triples.

3.3 Tree Attention Networks

Graph convolution network (GCN) [13] recursively propagates embeddings along high-order connectivity. Graph attention network (GAT) [27] uses masked attention to generate attentive weights for the first-order neighbors of the node. Despite the success of GCN and GAT, they are not suitable for dealing with the knowledge graph as it is a heterogeneous graph. To exploit the advantages of both and be better for handling the knowledge graph, we propose a Tree Attention Network that propagates embeddings in a bottom-up way.

3.3.1 Structure Transformation. For a given online video i, considering its corresponding entity h^i in knowledge graph, we use $\mathcal{G}_i = \{(h^i, r, t)|(h^i, r, t) \in \mathcal{G}\}$ to represent the set of triples where h^i is the head entity, which is termed as ego-network [22]. We convert relations of $\mathcal{G}_i$ into nodes and view neighbors under the same relation as a group, thus forming a tree structure, denoted by $\mathcal{T}_i$, where h^i is the root node.

3.3.2 Information Propagation. Here, we use $\mathcal{N}_k$ to represent the first-order neighbors of node k in $\mathcal{T}_i$. To characterize the neighboring topology of k, we calculate the linear combination of k's neighbors:

$$\boldsymbol{e}^k_{\mathcal{N}_k} = \sum_{j \in \mathcal{N}_k} \pi(\boldsymbol{e}_k, \boldsymbol{e}_j)\boldsymbol{e}_j, \tag{3}$$

where $\boldsymbol{e}_k$ and $\boldsymbol{e}_j$ denote the embedding of node k and j respectively, and $\pi(\boldsymbol{e}_k, \boldsymbol{e}_j)$ is the normalized attention score that controls how much information being propagated from node j to node k.

We implement $\pi(\boldsymbol{e}_k, \boldsymbol{e}_j)$ via attention mechanism, which can be computed in the following:

$$\pi(\boldsymbol{e}_k, \boldsymbol{e}_j) = \text{LeakyReLU}(\boldsymbol{W}_1(\boldsymbol{W}_2\boldsymbol{e}_k || \boldsymbol{W}_3\boldsymbol{e}_j)), \tag{4}$$

where we follow the way in [27] to select LeakyReLU [18] as the nonlinear activation function. $\boldsymbol{W}_1, \boldsymbol{W}_2$ and $\boldsymbol{W}_3$ are trainable parameters. Note that we have introduced two different linear transformation matrices for $\boldsymbol{e}_k$ and $\boldsymbol{e}_j$ because they are different type nodes (entity node, relation node). Hereafter, we normalize the attention scores across all nodes connected with k by adopting the softmax function:

$$\pi(\boldsymbol{e}_k, \boldsymbol{e}_j) = \frac{\exp(\pi(\boldsymbol{e}_k, \boldsymbol{e}_j))}{\sum_{s \in \mathcal{N}_k} \exp(\pi(\boldsymbol{e}_k, \boldsymbol{e}_s))}. \tag{5}$$

3.3.3 Information Aggregation. This step is to aggregate the node representation $\boldsymbol{e}_k$ and its corresponding neighborhood representation $\boldsymbol{e}^k_{\mathcal{N}_k}$ as the new representation of node k. Following [28], we use three methods to implement the aggregation function $f(\boldsymbol{e}_k, \boldsymbol{e}^k_{\mathcal{N}_k})$.

- *Sum aggregator* sums two representations up and uses a nonlinear transformation:

$$f_{sum} = g(\boldsymbol{W}_4(\boldsymbol{e}_k + \boldsymbol{e}^k_{\mathcal{N}_k})), \tag{6}$$

  where g is the nonlinear function such as LeakyReLU and $\boldsymbol{W}_4$ is the trainable weight matrix to transfer the current representations into the common space for propagation.
- *Concat aggregator* concatenates the two representations before applying a nonlinear function:

$$f_{concat} = g(\boldsymbol{W}_4(\boldsymbol{e}_k || \boldsymbol{e}^k_{\mathcal{N}_k})), \tag{7}$$

  where $||$ is the concatenation operation.
- *Bi-interaction aggregator* takes into account two kinds of two interactions between $\boldsymbol{e}_k$ and $\boldsymbol{e}^k_{\mathcal{N}_k}$:

$$f_{Bi-interaction} = g(\boldsymbol{W}_4(\boldsymbol{e}_k + \boldsymbol{e}^k_{\mathcal{N}_k})) + g(\boldsymbol{W}_5(\boldsymbol{e}_k \odot \boldsymbol{e}^k_{\mathcal{N}_k})), \tag{8}$$

  where $\odot$ denotes the Hadamard (element-wise) multiplication, and $\boldsymbol{W}_5$ is the trainable weight matrix.

