Chapter 8
Heterogeneous Graph Representation for Text Mining

Abstract Heterogeneous graph representation techniques can be applied in many real-world applications. Even the natural languages that are usually modeled as sequence data can also be constructed as a heterogeneous graph by some techniques, so as to widely and accurately capture the complex interactions among the words, entities, topics, instances and other components of the texts. In this chapter, we focus on summarizing the heterogeneous graph representation applications on text mining. Particularly, we introduce several heterogeneous graph-based text mining methods, including HGAT for short text classification, GUND and GNewsRec for news recommendation. In the field of heterogeneous graph representation for text mining, methods mainly contain two key components: heterogeneous graph construction from texts and heterogeneous graph representation algorithm for tasks. We will roughly illustrate heterogeneous graph modeling for text mining tasks from these two points.

8.1 Introduction

With the rapid development of online social media and e-commerce, the text corpus on the Internet has grown tremendously, including short texts such as queries, reviews, tweets, etc. [30], and long texts such as news, articles, papers, etc. Therefore, there is a pressing need to successfully and accurately analyze them. For example, as the most basic task, text classification could categorize these text corpus into several groups, thus facilitating storage and rapid retrieval [1, 22]. News recommendation could keep users from information overloading and help them quickly find their interests [5, 43, 35].

However, many of the text analysis tasks will face the problem of data sparsity [26, 41]. Fortunately, graphs, especially heterogeneous graphs, have powerful capabilities in integrating extra information and modeling interactions among objects. Hence researchers explore to construct a suitable heterogeneous graph for these texts, containing different types of objects (e.g., words, entities, topics, instances and

other components of the texts) and one/multiple types of edges connecting the objects together, which could be beneficial for overcoming data sparsity problem and improve many natural language processing tasks. Moreover, different tasks will encounter some of their own unique challenges, which can also be dealt with by the properly constructed heterogeneous graphs followed by a well-designed heterogeneous graph representation method.

In this chapter, we focus on the methods on the two aforementioned typical tasks: short text classification and news recommendation. Specifically, in the task of short text classification, in addition to the challenge of data sparsity, there are also problems such as ambiguity and lack of labeled data. To address these issues, we introduce a novel **H**eterogeneous **G**raph **AT**tention network (named **HGAT**) [18] for semi-supervised short text classification in Section 8.2. In terms of the long texts, in the task of news recommendation, to tackle the problem that existing methods ignore considering the latent topic information and users' long-term and short-term interests, we introduce a novel **G**raph neural **News Rec**ommendation model (named **GNewsRec**) [11] in Section 8.3. Moreover, we introduce another news recommendation method, **G**raph neural **N**ews recommendation model with **U**nsupervised preference **D**isentanglement (named **GNUD**) [12], which further considers users' great diversity of preferences.

8.2 Short Text Classification

8.2.1 Overview

Short text classification can be widely applied in many domains, ranging from sentiment analysis to news tagging/categorization and query intent classification [1, 22]. Nevertheless, short text classification is nontrivial due to the following challenges. Firstly, short texts are usually semantically sparse and ambiguous, lacking contexts [26]. Although some methods have been proposed to incorporate additional information such as entities [39, 37], they are unable to consider the relational data such as the semantic relations among entities. Secondly, the labeled training data is limited, which leads to traditional and neural supervised methods [38, 15, 50, 28, 29] ineffective. As such, how to make full use of the limited labeled data and large number of unlabeled data has become a key problem for short text classification [1]. Finally, it needs to capture the importance of different information that is incorporated to address sparsity at multiple granularity levels and reduce the weights of noisy information to achieve more accurate classification results.

In this section, we introduce a novel heterogeneous graph neural network based method for semi-supervised short text classification [18], which makes full use of both limited labeled data and large unlabeled data by allowing information propagation through the automatically constructed graph. Particularly, it first presents a flexible heterogeneous graph framework for modeling the short texts, which is

able to incorporate any additional information (e.g., entities and topics) as well as capture the rich relations among the texts and the additional information. Then, it proposes **H**eterogeneous **G**raph **AT**tention networks (**HGAT**) to embed the HG for short text classification based on a new dual-level attention mechanism including node-level and type-level attentions. The HGAT method considers the heterogeneity of different node types. Additionally, the dual-level attention mechanism captures both the importance of different neighboring nodes (reducing the weights of noisy information) and the importance of different node (information) types to a current node. Finally, extensive experimental results demonstrate that the proposed HGAT model significantly outperforms seven state-of-the-art methods across six benchmark datasets.

8.2.2 HG Modeling for Short Texts

We first present the HG framework for modeling the short texts, which enables integration of any additional information and captures the rich relations among the texts and the added information. In this way, the sparsity of the short texts is alleviated.

Previous studies have exploited latent topics [49] and external knowledge (e.g., entities) from knowledge bases to enrich the semantics of the short texts [39, 37]. However, they fail to consider the semantic relation information, such as entity relations. This HG framework for short texts is flexible for integrating any additional information and modeling their rich relations.

Here, two types of additional information are considered, i.e., topics and entities. As shown in Fig. 8.1, the HG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is constructed containing the short texts $D = \{d_1, ..., d_m\}$, topics $T = \{t_1, ..., t_K\}$, and entities $E = \{e_1, ..., e_n\}$ as nodes, i.e., $\mathcal{V} = D \cup T \cup E$. The set of edges $\mathcal{E}$ represent their relations. The details of constructing the network are described as follows.

First, the latent topics T are mined to enrich the semantics of short texts using LDA [3]. Each topic $t_i = (\theta_1, ..., \theta_w)$ (w denotes the vocabulary size) is represented by a probability distribution over the words. Each document is assigned to the top P topics with the largest probabilities. Thus, the edge between a document and a topic is built if the document is assigned to the topic.

Second, recognize the entities E in the documents D and map them to Wikipedia with the entity linking tool TAGME[1]. The edge between a document and an entity is built if the document contains the entity. An entity is taken as a whole word and learn the entity embeddings using word2vec[2] based on the Wikipedia corpus. To further enrich the semantics of short texts and advance the information propagation, HGAT considers the relations between entities. Particularly, if the similarity score (cosine

[1] https://sobigdata.d4science.org/group/tagme/

[2] https://code.google.com/archive/p/word2vec/

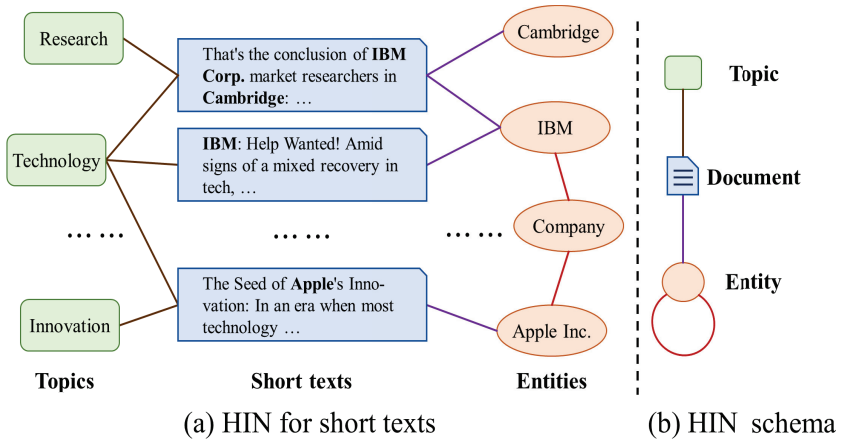


Fig. 8.1 An example of HG for short texts on AGNews. (Note that HIN, Heterogeneous Information Network, in above figure can be seen as an alias of heterogeneous graph. Here we use the original figure of [18].)

similarity) between two entities, computed based on their embeddings, is above a predefined threshold δ, an edge is built between them.

By incorporating the topics, entities and the relations, the semantics of the short texts are enriched and thus greatly benefit the following classification task. For example, as shown in Fig. 8.1, the short text "the seed of Apple's Innovation: In an era when most technology..." is semantically enriched by the relations with the entities "Apple Inc." and "company", as well as the topic "technology". Thus, it can be correctly classified into the category of "business" with high confidence.

8.2.3 The HGAT Model

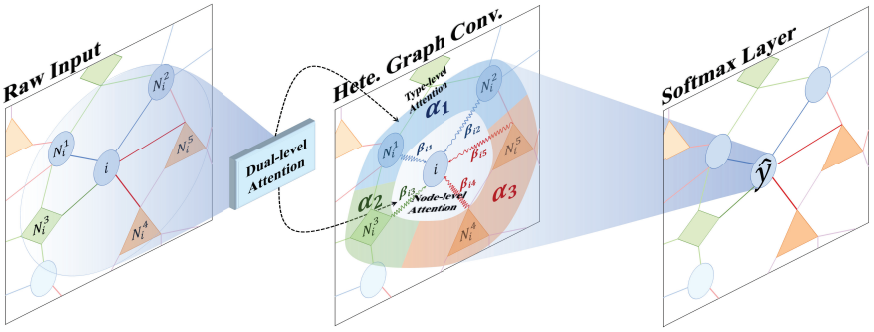


Fig. 8.2 Illustration of the model HGAT.

We then introduce the HGAT model (shown in Fig. 8.2) to embed the HG for short text classification based on a new dual-level attention mechanism including node level and type level. HGAT considers the heterogeneity of different types of information with heterogeneous graph convolution. In addition, the dual-level attention mechanism captures the importance of different neighboring nodes (reducing the weights of noisy information) and the importance of different node (information) types to a specific node. Finally, it predicts the labels of documents through a softmax layer.

We first describe the heterogeneous graph convolution in HGAT, considering the heterogeneous types of nodes (information).

