Knowledge-Based Systems 196 (2020) 105750

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Deep model with neighborhood-awareness for text tagging☆

Shaowei Qin [a], Hao Wu [a,*], Rencan Nie [a], Jun He [b]

[a] School of Information Science and Engineering, Yunnan University, No. 2 North Green Lake Road, Kunming 650091, China
[b] Nanjing University of Information Science and Technology, Nanjing 210044, China

ARTICLE INFO

Article history:
Received 12 December 2019
Received in revised form 5 March 2020
Accepted 7 March 2020
Available online 10 March 2020

Keywords:
Neighborhood-aware
Negative sampling
Deep neural networks
Text tagging

ABSTRACT

In recent years, many efforts based on deep learning have been made to address the issue of text tagging. However, these work generally neglect to consider the neighborhood effect which may help improve the accuracy of predictions. For this, we present a neighborhood-aware deep model for text tagging (NATT). A neural component which combines bi-directional recurrent neural network and self-attention mechanism, is firstly selected as the text encoder to encode the target document into one feature vector. Then, k-nearest-neighbor documents of the target document are identified and encoded into feature vectors one by one with the same text encoder. Simultaneously, an independent attention module is introduced to aggregate these neighboring documents into a special feature vector, which will represent features of the neighborhood. Finally, the two feature vectors are fused to match the embedding vectors of tags. To optimize the NATT model, we build the objective function with pairwise hinge loss and specially develop a neighborhood-aware negative sampling strategy to form training data. Experimental results on four datasets demonstrate that NATT outperforms some state-of-the-art neural models. Additionally, NATT is economical on achieving the best results with less training epochs and a smaller number of nearest neighbors.

1. Introduction

With the advent of Web 2.0 era, people can use a variety of application interfaces to generate a large amount of textual information. Textual documents are part of the prominent carriers to facilitate information sharing and propagation. The purpose of text tagging is to suggest tags for these documents through human or intelligent algorithms, so as to facilitate utilization. To achieve automatic emergence of document semantics, social tagging becomes popular with the help of social media [1], such as Delicious, CiteULike, Folksonomy and so on. However, the tags/labels generated by human may be casual and diverse, which is not conducive to information organization. Other ways concentrate on automatic text tagging, which can be divided into two major sub-categories: *multilabel classification* and *collaborative representation learning*. Multilabel classification is to transform sparse and high dimensional documents into dense vector representations, and then classify them to the given labels through an activation function [2]. Generally, the labels are ranked by their assigned probability values. This leads to that classification methods cannot distinguish the polarity of tags corresponding to the target documents, so it cannot effectively optimize the ranking of tags, which is a key issue regarding the top-n recommendation tasks [3]. In contrast, collaborative representation learning aims at mapping both documents and labels into a latent feature space and then performs matching using a distance or similarity metric [4,5]. The model can utilize different loss functions in combination with negative sampling to fulfill ranking optimization [4,6].

Document modeling techniques are usually shared by multilabel classification and collaborative representation learning. Traditional modeling approaches, such as vector space model and topic modeling, are built on "*bag of words model* (*BoW*)". The *BoW* ignores the contexts of words, such as surrounding words of the word and word order, which are the important features to improve word or document representation learning [7]. To overcome the shortcomings of the BoW, a major trend is to explore document modeling with neural models, such as convolutional neural networks [8,9], recurrent neural networks [10], hierarchical attention network [11], and direct document embedding [12]. These efforts follow the same research paradigm, that is, constantly improving neural models to strengthen the effect of document representation. The neighborhood effect (that is, *similar documents tend to use similar or even the same tags*), however, is always neglected when designing neural models.

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2020.105750.

* Corresponding author.
E-mail addresses: qinshaowei.ynu@gmail.com (S. Qin), haowu@ynu.edu.cn (H. Wu), rcnie@ynu.edu.cn (R. Nie), jhe@nuist.edu.cn (J. He).

https://doi.org/10.1016/j.knosys.2020.105750
0950-7051/

The principle of neighborhood effect has been well studied in item-based collaborative filtering [13], and also adjusted in constructing the widely-used *k-nearest neighbors classifier* for the task of document classification [14]. It has shown advantages of robustness and simplicity in both domains [13]. Obviously, the potential benefits can be expected by introducing neighborhood effect into the existing neural classification models. In addition, the traditional *k*-nearest neighbors model does not involve a learning phase, leads to that the performance seriously depends on the weighting strategy of nearest neighbors, where a predefined similarity measurement is always required to measure closeness of the nearest neighbors [15].