3.3.4 Bottom-up Propagation. To obtain local neighborhood information of root node h^i and attention scores between each node, we use a bottom-up propagation strategy, which propagates and aggregates information from the leaf nodes to the root node of $\mathcal{T}_i$. TAN provides a fine-grained learning process, which allows our model to emphasize the importance of different entities under the same relation while improving the interpretability of our model.

3.4 Attention-based LSTM

There are two primary motivations for us to choose the Attention-based LSTM [29] modeling the temporal evolution process of popularity. First, as the most widely used Recurrent Neural Network (RNN) structure, LSTM [8] has the capability to model long-term historical information of temporal sequences. Second, the attention mechanism can not only capture important information of temporal sequences but also improve the interpretability of the model. Attention-based LSTM network combines both advantages well.

We firstly feed $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)$ into LSTM. With its gate mechanism, including memory gate, input gate, and forget gate, LSTM can remember what should be remembered and forget what should be forgotten. Formally, each cell in LSTM can be computed as follows:

$$i_t = \sigma(\boldsymbol{W}_i\boldsymbol{x}_t + \boldsymbol{U}_i\boldsymbol{c}_{t-1} + \boldsymbol{V}_i\boldsymbol{h}^c_{t-1} + \boldsymbol{b}_i), \tag{9}$$

$$f_t = \sigma(\boldsymbol{W}_f\boldsymbol{x}_t + \boldsymbol{U}_f\boldsymbol{c}_{t-1} + \boldsymbol{V}_f\boldsymbol{h}^c_{t-1} + \boldsymbol{b}_f), \tag{10}$$

$$\boldsymbol{c}_t = f_t * \boldsymbol{c}_{t-1} + i_t * tanh(\boldsymbol{W}_c\boldsymbol{x}_t + \boldsymbol{V}_c\boldsymbol{h}^c_{t-1} + \boldsymbol{b}_c), \tag{11}$$

$$o_t = \sigma(\boldsymbol{W}_o\boldsymbol{x}_t + \boldsymbol{U}_o\boldsymbol{c}_{t-1} + \boldsymbol{V}_o\boldsymbol{h}^c_{t-1} + \boldsymbol{b}_o), \tag{12}$$

$$\boldsymbol{h}^c_t = o_t * tanh(\boldsymbol{c}_t), \tag{13}$$

where $\boldsymbol{W}_i$, $\boldsymbol{W}_f$, $\boldsymbol{W}_o$ and $\boldsymbol{b}_i$, $\boldsymbol{b}_f$, $\boldsymbol{b}_c$, $\boldsymbol{b}_o$ are the trainable weight matrices and biases, respectively. σ is the activation function *sigmoid*. In addition, i_t represents input gate state, f_t forget gate state, $\boldsymbol{c}_t$ cell state, o_t output gate and $\boldsymbol{h}^c_t$ the hidden layer output in the current time-step.

After that, we obtain a hidden vector sequence $\boldsymbol{H} = (\boldsymbol{h}^c_1, \boldsymbol{h}^c_2, ..., \boldsymbol{h}^c_n)$ generated by LSTM. Then, we use the attention mechanism to select important vectors in hidden vector sequences for learning more informative contextual representations. The attention weight α^c_i of the i-th hidden vector in $\boldsymbol{H}$ and the output vector $\boldsymbol{h}^C$ are computed as:

$$a^c_i = \boldsymbol{q}^T \tanh(\boldsymbol{V}_c * \boldsymbol{h}^c_i + \boldsymbol{v}_c), \tag{14}$$

$$\alpha^c_i = \frac{\exp(a^c_i)}{\sum_{j=1}^n \exp(a^c_j)}, \tag{15}$$

$$\boldsymbol{h}^C = \sum_{i=1}^n \alpha^c_i \boldsymbol{h}^c_i, \tag{16}$$

where $\boldsymbol{V}_c$ and $\boldsymbol{v}_c$ are projection parameters, and $\boldsymbol{q}$ is the query vector.