As known, GCN [16] is a multi-layer neural network that operates directly on a homogeneous graph and induces the embedding vectors of nodes based on the properties of their neighborhoods. Formally, consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ and $\mathcal{E}$ represent the set of nodes and edges respectively. Let $X \in \mathbb{R}^{|\mathcal{V}| \times q}$ be a matrix containing the nodes with their features $x_v \in \mathbb{R}^q$ (each row x_v is a feature vector for a node v). For the graph $\mathcal{G}$, we introduce its adjacency matrix $A' = A + I$ with added self-connections and degree matrix M, where $M_{ii} = \sum_j A'_{ij}$. Then the layer-wise propagation rule is as follows:

$$H^{(l+1)} = \sigma(\tilde{A} \cdot H^{(l)} \cdot W^{(l)}). \tag{8.1}$$

Here, $\tilde{A} = M^{-\frac{1}{2}} A' M^{-\frac{1}{2}}$ represents the symmetric normalized adjacency matrix. $W^{(l)}$ is a layer-specific trainable transformation matrix. $\sigma(\cdot)$ denotes an activation function such as ReLU. $H^{(l)} \in \mathbb{R}^{|\mathcal{V}| \times q}$ denotes the hidden representations of nodes in the l^{th} layer. Initially, $H^{(0)} = X$.

Unfortunately, GCN cannot be directly applied to the HG for short texts due to the node heterogeneity issue. Specifically, in the HG, there are three types of nodes: documents, topics and entities with different feature spaces. For a document $d \in D$, the TF-IDF vector is used as its feature vector x_d. For the topic $t \in T$, the word distribution is used to represent the topic $x_t = \{\theta_i\}_{i=[1,w]}$. For each entity, to make full use of relevant information, the entity x_v is represented by concatenating its embedding and TF-IDF vector of its Wikipedia description text.

A straightforward way to adapt GCN for the HG containing different types of nodes $\mathcal{T} = \{\tau_1, \tau_2, \tau_3\}$ is to construct a new large feature space by concatenating together the feature spaces of different types of nodes. For example, each node is denoted as a feature vector with 0 values for the irrelevant dimensions for other types. We name this basic method for adapting GCN to HG as *GCN-HIN*[3]. However, the performance of GCN-HIN is limited since it ignores the heterogeneity of different information types.

To address the issue, the heterogeneous graph convolution is proposed to consider the difference of various types of information and project them into an implicit common space with their respective transformation matrices.

[3] Here we follow the original naming of [18].

$$H^{(l+1)} = \sigma(\sum_{\tau \in \mathcal{T}} \tilde{A}_\tau \cdot H_\tau^{(l)} \cdot W_\tau^{(l)}), \tag{8.2}$$

where $\tilde{A}_\tau \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}_\tau|}$ is the submatrix of $\tilde{A}$, whose rows represent all the nodes and columns represent their neighboring nodes with the type τ. The representation of the nodes $H^{(l+1)}$ is obtained by aggregating information from the features of their neighboring nodes $H_\tau^{(l)}$ with different types τ using different transformation matrix $W_\tau^{(l)} \in \mathbb{R}^{q^{(l)} \times q^{(l+1)}}$. The transformation matrix $W_\tau^{(l)}$ considers the heterogeneity of different feature spaces and projects them into an implicit common space $\mathbb{R}^{q^{(l+1)}}$. Initially, $H_\tau^{(0)} = X_\tau$.

Then, we present the dual-level attention mechanism. Typically, given a specific node, different types of neighboring nodes may have different impacts on it. For example, the neighboring nodes of the same type may carry more useful information. Additionally, different neighboring nodes of the same type could also have different importance. To capture both the different importance at both node level and type level, a new dual-level attention mechanism is designed as follows.

Type-level Attention Given a specific node v, the type-level attention learns the weights of different types of neighboring nodes. Specifically, first represent the embedding of the type τ as $h_\tau = \sum_{v'} \tilde{A}_{vv'} h_{v'}$, which is the sum of the neighboring node features $h_{v'}$ where the nodes $v' \in \mathcal{N}_v$ and are with the type τ. Then, calculate the type-level attention scores based on the current node embedding h_v and the type embedding h_τ:

$$a_\tau = \sigma(\mu_\tau^T \cdot [h_v || h_\tau]), \tag{8.3}$$

where μ_τ is the attention vector for the type τ, $||$ means "concatenate", and $\sigma(\cdot)$ denotes the activation function, such as Leaky ReLU.

Then the type-level attention weights can be obtained by normalizing the attention scores across all the types with the softmax function:

$$\alpha_\tau = \frac{\exp(a_\tau)}{\sum_{\tau' \in \mathcal{T}} \exp(a_{\tau'})}. \tag{8.4}$$

Node-level Attention The node-level attention is designed to capture the importance of different neighboring nodes and reduce the weights of noisy nodes. Formally, given a specific node v with the type τ and its neighboring node $v' \in \mathcal{N}_v$ with the type τ', compute the node-level attention scores based on the node embeddings h_v and $h_{v'}$ with the type-level attention weight $\alpha_{\tau'}$ for the node v':

$$b_{vv'} = \sigma(\nu^T \cdot \alpha_{\tau'} [h_v || h_{v'}]), \tag{8.5}$$

where ν is the attention vector. Then normalize the node-level attention scores with the softmax function:

$$\beta_{vv'} = \frac{\exp(b_{vv'})}{\sum_{i \in \mathcal{N}_v} \exp(b_{vi})}. \tag{8.6}$$

Finally, the dual-level attention mechanism including type-level and node-level attentions is incorporated into the heterogeneous graph convolution by replacing Eq. 8.2 with the following layer-wise propagation rule:

$$H^{(l+1)} = \sigma(\sum_{\tau \in \mathcal{T}} \mathcal{B}_\tau \cdot H_\tau^{(l)} \cdot W_\tau^{(l)}). \tag{8.7}$$

Here, $\mathcal{B}_\tau$ represents the attention matrix, whose element in the v^{th} row v'^{th} column is $\beta_{vv'}$ in Eq. 8.6.

After going through an L-layer HGAT, the embeddings of nodes (including short texts) in the HG can be obtained. The short text embeddings $H^{(L)}$ are then fed to a softmax layer for classification. Formlly,

$$Z = \text{softmax}(H^{(L)}). \tag{8.8}$$

During model training, the cross-entropy loss is exploited over training data with the L2-norm. Formally,

$$\mathcal{L} = -\sum_{i \in D_{\text{train}}} \sum_{j=1}^{C} Y_{ij} \cdot \log Z_{ij} + \eta \|\mathcal{T}heta\|_2, \tag{8.9}$$

where C is the number of classes, D_{train} is the set of short text indices for training, Y is the corresponding label indicator matrix, Θ is model parameters, and η is regularization factor. For model optimization, HGAT adopt the gradient descent algorithm.

8.2.4 Experiments

8.2.4.1 Experimental Settings

Datasets Extensive experiments are conducted on 6 benchmark short text datasets: **AGNews** [50], we randomly select 6,000 pieces of news from AGNews, evenly distributed into 4 classes; **Snippets** [26], it is composed of the snippets returned by a web-search engine; **Ohsumed**[4] [48], it is a benchmark bibliographic classification dataset where the documents with multiple labels are removed. Here use the titles for short text classification; **TagMyNews** [33], it contains English news from really simple syndication (RSS) feeds. Here use the news titles as instances; **MR** [25], it is a movie review dataset, in which each review only contains one sentence annotated with positive or negative for binary sentiment classification; **Twitter**, provided by a library of Python, NLTK[5], it is also a binary sentiment classification dataset. For each dataset, we randomly select 40 labeled documents per class, half of which for

[4] http://disi.unitn.it/moschitti/corpora.htm

[5] https://www.nltk.org/

training and the other half for validation. Following [16], all the left documents are for testing, which are also used as unlabeled documents during training. All the datasets are preprocessed as follows. We remove non-English characters, the stop words, and low-frequency words appearing less than 5 times. Table 8.1 shows the statistics of the datasets, including the number of documents, the number of average tokens and entities, the number of classes, and the proportion of texts containing entities in parentheses. In these datasets, most of the texts (around 80%) contain entities.

Table 8.1 Statistics of the datasets.

	#docs	#tokens	#entities	#classes
AGNews	6,000	18.4	0.9 (72%)	4
Snippets	12,340	14.5	4.4 (94%)	8
Ohsumed	7,400	6.8	3.1 (96%)	23
TagMyNews	32,549	5.1	1.9 (86%)	7
MR	10,662	7.6	1.8 (76%)	2
Twitter	10,000	3.5	1.1 (63%)	2

Baselines To comprehensively evaluate the proposed method for semi-supervised short text classification, we compare it with the following nine state-of-the-art methods: **SVM**: SVM classifiers using TF-IDF features and LDA features [3], are denoted as SVM+TFIDF and SVM+LDA, respectively. **CNN**: CNN [15] with 2 variants: 1) CNN-rand, whose word embeddings are randomly initialized, and 2) CNN-pretrain, whose word embeddings are pre-trained with Wikipedia Corpus. **LSTM**: LSTM [20] with and without pre-trained word embeddings, named LSTM-rand and LSTM-pretrain, respectively. **PTE**: A semi-supervised representation learning method for text data [31]. It firstly learns word embedding based on the heterogeneous text networks containing three bipartite networks of words, documents and labels, then averages word embeddings as document embeddings for text classification. **TextGCN**: Text GCN [48] models the text corpus as a graph containing documents and words as nodes, and applies GCN for text classification. **HAN**: HAN [42] embeds HGs by first converting an HG to several homogeneous sub-networks through pre-defined meta-paths and then applying graph attention networks. For fair comparison, all of the above baselines, such as SVMs, CNN and LSTM, have used entity information.