In order to explore the neighborhood effect in neural classification models and solve the weighting issue of nearest-neighbors by the way, we propose a neighborhood-aware deep model to enhance the effectiveness of text tagging. To focus on the nature of work, we choose a popular neural network component, which combines the bi-directional recurrent neural network and self-attention mechanism [16] to encode a target document into one feature vector. To inject the neighborhood information into the model, *k*-nearest-neighbors of the target document is identified and encoded one by one with the same text encoder. Simultaneously, the feature vectors of neighboring documents are synthesized into another feature vector to represent features from the neighborhood. In particular, an independent attention module is employed to automatically assign weight to overcome the difficulty in weighting nearest neighbors. Finally, the two feature vectors are fused to match the embedding vectors of tags and then to generate recommendations. We construct the objective function with a pairwise hinge loss for model training. Especially, a simple yet effective neighborhood-aware strategy is proposed in sampling negative tags to optimize the ranking of tags. Intensive experiments have been conducted and revealed the merits of our proposed method.

The main contributions of this paper are summarized as below:

- We propose NATT which exploits the neighborhood effect on both text encoding and negative sampling for text tagging.
- We show that neighborhood-awareness strategy can significantly improve the accuracy of tag predictions through intensive experiments.
- NATT is economical on achieving the best results with less training epochs and a smaller number of nearest neighbors.

The remainder of this paper is arranged as follows: we present related works in Section 2. In Section 3, we detail the neighborhood-aware neural model and discuss how to learn parameters with negative sampling. Experimental results and discussion are presented in Section 4. We draw conclusions and point out future works in Section 5.

2. Related works

As for social tagging, users apply public tags to online items, typically to make those items easier for themselves or others to find later. However, social tagging is expensive, and there are also cases of incorrect tags and inconsistencies caused by subjectivity, which brings difficulties to organize the documents. Also, without users' participation, the cold-start problem cannot be solved. Instead, we focus on the related works of automatic text tagging. In our opinion, such work basically can be divided into multi-label classification, collaborative representation learning and item-based collaborative filtering.

2.1. Multi-label classification

Multilabel classification focuses on transforming sparse and high dimensional documents into dense vector representations, and then classifying them to the given labels through an activation function [2]. Typical works include CNN-KIM [8], hierarchical attention network [11], fastText for textual documents [12].

[8] firstly reports on a series of experiments with convolutional neural networks (CNN) trained on top of pre-trained word vectors for sentence-level classification tasks. They show that a simple CNN with slight hyper-parameter tuning and static vectors achieves excellent results on sentiment analysis and question classification. Similar to CNN-KIM, Liu et al. [9] present the first attempt at applying deep learning to the extreme multi-label text classification, with a family of new CNN models particularly tailored for multi-label classification. Instead using CNN, Yang et al. [11] propose a hierarchical attention network for document classification. It consists of two levels of attention mechanisms, which attends differentially to contents at word-level and sentence-level when representing documents. Zhou et al. [10] propose to sample more meaningful information of the text by applying *2D max-pooling* operation and *2D convolution* over Bidirectional LSTM. It outperforms those methods utilizing 1D max-pooling operation or attention-based operation in some classification tasks. Wang et al. [17] propose to view text classification as a label-word joint embedding problem and introduce an attention framework that measures the compatibility of embeddings between text sequences and labels. The attention is learned on a training set of labeled samples to ensure that the relevant words are weighted higher than the irrelevant ones given a text sequence. Grave et al. [12] extend word2vec to deal with sentence and document classification. Unlike unsupervised manner from word2vec, the word features in fastText can be averaged together to form a fine sentence representation. In several tasks, fastText achieves comparable performance to some state-of-the-art deep learning methods and observes a significant acceleration.

Generally, the labels are ranked by their assigned probability values in the classification-based methods. This leads to that the classification method cannot distinguish the polarity of tags corresponding to the target documents, so it may not effectively optimize the ranking of tags which is critical regarding the top-n recommendation tasks [3]. In addition, tags are not embedded, making it sometimes difficult to capture latent semantics.