3.5 Adaptively Temporal Feature Fusion

Let $\boldsymbol{h}^C$ and $\boldsymbol{h}^E$ denote the temporal feature vector learned from the Attention-based LSTM, and the content feature vector learned from the Tree Attention Network, respectively. Now, the critical question is how to design an effective and stable feature fusion module. The traditional early fusion schemes (such as concatenation and point-wise addition) lack flexibility and cannot capture deep information. Inspired by [15], we believe that as the age of the video increases, the importance of content features gradually decreases, while the importance of temporal features gradually increases. Based on the above view, we propose an Adaptively Temporal Feature Fusion scheme (ATFF), which dynamically fuses features with the current temporal information (the age of video).

The first part of ATFF is an information valve based on a learnable exponential decay function, which controls the inflow of different information at different times. The formula is expressed as follows:

$$\widetilde{\boldsymbol{h}}^E = \varphi(\Delta t) * \boldsymbol{h}^E, \tag{17}$$

$$\widetilde{\boldsymbol{h}}^C = (1 - \varphi(\Delta t)) * \boldsymbol{h}^C, \tag{18}$$

$$\varphi(\Delta t) = \exp(-\boldsymbol{\theta} * (\boldsymbol{W}_\varphi \Delta t + \boldsymbol{b}_\varphi)), \tag{19}$$

where $\Delta t = t_{predicted} - t_{publish}$, $\varphi(\cdot)$ is a learnable exponential decay function used to simulate the importance of information decay over time, and $\boldsymbol{\theta}$ is a trainable parameter controlling the decay rate of $\varphi(\cdot)$.

After dealing with external influences, we need to consider the internal interactions between different features. The second part of ATFF uses the attention mechanism [26] for internal interaction between temporal feature representation and content feature representation. We first apply a linear transformation $\boldsymbol{W}_g$ to $[\widetilde{\boldsymbol{h}}^E, \widetilde{\boldsymbol{h}}^C]$ to obtain a global vector $\boldsymbol{r}$. Then we calculate the dot product of the global vector and each feature vector to obtain the attention weights. With the attentive weights, we can get the final representation $\boldsymbol{F}$.

$$\boldsymbol{r} = [\widetilde{\boldsymbol{h}}^E, \widetilde{\boldsymbol{h}}^C] * \boldsymbol{W}_g, \tag{20}$$

$$\alpha_i = \frac{\exp(\boldsymbol{r}^T * \widetilde{\boldsymbol{h}}^i)}{\sum_{k \in \{E,C\}} \exp(\boldsymbol{r}^T * \widetilde{\boldsymbol{h}}^k)}, \tag{21}$$

$$\boldsymbol{F} = \sum_{i \in \{E,C\}} \alpha_i * \widetilde{\boldsymbol{h}}^i. \tag{22}$$

After that, we utilize a simple one-layer feed-forward neural network to obtain the final popularity of the video, which is calculated as follows:

$$\hat{y} = \text{ReLU}(\boldsymbol{W}_F \boldsymbol{F} + \boldsymbol{b}_F). \tag{23}$$

3.6 Model Training

We define the loss function for video popularity prediction task as follows:

$$\mathcal{L}_{PP} = \sum_{i \in D} \sum_{t \in \{t_s, t_s+s, t_s+2*s, \ldots, t_e\}} MSE(y_t^i, \hat{y}_t^i), \tag{24}$$

where D is the training video set, t is the prediction time point, t_s is the time point at which the video is first predicted, t_e is the time point at which the video is last predicted, s is the step size of the sliding window (set as 24 in our experiment), $\hat{y}^i$ is the predicted value, and y^i is the target value in the ground-truth.

4 EXPERIMENTS

In this section, we conduct comprehensive experiments on two real-world datasets to answer the following questions:

- **Q1**: How does KTFN perform compared with other models?
- **Q2**: How do different components affect KTFN?
- **Q3**: How does the different information contribute to the prediction performance?
- **Q4**: Can KTFN provide a reasonable explanation for the prediction results?

4.1 Dataset

We collect a medium-video[4] dataset and a micro-video[5] dataset from the server logs of Xigua[6] and Douyin[7], respectively, both of them are online video-sharing platforms owned by ByteDance[8]. Specifically, we randomly sample 72,372 videos published from April 1, 2021, to April 14, 2021, as the test set. We firstly record the authors of the videos in the test set, and then 408,202 videos published by these authors from March 1, 2021, to March 31, 2021, are further selected as our training set. For each video, the hourly view information is available, so we collect these hourly views and form a sequence according to the timestamp order, in which each time point represents the number of views in one hour. For the constructed sequence of each video, we then use the sliding window algorithm to split them into multiple records, where the size of the source window, the size of the target window, and the step size are $n = 24$, $m = 72$, and $s = 24$, respectively. Then, we use the same approach to process the dataset collected from Douyin. Besides the time-series data, we need to construct the knowledge graph for each dataset. In the video knowledge graph, the types of entities include "Video", "Duration", "Keywords", "Category", "Publish hour", "Author", "Fans", and "Level of the author" in the internal system. The basic statistics and distributions of the two datasets and the knowledge graph are shown in Table 1, 2 and Figure 3, 4, 5, respectively.