Parameter Settings The parameter values of K, T and δ are chosen according to the best results on the validation set. To construct HG for short texts, set the number of topics $K = 15$ in LDA for the datasets AGNews, TagMyNews, MR and Twitter, and $K = 20$ for Snippets and $K = 40$ for Ohsumed. For all the datasets, each document is assigned to top $P = 2$ topics with the largest probabilities. The similarity threshold δ between entities is set $\delta = 0.5$. Following previous studies [32], we set the hidden dimension of the model HGAT and other neural models to $d = 512$ and the dimension of pre-trained word embeddings to 100, and the layer number L of HGAT, GCN-HIN

and TextGCN as 2. For model training, the learning rate is set 0.005, dropout rate 0.8 and the regularization factor $\eta = 5e\text{-}6$. Early stopping is applied to avoid overfitting.

8.2.4.2 Main Results

Table 8.2 Test accuracy (%) of different models on six standard datasets. The second best results are underlined. The note * means the proposed model significantly outperforms the baselines based on t-test ($p < 0.01$).

Dataset	SVM +TFIDF	SVM +LDA	CNN -rand	CNN -pretrain	LSTM -rand	LSTM -pretrain	PTE	TextGCN	HAN	HGAT
AGNews	57.73	65.16	32.65	67.24	31.24	66.28	36.00	<u>67.61</u>	62.64	**72.10***
Snippets	63.85	63.91	48.34	77.09	26.38	75.89	63.10	<u>77.82</u>	58.38	**82.36***
Ohsumed	41.47	31.26	35.25	32.92	19.87	28.70	36.63	<u>41.56</u>	36.97	**42.68***
TagMyNews	42.90	21.88	28.76	57.12	25.52	<u>57.32</u>	40.32	54.28	42.18	**61.72***
MR	56.67	54.69	54.85	58.32	52.62	<u>60.89</u>	54.74	59.12	57.11	**62.75***
Twitter	54.39	50.42	52.58	56.34	54.80	<u>60.28</u>	54.24	60.15	53.75	**63.21***

Table 8.2 shows the classification accuracy of different methods on 6 benchmark datasets. One can see that HGAT method significantly outperforms all the baselines by a large margin, which shows the effectiveness of the proposed method on semi-supervised short text classification.

The traditional method SVMs based on the human-designed features, achieve better performance than the deep models with random initialization, i.e., CNN-rand and LSTM-rand in most cases. While CNN-pretrain and LSTM-pretrain using the pre-trained vectors achieve significant improvements and outperform SVMs. The graph based model PTE achieves inferior performance compared to CNN-pretrain and LSTM-pretrain. The reason may be that PTE learns text embeddings based on word co-occurrences, which, however, are sparse in short text classification. Graph neural network based models TextGCN and HAN achieve comparable results with the deep models CNN-pretrain and LSTM-pretrain. The proposed model HGAT consistently outperforms all the state-of-the-art models by a large margin, which shows the effectiveness of the proposed method. The reasons include that 1) HGAT constructs a flexible HG framework for modeling the short texts, enabling integration of additional information to enrich the semantics and 2) we propose a novel model HGAT to embed the HG for short text classification based on a new dual-level attention mechanism. The attention mechanism not only captures the importance of different neighboring nodes (reducing the weights of noisy information) but also the importance of different types of nodes.

Table 8.3 Test accuracy (%) of HGAT variants.

Dataset	GCN -HIN	HGAT w/o ATT	HGAT -Type	HGAT -Node	HGAT
AGNews	70.87	70.97	71.54	71.76	**72.10***
Snippets	76.69	80.42	81.68	81.93	**82.36***
Ohsumed	40.25	41.31	41.95	42.17	**42.68***
TagMyNews	56.33	59.41	60.78	61.29	**61.72***
MR	60.81	62.13	62.27	62.31	**62.75***
Twitter	61.59	62.35	62.95	62.45	**63.21***

8.2.4.3 Comparison of Variants of HGAT

We also compare the model HGAT with four variants to validate the effectiveness of the HGAT model. As shown in Table 8.3, the basic model GCN-HIN directly applies GCN on the constructed HG for short texts by concatenating the feature spaces of different types of information. It does not consider the heterogeneity of various information types. HGAT w/o ATT considers the heterogeneity through the proposed heterogeneous graph convolution, which projects different types of information to an implicit common space with respective transformation matrices. HGAT-Type and HGAT-Node respectively consider only type-level attention and node-level attention.

One can see from Table 8.2, HGAT w/o ATT consistently outperforms GCN-HIN on all datasets, demonstrating the effectiveness of the proposed heterogeneous graph convolution which considers the heterogeneity of various information types. HGAT-Type and HGAT-Node further improve HGAT w/o ATT by capturing the importance of different information (reducing the weights of noisy information). HGAT-Node achieves better performance than HGAT-Type, indicating that node-level attention is more important. Finally, HGAT significantly outperforms all the variants by considering the heterogeneity and applying dual-level attention mechanism including node-level and type-level attentions.

8.2.4.4 Case Study

As Fig. 8.3 shows, a short text from AGNews is taken as an example (which is classified to the class of sports correctly) to illustrate the dual-level attention of HGAT. The type-level attention assigns high weight (0.7) to the short text itself, while lower weights (0.2 and 0.1) to entities and topics. It means that the text itself contributes more for classification, than the entities and topics. The node-level attention assigns different weights to neighboring nodes. The node-level weights of nodes belonging to a same type sum to 1. As one can see, the entities e_3 (Atlanta Braves, a baseball team), e_4 (Dodger Stadium, a baseball gym), e_1 (Shawn Green, a baseball player) have higher weights than e_2 (Los Angeles, referring to a city at most time). The topics t_1 (game) and t_2 (win) have almost the same importance for

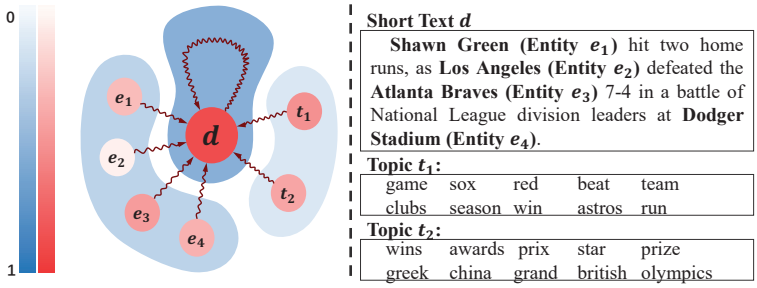


Fig. 8.3 Visualization of the dual-level attention including node-level attention (shown in red) and type-level attention (shown in blue). Each topic t is represented by top 10 words with highest probabilities.

classifying the text to the class of sports. The case study shows that the proposed dual-level attention can capture key information at multiple granularities for classification and reduce the weights of noisy information.

The more detailed method description and experiment validation can be seen in [18] and [47].

8.3 News Recommendation with Long/Short-term Interest Modeling

8.3.1 Overview

News recommender systems that automatically recommend a small set of news articles for satisfying user's preferences, have growingly attracted attentions in both industry and academic [5, 43, 35]. However, most existing methods [5, 34, 14, 35, 51, 13, 6, 17] suffer from the data sparsity problem, since they fail to extensively exploit high-order structure information (e.g., the $u_1 - d_1 - u_2$ relationship indicates the behavior similarity between the users u_1 and u_2). In addition, most of them ignore the latent topic information which would help indicate a user's interest and alleviate the sparse user-item interactions. The intuition is that news items with few user clicks can aggregate more information with the bridge of topics. What's more, the existing methods on news recommendation do not consider the user's long-term and short-term interests. A user usually has relatively stable long-term interests and may also be temporally attracted to certain things, i.e., short-term interests, which should be considered in news recommendation. For example, a user may continuously concern about political events, which is a long-term interest. In contrast, certain breaking news events such as attacks usually attract temporary interests.

To address the above issues, a novel **G**raph Neural **News Rec**ommendation model (**GNewsRec**) is proposed with long-term and short-term user interest modeling.

First, a heterogeneous user-news-topic graph is constructed to explicitly model the interactions among users, news and topics with complete historic user clicks. The topic information can help better reflect a user's interest and alleviate the sparsity issue of user-item interactions. To encode the high-order relationships among users, news items and topics, we take advantage of graph neural networks (GNN) to learn user and news representations by propagating embeddings over the graph. The learned user embeddings with complete historic user clicks are supposed to encode a user's long-term interest. GNewsRec also models a user's short-term interest using recent user reading history with an attention based LSTM [10, 19] model. It combines both long-term and short-term interests for user modeling, which are then compared to the candidate news representation for prediction.

8.3.2 Problem Formulation

The news recommendation problem in this work can be illustrated as follows. Given the click histories for K users $U = \{u_1, u_2, \cdots, u_K\}$ over M news items $I = \{d_1, d_2, \cdots, d_M\}$, the user-item interaction matrix $Y \in \mathbb{R}^{K \times M}$ is defined according to users' implicit feedback, where $y_{u,d} = 1$ indicates the user u clicked the news d, otherwise $y_{u,d} = 0$. Additionally, from the click history with timestamps, the recent click sequence $s_u = \{d_{u,1}, d_{u,2}, \cdots, d_{u,n}\}$ is obtained for a specific user u, where $d_{u,j} \in I$ is the j-th news the user u clicked.

Given the user-item interaction matrix Y as well as the users' recent click sequences S, the news recommendation problem aims to predict whether a user u has potential interest in a news item d which he/she has not seen before. This work considers the title and profile (a given set of entities E and their entity types C from the news page content) of news as features. Each news title T contains a sequence of words $T = \{w_1, w_2, \cdots, w_m\}$. The profile contains a sequence of entities $E = \{e_1, e_2, \cdots, e_n\}$ as well as its type set $C = \{c_1, c_2, \cdots, c_n\}$, where c_j is the type of the j-th entity e_j.