2.2. Collaborative representation learning

Compared with multilabel classification, collaborative representation learning aims at mapping both documents and labels into a latent feature space, and then perform matching using a distance or similarity metric [4,5]. It enables to utilize different objective functions in combination with negative sampling to fulfill ranking optimization of tags [4,6].

TagSpace [4] is developed for large scale ranking tasks and applied to hashtag prediction. It represents both words and the entire textual post as embeddings via a convolutional network. Also, the labels are mapping into the same latent feature space. After estimating the similarity between the label feature vector and the text feature vector, the ranking list of tags is generated to realize recommendation.

In another approach, researchers have proposed utilizing textual information in the collaborative factorization of user–item matrix. In particular, recent work concentrates extracting semantic information of textual items by deep neural networks. Collaborative deep learning (CDL) [6] combines the merits of both probabilistic matrix factorization and Stack Denoising AutoEncoder (SDAE) to learn more powerful representations of items.

CDL makes a success of using texts of items for recommendations. However, SDAE analyzes "*bag-of-words model*" of item descriptions to generate latent models and ignores the contexts of words, such as surrounding words of the word and word order. To overcome this defect, Convolutional Matrix Factorization (ConvMF) [18] exploits a convolutional network to make a deeper understanding of item descriptions and thus generate better item latent models. Although CDL and ConvMF are designed for the item recommendation, if we switch the role of users as tags, we can realize tag recommendation by exploiting the learned tag embedding representation and the multilayered neural component of textual items. This can be done by maximizing the following loss function:

$$\mathcal{L}(U,V,W) = -\sum_i \sum_j \frac{c_{i,j}}{2}(r_{ij} - u_i^T v_j)^2 - \frac{\lambda_u}{2}\sum_i \|u_i\|_2^2 - \frac{\lambda_v}{2}\sum_j \|v_j - nn(W,d_j)\|_2^2 - \frac{\lambda_w}{2}\|W\|_2^2. \quad (1)$$

$v_j \in V$ represents the latent feature vectors of existing textual items j, $u_i \in U$ represents the latent feature vectors of existing tags i. λ_u, λ_v, λ_w are regularization parameters, and c_{ij} is the confidence parameter for rating r_{ij}. The implicit rating indicating whether tag i will adopt by text j, is predicted by the inner product between their latent representations, $\hat{r}_{ij} = u_i^T \cdot v_j$. $nn(W, d_j)$ is the parameterized neural component (e.g. a convolutional neural network) to encode the text of d_j. For tagging a new input d_j, we can just take the inner product between U and $nn(W, d_j)$, and ranking all the existing tags against these relevance scores, a.k.a, $\hat{r}_{ij} = u_i^T \cdot nn(W, d_j)$. For the details of model learning, readers can refer to the work presented in [19].

2.3. Item-based collaborative filtering

In addition to multilabel classification and collaborative representation learning, item-based collaborative filtering [20] can be used as a classical non-parametric method in the text tagging. It predicts labels for a target document d by aggregating opinion of neighboring documents as follows:

$$r(d,t) = \sum_{d' \in \mathcal{N}_d} s(d,d') * r(d',t), \quad (2)$$

where $\mathcal{N}(d)$ is the k-nearest-neighbors of the document d, $s(d, d')$ is the similarity function of documents, $r(d, t)$ is the relevance between a label t and the document d. By ranking all candidate tags through $r(d, t)$, the top-n tags can be selected for tagging d. For simplicity, we call this method as ItemKNN.

ItemKNN can use the neighborhood effect to improve the accuracy of tag predictions and provide the recommendations with a better explainability. The disadvantage is that the performance seriously depends on the selection of similarity measurement. The attention strategies have been successfully applied in collaborative filtering to aggregate the embeddings of items or components [21,22], which provides more intuitive outcomes than several predefined aggregation strategies, such as concatenation, average pooling and max pooling [22]. To overcome the disadvantage of ItemKNN, we use the attention model to automatically assign weights in the fusion process of the representation vectors of neighboring texts.