Table 1: Statistical information for two datasets.

Dataset	# users	Set	# videos	# records	Publish date
Xigua	27, 063	Training	408, 202	5, 409, 563	3.1-3.31
		Test	72, 372	795, 597	4.1-4.14
Douyin	50, 675	Training	383, 452	1, 736, 549	7.1-7.14
		Test	70, 395	298, 294	7.15-7.21

Table 2: Knowledge graph information.

Dataset	# entities	# relations	# triples
Xigua	697, 175	7	4, 236, 962
Douyin	638, 668	7	3, 538, 749

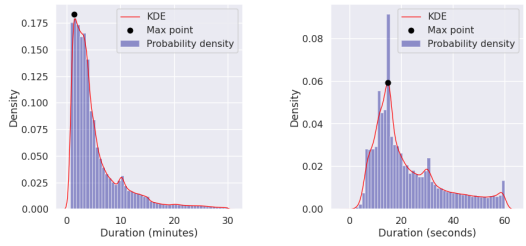
4.2 Experimental Setup

4.2.1 ***Metrics.*** To evaluate the performance of different methods, we adopt three widely-used metrics: *Mean Squared Error (MSE)*, *Mean Absolute Error (MAE)*, and *Accuracy* [4], where *Accuracy* measures the proportion of videos correctly predicted for a given error tolerance ϵ, defined as:

$$ACC = \frac{1}{N} \sum_{i=1}^{N} |\{|\frac{y^i - \hat{y}^i}{y^i}| < \epsilon\}|, \tag{25}$$

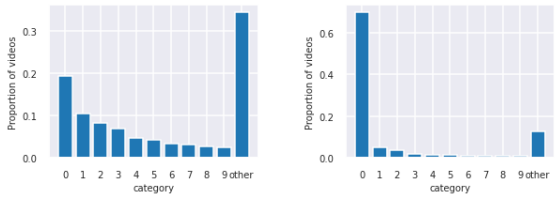
where N is the size of test set, and we set $\epsilon = 0.2$ in this paper.

[4]video duration from 1 minute to 30 minutes
[5]video duration within 1 minute
[6]https://www.ixigua.com/
[7]https://www.douyin.com/recommend
[8]https://www.bytedance.com/



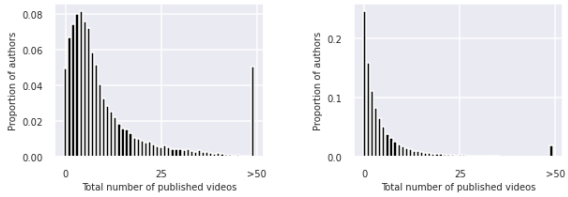
(a) Xigua (b) Douyin

Figure 3: Kernel density estimation plots of video duration.



(a) Xigua (b) Douyin

Figure 4: The distribution of video categories.



(a) Xigua (b) Douyin

Figure 5: The distribution of the total number of videos published by authors

4.2.2 ***Baselines.*** To demonstrate the effectiveness of our model, we choose to compare with the following baselines:

- **MLR** [20]. Multivariate Linear Regression takes the linear combination of multiple variables as predictive values. To make it suitable for our task, we take both time-series and processed content features as inputs.
- **SVR** [11]. Khosla et al. employ a linear kernel Support Vector Regression model that predicts popularity using time-series data as features. We do the same pre-processing as MLR to make the SVR suitable for our task.
- **DA-RNN** [21]. A Dual-stage Attention-based Recurrent Neural Network is a time-series prediction model based on the encoder-decoder model.
- **LSTnet** [14]. A Long-and Short-term Time-series network is proposed for time-series prediction, which uses CNN to model short-term dependencies and exploits a skip-RNN to discover long-term patterns of time-series.
- **KBPPN** [4]. KB-enhanced Popularity Prediction Network introduces knowledge bases into online content popularity prediction and integrates content feature representation and temporal feature representation of online items with a gate mechanism.

Table 3: Comparison of different models.