8.3.3 GNewsRec Method

In this subsection, we present our graph neural news recommendation model GNewsRec with long-term and short-term interest modeling. Our model takes full advantage of the high-order structure information between users and news items by first constructing a heterogeneous graph modeling the interactions and then applying GNN to propagate the embeddings. As illustrated in Fig. 8.4, GNewsRec contains three main parts: CNN for text information extraction, GNN for long-term user interest modeling and news modeling, and attention based LSTM model for short-term user interest modeling. The first part extracts the news feature from the news title and profile through CNN. The second part constructs a heterogeneous user-news-topic

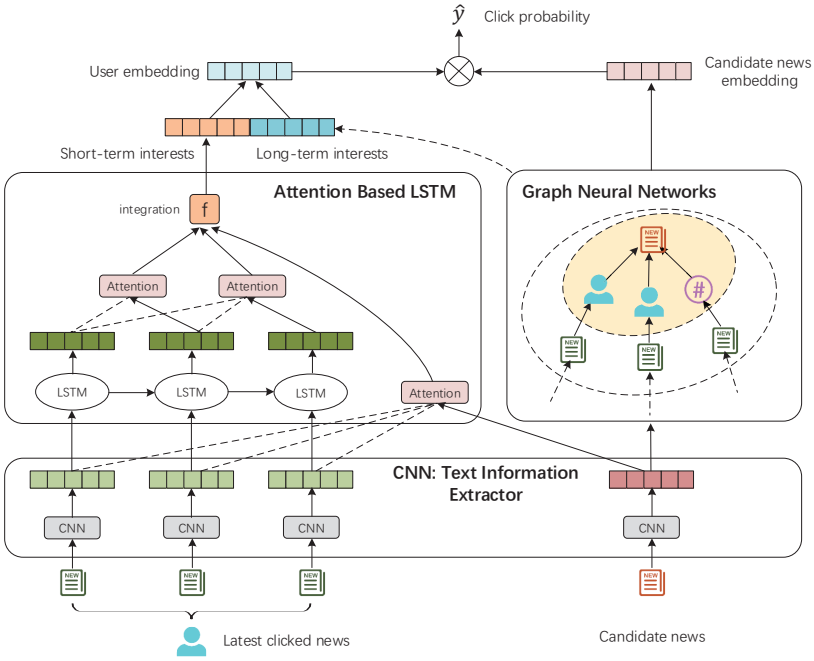


Fig. 8.4 The framework of GNewsRec.

graph with complete historic user clicks and applies GNN to encode high-order structure information for recommendation. The incorporated latent topic information can alleviate the user-item sparsity since news items with few user clicks can aggregate more information with the bridge of topics. The learned user embeddings with complete historic user clicks are supposed to encode the relatively stable long-term user interest. GNewsRec also models the user's short-term interest with recent reading history through an attention based LSTM in the third part. Finally, it combines a user's long-term and short-term interests for user representation, then compares and matches it to candidate news representation for recommendation. The three parts are detailed as follows.

8.3.3.1 Text Information Extractor

GNewsRec uses two parallel CNNs as the news text information extractor, which respectively take the title and profile of news as inputs and learn the title-level and profile-level representations of news. The concatenation of such two representations is regarded as the final text feature representation of news.

Specifically, the title is represented as $T = [w_1, \cdots, w_m]^T$ and the profile as $P = [e_1, f(c_1), e_2, f(c_2), \cdots, e_n, f(c_n)]^T$, where $P \in \mathbb{R}^{2n \times k_1}$ and k_1 is the dimension

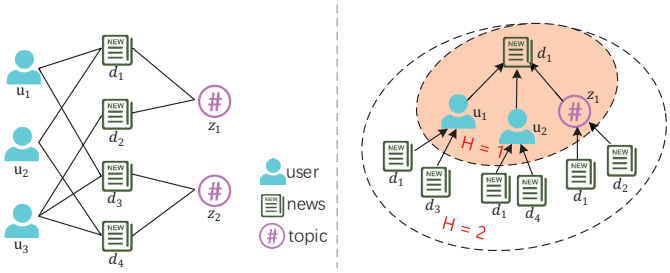


Fig. 8.5 Heterogeneous user-news-topic graph(left) and two-layers GNN(right).

of entity embedding. $f(c) = W_c\ c$ is the transformation function. $W_c \in \mathbb{R}^{k_1 \times k_2}$ (k_2 is the dimension of entity type embedding) is the trainable transformation matrix.

The title T and profile P are respectively fed into two parallel CNNs that have separate weight parameters. Hence their feature representations are separately obtained as $\widetilde{T}$ and $\widetilde{P}$ through two parallel CNNs. Finally $\widetilde{T}$ and $\widetilde{P}$ are concatenated as the final news text feature representation:

$$d = f_c([\widetilde{T}; \widetilde{P}]), \tag{8.10}$$

where $d \in \mathbb{R}^D$ and f_c is a densely connected layer.

8.3.3.2 Long-term User Interest Modeling and News Modeling

To model long-term user interest and news, GNewsRec first constructs a heterogeneous user-news-topic graph with users' complete historic clicks. The incorporated topic information can help better indicate a user's interest and alleviate the sparsity of user-item interactions. Then it applies graph convolutional networks for learning embeddings of users and news items, which encodes the high-order information between users and items through propagating embeddings over the graph.

Heterogeneous User-News-Topic Graph GNewsRec incorporates the latent topic information in news articles to better indicate the user's interest and alleviate the user-item sparsity issue. Hence, a heterogeneous undirected graph $G = (V, R)$ is constructed as illustrated in the left part of Fig. 8.5, where V and R are respectively the sets of nodes and edges. Our graph contains three types of nodes: users U, news items I and topics Z. The topics Z can be mined through the topic model LDA [3].

The user-item edges are built if the user u clicked a news item d, i.e., $y_{u,d} = 1$. For each news document d, its topic distribution is obtained as $\theta_d = \{\theta_{d,i}\}_{i=1,\cdots,\mathcal{K}}$, $\sum_{i=1}^{\mathcal{K}} \theta_i = 1$ through LDA. Next, the connection of the news document d and the topic z is built with the largest probability.

Note that for testing, the estimated LDA model can infer the topics of new documents [23]. In this way, the new documents that do not existed in the graph can be connected with the constructed graph and update their embeddings through graph convolution. Hence, the topic information can alleviate the cold start problem as well as the sparsity issue of user-item interactions.

GNN for Heterogeneous User-News-Topic Graph With the constructed heterogeneous user-news-topic graph, GNewsRec then applies GNN [9, 36, 40] to capture high-order relationships between users and news by propagating the embeddings through it. Following are the general form of computing a certain node embedding of a single GNN layer:

$$h_{\mathcal{N}_v} = \text{AGGREGATE}(\{W^t h_u^t, \forall u \in \mathcal{N}_v\}), \tag{8.11}$$

$$h_v = \sigma(W \cdot h_{\mathcal{N}_v} + b), \tag{8.12}$$

where AGGREGATE is the aggregator function to aggregate information from neighboring nodes. Here GNewsRec uses the mean aggregator which simply takes the elementwise mean of the vectors of the neighbors. $\mathcal{N}_v$ denotes the neighborhood of a certain node v and W^t is trainable transformation matrix for transforming different types of nodes h_u^t into the same space. W and b are the weight matrices and bias of one GNN layer to update the center node embedding h_v.

In particular, considering the candidate pair of user u and news d, GNewsRec uses $U(d)$ and $Z(d)$ [6] to respectively denote the set of users and topics directly connected to the news document d. In real applications, the size of $U(d)$ may vary significantly over all news documents. To keep the computational pattern of each batch fixed and more efficient, GNewsRec uniformly samples a set of neighbors $S(d)$ with fixed size for each news d instead of using all its neighbors, where the size $|S(d)| = L_u$.[7] Following Eq. 8.11 and 8.12, to characterize the topological proximity structure of news d, firstly, we compute the linear average combination of all its sampled neighbors as follows:

$$d_{\mathcal{N}} = \frac{1}{|S(d)|} \sum_{u \in S(d)} W_u u + \frac{1}{|Z(d)|} \sum_{z \in Z(d)} W_z z, \tag{8.13}$$

where $u \in \mathbb{R}^D$ and $z \in \mathbb{R}^D$ are the representations of the neighboring user and topic of news d. Note that u and z are initialized randomly, while d is initialized with the text feature embedding obtained from text information extractor (Sectioin 4.1). $W_u \in \mathbb{R}^{D \times D}$ and $W_z \in \mathbb{R}^{D \times D}$ are respectively the trainable transformation matrix for users and topics, which map them from the different spaces to the same space of news embeddings.

Then the candidate news embedding is updated with the neighborhood representation $d_{\mathcal{N}}$ by:

---

[6] Here, we assume each news has only one topic, i.e., $|Z(d)| = 1$

[7] $S(d)$ may contain duplicates if $|U(d)| < L_u$. If $U(d) = \emptyset$, then $S(d) = \emptyset$.

$$\widetilde{d} = \sigma(W^1 \cdot d_{\mathcal{N}} + b^1), \tag{8.14}$$

where σ is the nonlinear function $ReLU$, and $W^1 \in \mathbb{R}^{D \times D}$ and $b^1 \in \mathbb{R}^D$ are transformation weight and bias of the first l9koayer of GNN, respectively.

This is a single layer GNN, where the final embedding of the candidate news is only dependent on its immediate neighbors. In order to capture high-order relationships between users and news, GNewsRec can extend the GNN from one layer to multiple layers, propagating the embeddings in a broader and deeper way. As shown in Fig. 8.5, 2-order news embeddings can be obtained as follows. We first get its 1-hop neighboring user embeddings u_l and topic embeddings z by aggregating their neighboring news embeddings using Eq. 8.11 and 8.12. Then their embeddings u_l and z are aggregated to get 2-order news embeddings $\widetilde{d}$. Generally speaking, the H-order representation of an news is a mixture of initial representations of its neighbors up to H hops away.

Through the GNN, one can get the final user and news embeddings u_l and $\widetilde{d}$ with high-order information encoded. The user embeddings learned with complete user click history are supposed to capture the relatively stable long-term user interests. However, we argue that a user could be temporally attracted to certain things, namely, a user has short-term interest, which should also be considered in personalized news recommendation.