Dataset	Model	MSE (↓)	MAE (↓)	ACC (↑)
Xigua	MLR [20]	2.323	1.001	0.300
	SVR [11]	2.067	0.896	0.362
	LSTnet [14]	2.024	0.851	0.407
	DA-RNN [21]	1.842	0.812	0.412
	KBPPN [4]	1.129	0.683	0.438
	KTFN	**0.646**	**0.496**	**0.502**
Douyin	MLR [20]	1.778	1.044	0.311
	SVR [11]	1.473	0.908	0.401
	LSTnet [14]	1.321	0.872	0.429
	DA-RNN [21]	1.149	0.762	0.436
	KBPPN [4]	0.861	0.627	0.468
	KTFN	**0.466**	**0.424**	**0.529**

4.2.3 ***Parameter setup.*** Hyper-parameters are updated based on the 20% of the training set. For TransR model, we set the margin $\gamma = 4$, the dimension of entity embedding and relation embedding are both fixed to 128. In the TAN, we set function g as ReLU for aggregators. The dimension of all hidden layers is set to 128. Except that θ is initialized to 1.0, all other parameters are initialized with Xavier [7]. And we optimize the model with Adam optimizer [12]. The batch size is set to 128. To avoid overfitting, we set dropout to 0.2.

4.2.4 ***Time and space complexity analysis.*** Suppose that $|\mathcal{G}|$ is the number of nodes in the knowledge graph, d is the embedding size, and n is the length of the input time-series. The space consumption of our model comes from two parts: the storage of entity and the relation embedding, and the storage of the weight matrix. Therefore, the space complexity of our model is $O(|\mathcal{G}|d + d^2)$.

For a given video i, its sub-graph in the knowledge graph is $\mathcal{G}_i$. $\mathcal{T}_i$ is the tree form of $\mathcal{G}_i$ (cf. Section 3.3.1). Let $|\mathcal{T}_i|$ and $|\mathcal{V}_i|$ represent the number of nodes and edges of $\mathcal{T}_i$, respectively. The time-consumption for predicting the popularity of video i mainly comes from three components. (1) The computational complexity of the Tree Attention Network is $O(|\mathcal{T}_i|d^2 + |\mathcal{V}_i|d)$. (2) The computational complexity of the Attention-based LSTM is $O(nd^2 + nd)$. (3) The ATFF has a linear computational complexity: $O(d)$. Therefore, the overall time complexity of KTFN is $O((|\mathcal{T}_i|+n)d^2+(|\mathcal{V}_i|+n+1)d)$.

The time cost in the inference phase is significant for online video popularity prediction systems. For online prediction, the TAN of KTFN only performs a single computation for each specific video, while ALSTM and ATFF will be executed at each prediction time point. We conduct experiments on GeForce RTX 3090 to see the specific time consumption of KTFN. We find that the time cost of LSTM, ALSTM and KTFN is 45ms, 57ms and 75ms, respectively, in a single-step prediction experiment with the batch size of 128.

4.3 Results

In this part, we firstly report the performance of all methods on two datasets and then investigate the impact of different factors (i.e., the choice of feature fusion schemes, the choice of information aggregators, different information missing, and hyper-parameter settings) on our model.

Table 4: Comparison of different feature fusion methods.

Dataset	Fusion	MSE (↓)	MAE (↓)	ACC (↑)
Xigua	Sum	1.427	0.756	0.400
	Concat	1.424	0.746	0.412
	Attention	0.872	0.572	0.462
	ATFF	**0.646**	**0.496**	**0.502**
Douyin	Sum	0.674	0.566	0.497
	Concat	0.664	0.563	0.496
	Attention	0.549	0.484	0.511
	ATFF	**0.466**	**0.424**	**0.529**

Table 5: Comparison of different aggregators.

Dataset	Aggregator	MSE (↓)	MAE (↓)	ACC (↑)
Xigua	Sum	0.747	0.533	0.482
	Concat	0.688	0.512	0.489
	Bi-interaction	**0.646**	**0.496**	**0.502**
Douyin	Sum	0.473	0.424	0.528
	Concat	0.504	0.428	0.527
	Bi-interaction	**0.466**	**0.424**	**0.529**

4.3.1 ***Comparison of different models (Q1).*** Table 3 shows the comparative results of the different models. By analyzing the results from Table 3, we draw the following conclusions:

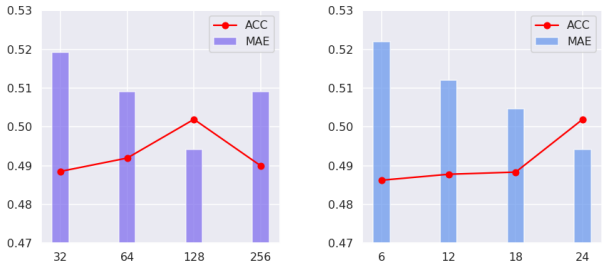
- MLR and SVR perform the worst on both datasets. We believe that the feature-based regression method excessively relies on the manually extracted features, while it is difficult to capture the deep connection between different features.
- LSTnet and DA-RNN have similar resluts. They both model popularity trends with LSTM, while they still lack predictive capacity because they do not consider the content features.
- KBPPN is the strongest among all benchmarks, which utilizes a gate mechanism to fuse the content features learned from the knowledge base with the temporal features learned from the LSTM. However, its fusion scheme ignores the influence of different features at different prediction time points, and it still lacks predictive capability.
- Compared with them, KTFN consistently yields the best performance on both datasets. We believe there are two main factors. (1) TAN can learn the local neighborhood information of entities extremely well. (2) Our proposed ATFF enables features from outside and inside to interact with each other via information valves and the global attention mechanism, making the feature fusion more flexible.

4.3.2 ***Comparison among KTFN variants (Q2).*** Further, we compare the variants of KTFN regarding the following two aspects to demonstrate the effectiveness of the KTFN framework design: the choice of feature fusion schemes and the choice of information aggregators. The results are shown in Table 4 and 5, from which we can draw the following conclusions:

- We find huge differences among the results obtained by different feature fusion schemes. Specifically, concatenation and point-wise addition perform the worst, mainly due to their lack of dynamism. The main reason for the improvement of the attention mechanism [26] over them is that it dynamically attends to the importance of different features. However, the attention mechanism still lacks flexibility since it ignores the amount of information contained in different features at different prediction time points. In particular, compared with the attention mechanism, ATFF improves over the attention fusion method w.r.t. MSE by **25.9%**, and **15.1%** and w.r.t. MAE by **13.2%**, and **12.4%** and w.r.t. ACC by **8.7%**, and **3.5%** in Xigua, Douyin, respectively. This verifies the validity of ATFF and also reveals that ATFF is an inseparable part of KTFN.
- From Table 5 we can observe that the *Bi-interaction* aggregator is superior to the additive and concatenated aggregators. The main why the *Bi-interaction* aggregator performs best lies in that entities can fully interact with their neighbors' information.

Table 6: Impact of different information.

Dataset	Missing	MSE (↓)	MAE (↓)	ACC (↑)
Xigua	No missing	**0.646**	**0.496**	**0.502**
	Duration	0.654	0.499	0.496
	Publish hour	0.668	0.506	0.489
	Keywords	0.663	0.513	0.482
	Category	0.706	0.518	0.485
	Author	1.053	0.685	0.380
Douyin	No missing	**0.466**	**0.424**	**0.529**
	Duration	0.467	0.426	0.528
	Publish hour	0.471	0.428	0.525
	Keywords	0.475	0.434	0.521
	Category	0.513	0.461	0.503
	Author	0.615	0.516	0.447



(a) Impact of hidden dimension (b) Impact of source window size

Figure 6: Parameter test results on Xigua. Performance (MAE, ACC) of KTFN with different hyper-parameters (*n*, *d*).

4.3.3 ***Impact of different information (Q3).*** In this section, we explore the impact of different kinds of information on the prediction results under the scenario of missing certain information in the test phase, including: "Duration", "Publish hour", "Keywords", "Category", and "Author". We conduct experiments with the trained KTFN model, and Table 6 summarizes the experimental results. We have the following observations:

- As expected, the lack of any information decreases the performance of KTFN, which indicates that the information we used is all effective for the prediction task.
- By considering the three metrics (*MSE*, *MAE*, *ACC*) together, We find that "Duration", "Publish hour", "Keywords", "Category", and "Author" have a progressively increasing effect on the performance of KTFN. We believe that most users usually do not consider the duration of a video as a key factor in whether to watch it or not. If a video is published in the middle of the night, it may receive fewer views, while it may receive more views if it is published in the break time, so "Publish hour" can influence the prediction results. "Keywords" and "Category" have a greater impact on the prediction results, the main reason is that people are more eager to watch hot topics.
- Finally, we observe a sharp decrease in KTFN performance by eliminating "Author" information, suggesting that author information is critical in popularity prediction. The popularity of videos is a long-tailed distribution, and videos posted by authors with more fans are more likely to be seen by people, leading to a winner-takes-all situation.

4.3.4 ***Impact of different hyper-parameters.*** In this section, we investigate how the hidden dimension and source window size influence the performance of KTFN on the Xigua dataset. The results are shown in Figure 6. From Figure 6a, we can observe that KTFN performs best when $d = 128$. Increasing d initially improves the performance because a larger d can encode more information, while a too-large d suffers from the detrimental effects of overfitting. From Figure 6b, we find that the performance of KTFN keeps increasing as n increases, because longer time-series contain more information. We can certainly choose a larger n for prediction, but if n is too large, it will cause storage pressure on the online system, and a trade-off between prediction performance and storage needs to be made according to the actual scenario.

4.4 Case Study (Q4)

To demonstrate the interpretability of KTFN, we randomly select two predicted videos from the test set for visualization, as shown in Figure 7. The left half of this sub-figure visualizes the results of TAN, and the right half shows the attention scores of ALSTM. We have the following observations:

- The TAN ensures similar attention scores in the relation-level by stably learning the common information of different entities (i.e. two different video entities in the same relation have very close attention scores). This shows that TAN is very effective in learning important features and filtering out useless entities.
- The attention scores of video entities on the relations "Createdby", "Belongsto", "Contains", "PublishesIn" and "HasDuration" decrease sequentially. This result aligns with our knowledge and the experimental results in Section 4.3.3 that the popularity of a video mainly depends on the author, the category and the content of the video.
- By comparing the video with 28 hours of release to the video with 5 hours of release, we find that ATFF is able to effectively filter and integrate features according to the age of the video.

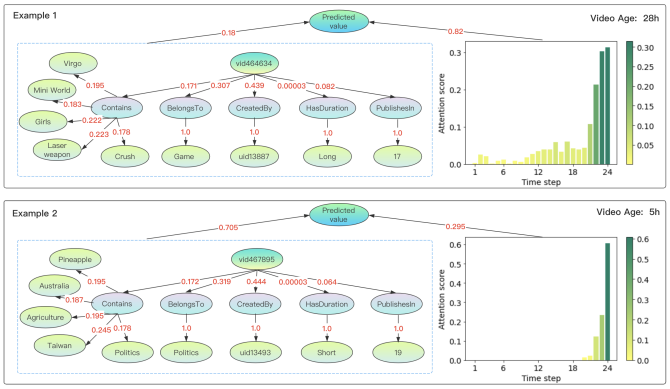


Figure 7: Attention diagram visualization of two examples in test set of Xigua dataset.

We also find that the content features of the video play a dominant role in the early release of the video for popularity prediction, which indicates that a superior feature extractor is crucial for popularity prediction.

5 CONCLUSION

In this paper, we introduce knowledge graphs into online video popularity prediction and propose a Knowledge-based Temporal Fusion Network (KTFN). KTFN consists of three components: Tree Attention Network (TAN) for learning video content feature representation, Attention-based LSTM (ALSTM) for learning video temporal feature representation, and an Adaptively Temporal Feature Fusion (ATFF) module to integrate the above two features dynamically. Specifically, TAN first converts graphs into trees and then propagates and aggregates information by utilizing the graph attention network (GAT) approach. It learns local neighborhood information of entities in a fine-grained manner, which enables our model to highlight the importance of different entities under the same relation and enhances the interpretability of the model. Then, we employ an Attention-based LSTM to learn the temporal feature representations. Finally, we propose an Adaptively Temporal Feature Fusion Scheme (ATFF) to dynamically integrate content features and temporal features. Specifically, ATFF first filters the feature vectors using a learnable exponential decay function and then combines the global attention mechanism for feature fusion. We collect a medium-video dataset and a micro-video dataset from the server logs of Xigua and Douyin, respectively. Extensive experiments on both datasets demonstrate the effectiveness and interpretability of KTFN.

ACKNOWLEDGMENTS

This work is supported by Guangdong Province Key Area R&D Program under grant No. 2018B010113001, National Natural Science Foundation of China under grant No. 61972189 and 61902145, the Shenzhen Key Lab of Software Defned Networking under grant No. ZDSYS20140509172959989, National Key R&D Program of China under grant No. 2021ZD0112501 and 2021ZD0112502, Philosophy and Social Sciences Intelligent Library Foundation of Jilin Province under grant No. 2021JLSKZKZB080.

REFERENCES

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

[2] Biao Chang, Hengshu Zhu, Yong Ge, Enhong Chen, Hui Xiong, and Chang Tan. 2014. Predicting the popularity of online serials with autoregressive models. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 1339–1348.

[3] Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. Cognitive graph for multi-hop reading comprehension at scale. *arXiv preprint arXiv:1905.05460* (2019), 1–10.