8.3.3.3 Short-term User Interest Modeling

This subsection presents how to model a user's short-term interest using his/her recent click history through an attention based LSTM model. We pay attention to not only the news contents but also the sequential information.

Attention over Contents Given a user u with his/her latest l clicked news $\{d_1, d_2, ..., d_l\}$[8], GNewsRec uses an attention mechanism to model the different impacts of the user's recent clicked news on the candidate news d:

$$u_j = tanh(W' d_j + b'), \tag{8.15}$$

$$u = tanh(Wd + b), \tag{8.16}$$

$$\alpha_j = \frac{exp(v^T (u + u_j))}{\sum_j exp(v^T (u + u_j))}, \tag{8.17}$$

$$u_c = \sum_j \alpha_j d_j, \tag{8.18}$$

where u_c is the user's current content-level interest embedding, α_j is the impact weight of clicked news $d_j (j = 1, \cdots, l)$ on candidate news d, $W', W \in \mathbb{R}^{D \times D}$, d_j, b_w, b_t, $v^T \in \mathbb{R}^D$, D is the dimension of news embedding.

[8] If the click history sequence length is less than l, it will be padded with zero embeddings.

Attention over Sequential Information Besides applying attention mechanism to model user current content-level interest, we also take attention of the sequential information of the latest clicked news, thus an attention based LSTM [10] is used to capture the sequential features.

As is shown in Fig. 8.4, LSTM takes user's clicked news embeddings as input, and output the user's sequential feature representation. Since each user's current click is affected by previous clicked news, the attention mechanism described above (for content-level interest modeling) is applied on each hidden state h_j and their previous hidden states $\{h_1, h_2, \cdots, h_{j-1}\}$ ($h_j = \text{LSTM}(h_{j-1}, d_j)$) of the LSTM to obtain richer sequential feature representation $s_j (j = 1, \cdots, l)$ at different click times. These features $(s_1, \cdots, s_l)$ are integrated by a CNN to get the final sequential feature representation $\widetilde{s}$ of user's latest l clicked history.

The concatenation of current content-level interest embedding and the sequence-level embedding is fed into a linear layer and get the final user's short-term interest embedding:

$$u_s = W_s[u_c; \widetilde{s}], \tag{8.19}$$

where $W_s \in \mathbb{R}^{D \times 2D}$ is the parameter matrix.

8.3.3.4 Prediction and Training

Finally, the user embedding u is computed by taking linear transformation over the concatenation of the long-term and short-term embedding vectors:

$$u = W[u_l; u_s], \tag{8.20}$$

where $W \in \mathbb{R}^{d \times 2d}$ is a parameter matrix to fuse into the final user embedding.

Then we compare the final user embedding u to the candidate news embedding $\widetilde{d}$, the probability of user u clicking news d is predicted by a DNN:

$$\hat{y} = DNN(u, \widetilde{d}). \tag{8.21}$$

To train our proposed model GNewsRec, positive samples are selected from the existing observed clicked reading history and equal amount of negative samples from unobserved reading. A training sample is denoted as $X = (u, x, y)$, where x is the candidate news to predict whether click or not. For each positive input sample, $y = 1$, otherwise $y = 0$. After our model, each input sample has a respective estimated probability $\hat{y} \in [0, 1]$ of the user whether will click the candidate news x. The cross-entropy loss is used as lost function:

$$\mathcal{L} = -\{\sum_{X \in \Delta^+} y \; log \; \hat{y} + \sum_{X \in \Delta^-} (1 - y) \; log(1 - \hat{y})\} + \lambda \|W\|_2, \tag{8.22}$$

where Δ^+ is the positive sample set and Δ^- is the negative sample set, $\|W\|_2$ is the L2 regularization to all the trainable parameters and λ is the penalty weight. Besides, dropout and early stopping are also applied to avoid over-fitting.

Table 8.4 Statistics of the dataset.

Number	Adressa-1week	Adressa-10week
#users	537,627	590,673
#news	14,732	49,994
#events	2,527,571	23,168,411
#vocabulary	116,603	279,214
#entity-type	11	11
#average words per title	4.03	4.10
#average entity per news	22.11	21.29

8.3.4 Experiments

8.3.4.1 Experimental Settings

Datasets Experiments are conducted on a real-world online news dataset Adressa[9] [7], which is a click log data set with approximately 20 million page visits from a Norwegian news portal as well as a sub-sample with 2.7 million clicks. We use the two light versions, named Adressa-1week, which collects news click logs as long as 1 week (from 1 January to 7 January 2017), and Adressa-10week, which collects 10 weeks (from 1 January to 31 March 2017) dataset. Following DAN [51], for each event, we just select the (sessionStart, sessionStop)[10], user id, news id, time-stamp, the title and profile of news for building our datasets. In terms of data splits, for the Adressa-1week dataset, the data is split as: the first 5 days' history data for graph construction and the latest l news clicked in the 5 days for short-term interest modeling, the 6-th day's for generating training pairs <u, d>, 20% of the last day's for validation and the left 80% for testing. Note that during testing, we reconstruct the graph with the previous 6 days' history data and use the latest l news clicked in the 6 days to model short-term user interest. Similarly, for the Adressa-10week dataset, in training period, the previous 50 days' data is used for graph construction, the following 10 days' for generating training pairs, 20% of the left 10 days' for validation and 80% for testing. The statistics of the final datasets are shown in Table 8.4.

Baselines The following state-of-the-art methods are used as baselines: **DMF** [45], a deep matrix factorization model, uses multiple non-linear layers to process raw rating vectors of users and items but ignores the news contents and take the implicit feedback as its input. **DeepWide** [4], a deep learning based model, combines the the linear model (Wide) and feed-forward neural network (Deep) to model low- and high-level feature interactions simultaneously. **DeepFM** [8], a general deep model for recommendation, combines a component of factorization machines and a component of deep neural networks that share the input to model low- and high-level feature interactions. **DKN** [35], a deep content based recommendation framework,

[9] http://reclab.idi.ntnu.no/dataset/

[10] sessionStart and sessionStop determine the session boundaries.

Table 8.5 Comparison of Different Models.

Model	Adressa-1week		Adressa-10week	
	AUC(%)	F1(%)	AUC(%)	F1(%)
DMF	55.66	56.46	53.20	54.15
DeepWide	68.25	69.32	73.28	69.52
DeepFM	69.09	61.48	74.04	65.82
DKN	75.57	76.11	74.32	72.29
DAN	75.93	74.01	76.76	71.65
GNewsRec	**81.16**	**82.85**	**78.62**	**81.01**

fuses semantic-level and knowledge-level representations of news by a multi-channel CNN. **DAN** [51], a deep attention based neural network for news recommendation, improves DKN [35] by considering the user's click equence information. Note that all the baseline models are based on deep neural networks. DMF is a collaborative filtering based model, while the others are all content based.

8.3.4.2 Comparisons of Different Models

In this subsection, experiments are conducted to compare our model with the state-of-the-art baseline models on two datasets, and the results are report in Table 8.5 in terms of AUC and $F1$ metrics.

As one can see from Table 8.5, our model consistently outperforms all the baselines on both datasets by more than 10.67% on F1 and 2.37% on AUC. We attribute the significant superiority of our model to its three advantages: (1) Our model constructs a heterogeneous user-news-topic graph and learns better user and news embeddings with high-order information encoded by GNN. (2) Our model considers not only the long-term user interest but also the short-term interest. (3) The topic information incorporated in the heterogeneous graph can help better reflect a user's interest and alleviate the sparsity issue of user-item interactions. The news items with few user clicks can still aggregate neighboring information through the topics. One can also find that all content-based models achieve better performance than the CF-based model DMF. This is because CF-based methods cannot work well in news recommendation due to cold-start problem. Our model as a hybrid model can combine the advantages of content-based models and CF-based model. In addition, new arriving documents without user clicks can also be connected to the existing graph via topics, and update their embeddings through GNN. Thus, our model can achieve better performance.

8.3.4.3 Comparisons of GNewsRec Variants

Further, we compare among the variants of GNewsRec to demonstrate the efficacy of the design of our model with respect to the following aspects: GNN for learning user

Table 8.6 Comparison of GNewsRec variants.

Model	Adressa-1week		Adressa-10week	
	AUC(%)	F1(%)	AUC(%)	F1(%)
GNewsRec without GNN	75.93	74.01	76.76	71.65
GNewsRec without short-term interest	79.00	80.53	77.03	80.21
GNewsRec without topic	79.27	80.73	77.21	80.32
GNewsRec	**81.16**	**82.85**	**78.62**	**81.01**

and news embeddings with high-order structure information encoded, combining of long-term and short-term user interests, and the incorporation of topic information. We can draw the following conclusion from the results shown in Table 8.6.

Firstly, as one can see from Table 8.6, there is a great decline in performance when the GNN module is removed while modeling long-term user interest and news, which encodes high-order relationships on the graph. This demonstrates the superiority of our model by constructing a heterogeneous graph and applying GNN to propagate the embeddings over the graph. Secondly, removing short-term interest modeling module will decrease the performance by around 2% in terms of both AUC and F1. It demonstrates that considering both long-term and short-term user interests is necessary. Thirdly, compared to the variant model without topic information, GNewsRec achieves significant improvements on both metrics. This is because that the topic information can alleviate the user-item sparsity issue as well as the cold-start problem. New documents with few user clicks can still aggregate neighboring information through topics. GNewsRec without topic performs slightly better than GNewsRec without short-term interest modeling, which shows that considering short-term interest is important.

The more detailed method description and experiment validation can be seen in [11].