[4] Hongjian Dou, Wayne Xin Zhao, Yuanpei Zhao, Daxiang Dong, Ji-Rong Wen, and Edward Y Chang. 2018. Predicting the popularity of online content with knowledge-enhanced neural networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1–8.

[5] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. 2016. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1555–1564.

[6] Hongchang Gao, Deguang Kong, Miao Lu, Xiao Bai, and Jian Yang. 2018. Attention convolutional neural network for advertiser-level click-through rate forecasting. In *Proceedings of the 2018 World Wide Web Conference*. 1855–1864.

[7] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. 249–256.

[8] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* (1997), 1735–1780.

[9] Liangjie Hong, Ovidiu Dan, and Brian D Davison. 2011. Predicting popular messages in twitter. In *Proceedings of the 20th International Conference Companion on World Wide Web*. 57–58.

[10] Valerie Isham and Mark Westcott. 1979. A self-correcting point process. *Stochastic processes and their applications* (1979), 335–347.

[11] Aditya Khosla, Atish Das Sarma, and Raffay Hamid. 2014. What makes an image popular?. In *Proceedings of the 23rd international conference on World Wide Web*. 867–876.

[12] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014), 1–15.

[13] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016), 1–14.

[14] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 95–104.

[15] Dongliang Liao, Jin Xu, Gongfu Li, Weijie Huang, Weiqing Liu, and Jing Li. 2019. Popularity prediction on online articles with deep fusion of temporal process and content features. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 200–207.

[16] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*. 2181–2187.

[17] Robert L Logan IV, Nelson F Liu, Matthew E Peters, Matt Gardner, and Sameer Singh. 2019. Barack's wife hillary: Using knowledge-graphs for fact-aware language modeling. *arXiv preprint arXiv:1906.07241* (2019), 1–10.

[18] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th International Conference on Machine Learning*. 1–6.

[19] Swapnil Mishra, Marian-Andrei Rizoiu, and Lexing Xie. 2018. Modeling popularity in asynchronous social media streams with recurrent neural networks. In *Proceedings of the 2018 International AAAI Conference on Web and Social Media*. 201–210.

[20] Henrique Pinto, Jussara M Almeida, and Marcos A Gonçalves. 2013. Using early view patterns to predict the popularity of youtube videos. In *Proceedings of the 6th ACM International Conference on Web search and data mining*. 365–374.

[21] 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. *arXiv preprint arXiv:1704.02971* (2017), 1–7.

[22] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. 2018. Deepinf: Social influence prediction with deep learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2110–2119.

[23] Marian-Andrei Rizoiu, Lexing Xie, Scott Sanner, Manuel Cebrian, Honglin Yu, and Pascal Van Hentenryck. 2017. Expecting to be HIP: Hawkes intensity processes for social media popularity. In *Proceedings of the 26th International Conference on World Wide Web*. 735–744.

[24] Suman Deb Roy, Tao Mei, Wenjun Zeng, and Shipeng Li. 2013. Towards cross-domain learning for social video popularity prediction. *IEEE Transactions on multimedia* (2013), 1255–1267.

[25] Linpeng Tang, Qi Huang, Amit Puntambekar, Ymir Vigfusson, Wyatt Lloyd, and Kai Li. 2017. Popularity prediction of facebook videos for higher quality streaming. In *Proceedings of the 2017 USENIX Conference on Usenix Annual Technical Conference*. 111–123.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 2017 Advances in Neural Information Processing Systems*. 5998–6008.

[27] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017), 1–12.

[28] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 950–958.

[29] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 606–615.

[30] Bo Wu, Tao Mei, Wen-Huang Cheng, and Yongdong Zhang. 2016. Unfolding temporal dynamics: Predicting social media popularity using multi-scale temporal decomposition. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. 272–278.

[31] Wei Zhang, Wen Wang, Jun Wang, and Hongyuan Zha. 2018. User-guided hierarchical attention network for multi-modal social image popularity prediction. In *Proceedings of the 2018 World Wide Web Conference*. 1277–1286.

[32] Qingyuan Zhao, Murat A Erdogdu, Hera Y He, Anand Rajaraman, and Jure Leskovec. 2015. Seismic: A self-exciting point process model for predicting tweet popularity. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1513–1522.

[33] Fan Zhou, Xovee Xu, Goce Trajcevski, and Kunpeng Zhang. 2021. A Survey of Information Cascade Analysis: Models, Predictions, and Recent Advances. *ACM Computing Surveys (CSUR)* (2021), 1–36.