8.4 News Recommendation with Preference Disentanglement

8.4.1 Overview

One of the core problems in news recommendation is how to learn better representations of users and news. Existing deep learning based methods [24, 35, 44, 51, 2] usually only focus on news contents, and seldom consider the collaborative signal in the form of high-order connectivity underlying the user-news interactions. Capturing high-order connectivity among users and news could deeply exploit structure characteristics and alleviate the sparsity, thus improving the recommendation performance [41]. For example, as shown in Fig. 8.6, the high-order relationship u_1–d_1–u_2 indicates the behavior similarity between u_1 and u_2 so that we may recommend d_3 to u_2 since u_1 clicked d_3, while d_1–u_2–d_4 implies d_1 and d_4 may have similar target users. Moreover, users may click different news due to their great diversity

of preferences. The real-world user-news interactions arise from highly complex latent preference factors. For example, as shown in Fig. 8.6, u_2 might click d_1 under her preference to entertainment news, while chooses d_4 due to her interest in politics. When aggregating neighborhood information along the graph, different importance of neighbors under different latent preference factors should be considered. Learning representations that uncover and disentangle these latent preference factors can bring enhanced expressiveness and interpretability, which nevertheless remains largely unexplored by the existing literatures on news recommendation.

To address the above issues, the user-news interactions are modeled as a bipartite graph and propose a novel **G**raph Neural **N**ews Recommendation Model with **U**nsupervised preference **D**isentanglement (**GNUD**). The model is able to capture the high-order connectivities underlying the user-news interactions by propagating the user and news representations along the graph. Furthermore, the learned representations are disentangled by a neighborhood routing mechanism, which dynamically identifies the latent preference factors that may have caused the click between a user and news, and accordingly assigning the news to a subspace that extracts and convolutes features specific to that factor. To force each disentangled subspace to independently reflect an isolated preference, a novel preference regularizer is also designed to maximize the mutual information measuring dependency between two random variables in information theory to strengthen the relationship between the preference factors and the disentangled embeddings. It further improves the disentangled representations of users and news.

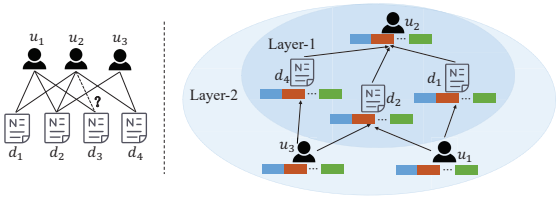


Fig. 8.6 An illustration of user-news interaction graph and high-order connectivity. The representations of user and news are disentangled with latent preference factors.

8.4.2 The GNUD Model

The news recommendation problem can be formalized as follows. Given the user-news historical interactions $\{(u,d)\}$, it aims to predict whether a user u_i will click a candidate news d_j that she has not seen before. In the following, we first introduce the news content information extractor which learns a news representation h_d from news content. Then we detail the proposed graph neural model GNUD with unsupervised preference disentanglement for news recommendation as shown in Fig. 8.7. The

model not only exploits the high-order structure information underlying the user-news interaction graph but also considers the different latent preference factors causing the clicks between users and news. A novel preference regularizer is also introduced to force each disentangled subspace independently reflect an isolated preference factor.

8.4.2.1 News Content Information Extractor

For a news article d, the original paper considers the title T and profile P (a given set of entities E and their corresponding entity types C from the news content) as features. The entities E and their corresponding entity types C are already given in the datasets. Each news title T consists of a word sequence $T = \{w_1, w_2, \cdots, w_m\}$. Each profile P contains a sequence of entities defined as $E = \{e_1, e_2, \cdots, e_p\}$ and corresponding entity types $C = \{c_1, c_2, \cdots, c_p\}$. It's denoted that the title embedding as $T = [w_1, w_2, \cdots, w_m]^T \in R^{m \times n_1}$, entity set embedding as $E = [e_1, e_2, \cdots, e_p]^T \in R^{p \times n_1}$, and the entity-type set embedding as $C = [c_1, c_2, \cdots, c_p]^T \in R^{p \times n_2}$. w, e and c are respectively the embedding vectors of word w, entity e, and entity type c. n_1 and n_2 are the dimension of word (entity) and entity-type embeddings. These embeddings can be pre-trained from a large corpus or randomly initialized. Following [51], it defines the profile embedding $P = [e_1, g(c_1), e_2, g(c_2), \cdots, e_p, g(c_p)]^T$ where $P \in R^{2p \times n_1}$. $g(c)$ is the transformation function as $g(c) = M_c c$, where $M_c \in R^{n_1 \times n_2}$ is a trainable transformation matrix.

Following DAN [51], we also use two parallel convolutional neural networks (PCNN) taking the title T and profile P of news as input to learn the title-level and profile-level representation $\widehat{T}$ and $\widehat{P}$ for news. Finally we concatenate $\widehat{T}$ and $\widehat{P}$, and get the final news representation h_d through a fully connected layer f:

$$h_d = f([\widehat{T}; \widehat{P}]). \tag{8.23}$$

8.4.2.2 Heterogeneous Graph Encoder

As illustrated in Fig. 8.7, to capture the high-order connectivity underlying the user-news interactions, the user-news interactions are modeled as a bipartite graph $\mathcal{G} = \{\mathcal{U}, \mathcal{D}, \mathcal{E}\}$, where $\mathcal{U}$ and $\mathcal{D}$ are the sets of users and news, $\mathcal{E}$ is the set of edges and each edge $e = (u, d) \in \mathcal{E}$ indicates that user u explicitly clicks news d. The model GNUD enables information propagation among users and news along the graph, thus capturing the high-order relationships among users and news. Additionally, GNUD learns disentangled embeddings that uncover the latent preference factors behind user-news interactions, enhancing expressiveness and interpretability. In the following, one single graph covolution layer with preference disentanglement is presented.

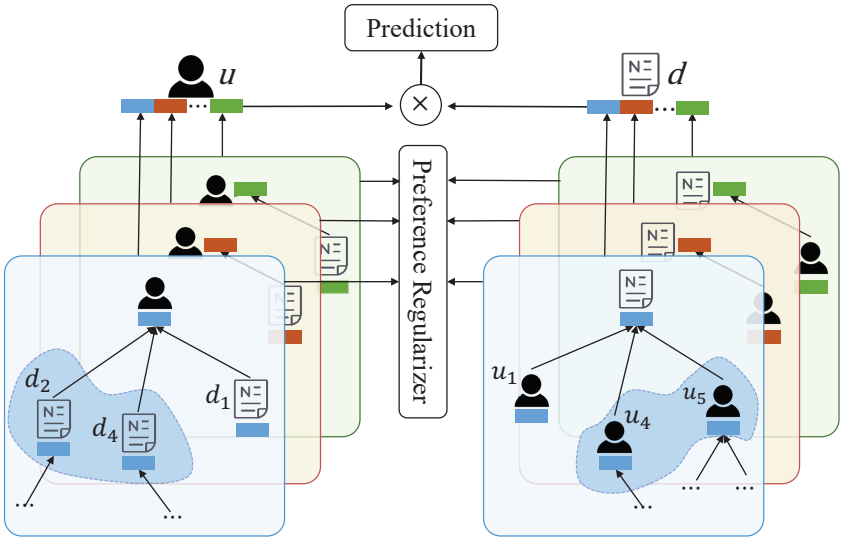


Fig. 8.7 Illustration of the proposed model GNUD.

Graph Convolution Layer with Preference Disentanglement Given the user-news bipartite graph $\mathcal{G}$ where the user embedding h_u is randomly initialized and news embedding h_d is obtained with the news content information extractor (Section 8.4.2.1), a graph convolutional layer aims to learn the representation y_u of a node u by aggregating its neighbors' features:

$$y_u = \text{Conv}(h_u, \{h_d : (u,d) \in \mathcal{E}\}). \tag{8.24}$$

Considering that users' click behaviors could be caused by different latent preference factors, it's proposed to derive a layer Conv$(\cdot)$ such that the output y_u and y_d are disentangled representations. Each disentangled component reflect one preference factor related to the user or news. The learned disentangled user and news embeddings can bring enhanced expressiveness and interpretability. Assuming that there are K factors, we would like to let y_u and y_d be composed of K independent components: $y_u = [z_{u,1}, z_{u,2}, \cdots, z_{u,K}]$, $y_d = [z_{d,1}, z_{d,2}, \cdots, z_{d,K}]$, where $z_{u,k}$ and $z_{d,k} \in R^{\frac{l_{out}}{K}}$ $(1 \le k \le K)$ (l_{out} is the dimension of y_u and y_d), respectively characterizing the k-th aspect of user u and news d related to the k-th preference factor. Note that in the following, we can focus on user u and describe the learning process of its representation y_u. The news d can be learned similarly, which is omitted.

Formally, given a u-related node $i \in \{u\} \bigcup \{d : (u,d) \in \mathcal{E}\}$, a subspace-specific projection matrix W_k is used to map the feature vector $h_i \in R^{l_{in}}$ into the k-th preference related subspace:

$$s_{i,k} = \frac{\text{ReLU}(W_k^\top h_i + b_k)}{\| \text{ReLU}(W_k^\top h_i + b_k) \|_2}, \tag{8.25}$$

where $W_k \in R^{l_{in} \times \frac{l_{out}}{K}}$, and $b_k \in R^{\frac{l_{out}}{K}}$. Note that $s_{u,k}$ is not equal to the final representation of the k-th component of u: $z_{u,k}$, since it has not mined any information from neighboring news yet. To construct $z_{u,k}$, the information needs mining from both $s_{u,k}$ and the neighborhood features $\{s_{d,k} : (u,d) \in \mathcal{E}\}$.

The main intuition is that when constructing $z_{u,k}$ characterizing the k-th aspect of u, it should only use the neighboring news articles d which connect with user u due to the preference factor k instead of all the neighbors. Therefore, a neighborhood routing algorithm [21] is applied to identify the subset of neighboring news that actually connect to u due to the preference factor k.

Neighborhood Routing Algorithm The neighborhood routing algorithm infers the latent preference factors behind user-news interactions by iteratively analyzing the potential subspace formed by a user and her clicked news. Formally, let $r_{d,k}$ be the probability that the user u clicks the news d due to the factor k. Then it's also the probability that we should use the news d to construct $z_{u,k}$. $r_{d,k}$ is an unobserved latent variable which can be inferred in an iterative process. The motivation of the iterative process is as follows. Given $z_{u,k}$, the value of the latent variables $\{r_{d,k} : 1 \le k \le K, (u,d) \in \mathcal{E}\}$ can be obtained by measuring the similarity between user u and her clicked news d under the k-th subspace, which is computed as Eq. 8.26. Initially, set $z_{u,k} = s_{u,k}$. On the other hand, after obtaining the latent variables $\{r_{d,k}\}$, one can find an estimate of $z_{u,k}$ by aggregating information from the clicked news, which is computed as Eq. 8.27:

$$r_{d,k}^{(t)} = \frac{\exp({z_{u,k}}^{(t)\top} s_{d,k})}{\sum_{k'=1}^{K} \exp({z_{u,k}}^{(t)\top} s_{d,k})}, \tag{8.26}$$

$$z_{u,k}^{(t+1)} = \frac{s_{u,k} + \sum_{d:(u,d)\in\mathcal{G}} r_{d,k}^{(t)} s_{d,k}}{\| s_{u,k} + \sum_{d:(u,d)\in\mathcal{G}} r_{d,k}^{(t)} s_{d,k} \|_2}, \tag{8.27}$$

where iteration $t = 0, \cdots, T-1$. After T iterations, the output $z_{u,k}^{(T)}$ is the final embedding of user u in the k-th latent subspace and we obtain $y_u = [z_{u,1}, z_{u,2}, \cdots, z_{u,K}]$.

The above shows a single graph convolutional layer with preference disentanglement, which aggregates information from the first-order neighbors. In order to capture information from high-order neighborhood and learn high-level features, multiple layers are stacked. Specially, L layers are used to get the final disentangled representation $y_u^{(L)} \in R^{K\Delta n}$ ($K\Delta n = l_{out}$) for user u and $y_d^{(L)}$ for news d, where Δn is the dimension of a disentangled subspace.

Preference Regularizer Naturally, we hope each disentangled subspace can reflect an isolated latent preference factor independently. Since there are no explicit labels indicating the user preferences in the training data, a novel preference regularizer is

also designed to maximize the mutual information measuring dependency between two random variables in information theory to strengthen the relationship between the preference factors and the disentangled embeddings. According to [46], the mutual information maximization can be converted to the following form.

Given the representation of a user u in k-th $(1 \leq k \leq K)$ latent subspace, the preference regularizer $P(k|z_{u,k})$ estimates the probability of the k-th subspace (w.r.t. the k-th preference) that $z_{u,k}$ belongs to:

$$P(k|z_{u,k}) = \text{softmax}(W_p \cdot z_{u,k} + b_p), \tag{8.28}$$

where $W_p \in R^{K \times \Delta n}$, and parameters in the regularizer $P(\cdot)$ are shared with all the users and news.

8.4.2.3 Model Training

For model training, a fully-connected layer is added:

$$y'_u = W^{(L+1)^\top} y_u^{(L)} + b^{(L+1)}, \tag{8.29}$$

where $W^{(L+1)} \in R^{K\Delta n \times K\Delta n}$, $b^{(L+1)} \in R^{K\Delta n}$. Then simple dot product is used to compute the news click probability score:

$$\hat{s}\langle u, d\rangle = y'^{\top}_u y'_d. \tag{8.30}$$

Once obtaining the click probability scores $\hat{s}\langle u, d\rangle$, we define the following base loss function for training sample (u, d) with the ground truth $y_{u,d}$:

$$\mathcal{L}_1 = -[y_{u,d} \ln(\hat{y}_{u,d}) + (1 - y_{u,d}) \ln(1 - \hat{y}_{u,d})], \tag{8.31}$$

where $\hat{y}_{u,d} = \sigma(\hat{s}\langle u, d\rangle)$. Then we add the preference regularization term of both u and d, which can be formulated as:

$$\mathcal{L}_2 = -\frac{1}{K} \sum_{k=1}^{K} \sum_{i \in \{u,d\}} \ln P(k|z_{i,k})[k]. \tag{8.32}$$

Finally, the overall training loss can be rewritten as:

$$\mathcal{L} = \sum_{(u,d) \in \mathcal{T}_{\text{train}}} ((1 - \lambda)\mathcal{L}_1 + \lambda \mathcal{L}_2) + \eta \|\Theta\|, \tag{8.33}$$

where $\mathcal{T}_{\text{train}}$ is training set. For each positive sample (u, d), a negative instance is sampled from unobserved reading history of u for training. λ is a balance coefficient. η is the regularization coefficient and $\|\Theta\|$ denotes all the trainable parameters.

Note that during training and testing, the news that have not been read by any users are taken as isolated nodes in the graph. Their representations are based on

only content feature h_d without neighbor aggregation, and can also be disentangled via Eq. 8.25.

8.4.3 Experiments

8.4.3.1 Experimental Settings

For datasets, we use the same settings of the dataset as Section 8.3. For comparisons, in addition to the aforementioned baselines in Section 8.3, the following state-of-the-art methods are included to be further compared with GNUD. **LibFM** [27], a feature-based matrix factorization method, concatenates the TF-IDF vectors of news title and profile as input. **CNN** [15], applies two parallel CNNs to word sequences in news titles and profiles respectively and concatenates them as news features. **DSSM** [13], a deep structured semantic model, models the user's clicked news as the query and the candidate news as the documents. **GNewsRec** [11], a graph neural network based method, combines long-term and short term interest modeling for news recommendation.

8.4.3.2 Comparision of Different Methods

Table 8.7 The performance of different methods on news recommendation.

Methods	Adressa-1week		Adressa-10week	
	AUC	F1	AUC	F1
LibFM	61.20±1.29	59.87±0.98	63.76±1.05	62.41±0.72
CNN	67.59±0.94	66.33±1.44	69.07±0.95	67.78±0.69
DSSM	68.61±1.02	69.92±1.13	70.11±1.35	70.96±1.56
DeepWide	68.25±1.12	69.32±1.28	73.28±1.26	69.52±0.83
DeepFM	69.09±1.45	61.48±1,31	74.04±1.69	65.82±1.18
DMF	55.66±0.84	56.46±0.97	53.20±0.89	54.15±0.47
DKN	75.57±1.13	76.11±0.74	74.32±0.94	72.29±0.41
DAN	75.93±1.25	74.01±0.83	76.76±1.06	71.65±0.57
GNewsRec	81.16±1.19	82.85±1.15	78.62±1.38	81.01±0.64
GNUD w/o Disen	78.33±1.29	79.09±1.22	78.24±0.13	80.58±0.45
GNUD w/o PR	83.12±1.53	81.67±1.56	80.61±1.07	80.92±0.31
GNUD	**84.01±1.16**	**83.90±0.58**	**83.21±1.91**	**81.09±0.23**

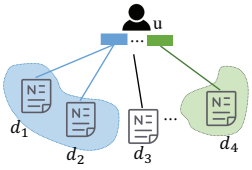
The comparisons between different methods are summarized in Table 8.7. One can observe that the proposed model GNUD consistently outperforms all the state-of-the-art baseline methods on both datasets. GNUD improves the best deep neural models DKN and DAN more than 6.45% on AUC and 7.79% on F1 on both datasets. The main reason is that GNUD fully exploits the high-order structure information in the user-news interaction graph, learning better representations of users and news.

Compared to the best-performed baseline method GNewsRec, GNUD achieves better performance on both datasets in terms of both AUC (+2.85% and +4.59% on the two datasets, respectively) and F1 (+1.05% and +0.08%, respectively). This is because that GNUD considers the latent preference factors that cause the user-news interactions and learns representations that uncover and disentangle these latent preference factors, which enhance expressiveness. From Table 8.7, one can also see that all the content-based methods outperform the CF based model DMF. This is because CF based methods suffer a lot from cold-start problem since most news are new coming. Except for DMF, all the deep neural network based baselines (e.g., CNN, DSSM, DeepWide, DeepFM, etc.) significantly outperform LibFM, which shows that deep neural models can capture more implicit but informative features for user and news representations. DKN and DAN further improve other deep neural models by incorporating external knowledge and applying a dynamic attention mechanism.

8.4.3.3 Comparison of GNUD variants

To further demonstrate the efficacy of the design of the model GNUD, we compare among the variants of it. As one can see from the last three lines in Table 8.7, when the preference disentanglement is removed, the performance of the model GNUD w/o Disen (GNUD without preference disentanglement) drops largely by 5.68% and 4.97% in terms of AUC on the two datasets (4.81% and 0.51% on F1), respectively. This observation demonstrates the effectiveness and necessity of preference disentangled representations of users and news. Compared to GNUD w/o PR (GNUD without preference regularizer), one can see that introducing the preference regularizer which enforces each disentangled embedding subspace independently reflect an isolated preference, can bring performance gains on both AUC (+0.89% and +2.6%, respectively) and F1 (+2.23% and +0.17%, respectively).

8.4.3.4 Case Study



News	Keywords
d_1	norway oljebransjen (Norway oil industry), norskehavet (Norwegian sea), helgelandskysten (Helgeland coast), hygen (hygen), energy (energy), trondheim (a city)
d_2	Statkraft (State Power Corporation of Norway), trønderenergi (tronder energy), snillfjord (snill fjord), trondheimsfjorden (trondheim fjord), vindkraft (wind power), energy (energy)
d_3	Bolig (residence), hage (garden), hjemme (home), fossen (waterfall), hus (house), home (home)
d_4	health-and-fitness (health and fitness), mørk sjokolade (dark chocolate), vitaminrike (vitamin), olivenolje (olive oil), grønnsaker (vegetables), helse (health)

Fig. 8.8 Visualization of a user's clicked news which belong to different disentangled subspaces w.r.t. different preference factors. Here six keywords (translated into English) is used to illustrate a news.

To intuitively demonstrate the efficacy of GNUD, we randomly sample a user u and extract her logs from the test set. The representation of user u is disentangled into $K = 7$ subspaces and we randomly sample 2 subspaces. For each one, the top news is visualized that user u pay most attention to (with the probability $r_{d,k}$ larger than a threshold). As shown in Fig. 8.8, different subspaces relect different preference factors. For example, one subspace (shown in blue) is related to "energy" as the top two news contain the keywords such as "oil industry", "hygen" and "wind power". The other subspace (shown in green) may indicate the latent preference factor about "healthy diet" as the related news contain the keywords such as "health", "vitamin" and "vegetables". The news d_3 about home has low probability in the both subspaces. It does not belong to any of the two preferences.

The more detailed method description and experiment validation can be seen in [12].

8.5 Conclusion

In recent years, heterogeneous graph based text mining has become a very popular research and industrial application direction. Considering the strong power of integrating additional information and modeling the relations between objects, heterogeneous graphs are widely explored to alleviate the data sparsity problem that is common in many tasks and applications. Therefore, it has gradually attracted attention from more researchers in the field of text mining that constructing a heterogeneous graph followed by a heterogeneous graph representation method. In this chapter, we have introduced three methods for text mining. HGAT, GNewsRec and GUND respectively construct a heterogeneous graph to model the input short texts or long news. Hence the following designed heterogeneous graph neural network can make better use of the textual and auxiliary information and successfully outperforms.

In the future, we can employ HG modeling to explore more other NLP tasks, such as relation extraction, question answering, etc. Moreover, it is also a valuable research direction to integrate graph-structured external knowledge, such as the knowledge graph, into the constructed heterogeneous graph for further improvement.

References

1. Aggarwal, C.C., Zhai, C.: A survey of text classification algorithms. Mining Text Data pp. 163–222 (2012)
2. An, M., Wu, F., Wu, C., Zhang, K., Liu, Z., Xie, X.: Neural news recommendation with long-and short-term user representations. In: ACL, pp. 336–345 (2019)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of machine Learning research **3**(Jan), 993–1022 (2003)

4. Cheng, H.T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., et al.: Wide & deep learning for recommender systems. In: DLRS@RecSys, pp. 7–10 (2016)
5. Das, A.S., Datar, M., Garg, A., Rajaram, S.: Google news personalization: scalable online collaborative filtering. In: WWW, pp. 271–280 (2007)
6. De Francisci Morales, G., Gionis, A., Lucchese, C.: From chatter to headlines: harnessing the real-time web for personalized news recommendation. In: WSDM, pp. 153–162 (2012)
7. Gulla, J.A., Zhang, L., Liu, P., Özgöbek, Ö., Su, X.: The adressa dataset for news recommendation. In: ICWI, pp. 1042–1048 (2017)
8. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: Deepfm: a factorization-machine based neural network for ctr prediction. In: IJCAI, pp. 1725–1731 (2017)
9. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NIPS, pp. 1024–1034 (2017)
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
11. Hu, L., Li, C., Shi, C., Yang, C., Shao, C.: Graph neural news recommendation with long-term and short-term interest modeling. Information Processing & Management **57**(2), 102,142 (2020)
12. Hu, L., Xu, S., Li, C., Yang, C., Shi, C., Duan, N., Xie, X., Zhou, M.: Graph neural news recommendation with unsupervised preference disentanglement. In: ACL, pp. 4255–4264 (2020)
13. Huang, P.S., He, X., Gao, J., Deng, L., Acero, A., Heck, L.: Learning deep structured semantic models for web search using clickthrough data. In: CIKM, pp. 2333–2338 (2013)
14. IJntema, W., Goossen, F., Frasincar, F., Hogenboom, F.: Ontology-based news recommendation. In: EDBT/ICDT Workshops, p. 16 (2010)
15. Kim, Y.: Convolutional neural networks for sentence classification. In: EMNLP, pp. 1746–1751 (2014)
16. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
17. Li, L., Wang, D., Li, T., Knox, D., Padmanabhan, B.: Scene: a scalable two-stage personalized news recommendation system. In: SIGIR, pp. 125–134 (2011)
18. Linmei, H., Yang, T., Shi, C., Ji, H., Li, X.: Heterogeneous Graph Attention Networks for Semi-supervised Short Text Classification. In: EMNLP, pp. 4821–4830 (2019)
19. Liu, M., Wang, X., Nie, L., Tian, Q., Chen, B., Chua, T.S.: Cross-modal moment localization in videos. In: MM, pp. 843–851 (2018)
20. Liu, P., Qiu, X., Huang, X.: Recurrent neural network for text classification with multi-task learning. In: IJCAI, pp. 2873–2879 (2016)
21. Ma, J., Cui, P., Kuang, K., Wang, X., Zhu, W.: Disentangled graph convolutional networks. In: ICML, pp. 4212–4221 (2019)
22. Meng, Y., Shen, J., Zhang, C., Han, J.: Weakly-supervised neural text classification. In: CIKM, pp. 983–992 (2018)
23. Newman, D., Smyth, P., Welling, M., Asuncion, A.U.: Distributed inference for latent dirichlet allocation. In: NIPS, pp. 1081–1088 (2008)
24. Okura, S., Tagami, Y., Ono, S., Tajima, A.: Embedding-based news recommendation for millions of users. In: KDD, pp. 1933–1942 (2017)
25. Pang, B., Lee, L.: Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In: ACL, pp. 115–124 (2005)
26. Phan, X.H., Nguyen, L.M., Horiguchi, S.: Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In: WWW, pp. 91–100 (2008)
27. Rendle, S.: Factorization machines with libfm. ACM Transactions on Intelligent Systems and Technology **3**(3), 57 (2012)
28. Shimura, K., Li, J., Fukumoto, F.: HFT-CNN: Learning hierarchical category structure for multi-label short text categorization. In: EMNLP, pp. 811–816. Brussels, Belgium (2018)
29. Sinha, K., Dong, Y., Cheung, J.C.K., Ruths, D.: A hierarchical neural attention-based text classifier. In: EMNLP, pp. 817–823. Brussels, Belgium (2018)

30. Song, G., Ye, Y., Du, X., Huang, X., Bie, S.: Short text classification: A survey. Journal of Multimedia **9**(5), 635 (2014)
31. Tang, J., Qu, M., Mei, Q.: Pte: Predictive text embedding through large-scale heterogeneous text networks. In: KDD, pp. 1165–1174 (2015)
32. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: NIPS, pp. 5998–6008 (2017)
33. Vitale, D., Ferragina, P., Scaiella, U.: Classification of short texts by deploying topical annotations. In: ECIR, pp. 376–387 (2012)
34. Wang, C., Blei, D.M.: Collaborative topic modeling for recommending scientific articles. In: KDD, pp. 448–456 (2011)
35. Wang, H., Zhang, F., Xie, X., Guo, M.: DKN: Deep knowledge-aware network for news recommendation. In: WWW, pp. 1835–1844 (2018)
36. Wang, H., Zhao, M., Xie, X., Li, W., Guo, M.: Knowledge graph convolutional networks for recommender systems. In: WWW, pp. 3307–3313 (2019)
37. Wang, J., Wang, Z., Zhang, D., Yan, J.: Combining knowledge with deep convolutional neural networks for short text classification. In: IJCAI, pp. 2915–2921 (2017)
38. Wang, S., Manning, C.D.: Baselines and bigrams: Simple, good sentiment and topic classification. In: ACL, pp. 90–94 (2012)
39. Wang, X., Chen, R., Jia, Y., Zhou, B.: Short text classification using wikipedia concept based document representation. In: ICITA, pp. 471–474 (2013)
40. Wang, X., He, X., Cao, Y., Liu, M., Chua, T.S.: Kgat: Knowledge graph attention network for recommendation. In: KDD, pp. 950–958 (2019)
41. Wang, X., He, X., Wang, M., Feng, F., Chua, T.S.: Neural graph collaborative filtering. In: SIGIR, pp. 165–174 (2019)
42. Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., Yu, P.S.: Heterogeneous graph attention network. In: WWW, pp. 2022–2032 (2019)
43. Wang, X., Yu, L., Ren, K., Tao, G., Zhang, W., Yu, Y., Wang, J.: Dynamic attention deep model for article recommendation by learning human editors' demonstration. In: KDD, pp. 2051–2059 (2017)
44. Wu, C., Wu, F., An, M., Huang, J., Huang, Y., Xie, X.: Npa: Neural news recommendation with personalized attention. In: KDD, pp. 2576–2584 (2019)
45. Xue, H.J., Dai, X., Zhang, J., Huang, S., Chen, J.: Deep matrix factorization models for recommender systems. In: IJCAI, pp. 3203–3209 (2017)
46. Yang, C., Sun, M., Yi, X., Li, W.: Stylistic chinese poetry generation via unsupervised style disentanglement. In: EMNLP, pp. 3960–3969 (2018)
47. Yang, T., Hu, L., Shi, C., Ji, H., Li, X., Nie, L.: HGAT: Heterogeneous graph attention networks for semi-supervised short text classification. ACM Transactions on Information Systems **39**(3) (2021)
48. Yao, L., Mao, C., Luo, Y.: Graph convolutional networks for text classification. In: AAAI, pp. 7370–7377 (2019)
49. Zeng, J., Li, J., Song, Y., Gao, C., Lyu, M.R., King, I.: Topic memory networks for short text classification. In: EMNLP, pp. 3120–3131 (2018)
50. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: NIPS, pp. 649–657 (2015)
51. Zhu, Q., Zhou, X., Song, Z., Tan, J., Guo, L.: Dan: Deep attention neural network for news recommendation. In: AAAI, vol. 33, pp. 5973–5980 (2019)