3. Methodology

3.1. Neural component for text encoding

Many models can be used to extract semantic representation of text (*a.k.a* encoding text) with the booming of deep learning. To focus on developing a neighborhood-aware prediction model for text tagging, we just choose a popular text-encoder, which combines bi-directional recurrent neural network (RNN) and self-attention mechanism [16]. On one hand, the semantics of words without importance can be ignored or kept small values with RNN, thus leaving more important semantics. Further, bi-directional RNN can hold complete semantics by following consideration of the forward and reverse sequence of text. Finally, attention mechanism provides an opportunity to capture the importance of words on matching tags. On the other hand, this model has been taken as a popular component for many natural language processing tasks [11,23], as well as modeling user interaction sequences and building sequential recommender systems [24]. Here, we briefly introduce its layered architecture, shown in Fig. 1.

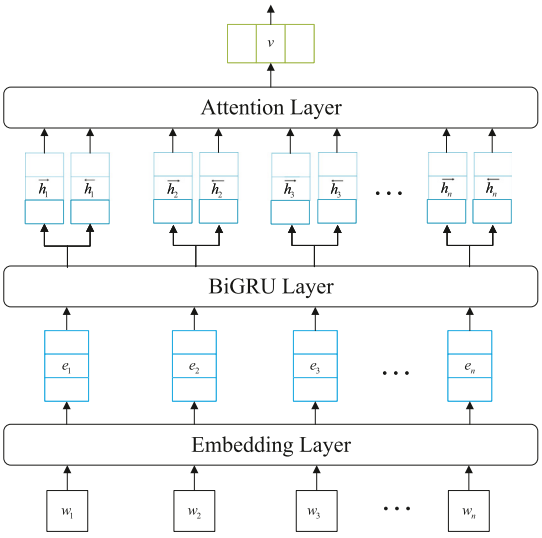


Fig. 1. The text-encoder.

3.1.1. Embedding layer

Given the word sequence of a document: $d \equiv \{w_1, w_2, \ldots, w_n\}$, each word will be mapped to a dense vector representation in the Embedding Layer as follows: $E = [e_1 \cdot e_2 \cdot e_3 \ldots e_n]$, where e_i corresponds to the embedding vector of the ith word to keep word order in the documents, and thus overcome the defects of the BoW model. Generally, the embedding vectors can be trained against a special dataset and shared by all inputs. However, it has been proved that the effect is better by using pre-trained word vectors. Therefore, we adopt the pre-trained word vectors to initialize the embedding layer in the later process.

3.1.2. BiGRU layer

Traditional recurrent neural networks (RNN) suffer from the long-term dependency problem in sequence modeling and the vanishing/exploding gradient problem. Gated RNN introduces recurrent units (e.g., LSTM and GRU), to ease each unit to remember the existence of a specific feature for a long series of steps, and effectively create shortcut paths to avoid vanishing gradients. GRU was first proposed by Chung et al. [25] in 2014. The idea is similar to LSTM, except that its internal structure is simpler and the amount of calculation is relatively small. Formally, GRU is defined as following:

$$\begin{aligned} z_i &= sigmoid\left(W_z \cdot [h_{i-1}, e_i] + b_z\right), \\ r_i &= sigmoid\left(W_r \cdot [h_{i-1}, e_i] + b_r\right), \\ \widetilde{h}_i &= tanh\left(W \cdot [r_i * h_{i-1}, e_i] + b\right), \\ h_i &= (1 - z_i) * h_{i-1} + z_i * \widetilde{h}_i. \end{aligned} \quad (3)$$

where e_i is the current input information, W_* is the weight, b_* is the bias term, z_i and r_i are the output values corresponding to the update gate and the reset gate at time i, $\tilde{h}_i$ and h_i are respectively the candidate output and the output of hidden layer at the moment. The *sigmoid* function is used to control information passing through the gate.

To fully exploit the textual information of documents, we adopt a bi-directional GRU layer to get the backward hidden state sequence $\overleftarrow{H} = \left(\overleftarrow{h}_1, \ldots, \overleftarrow{h}_n\right)$ and the forward hidden state sequence $\overrightarrow{H} = \left(\overrightarrow{h}_1, \ldots, \overrightarrow{h}_n\right)$ given by the input d. Two collections will further concatenate into a new collection as H. In order to express the calculation process conveniently, this process is written as: $H = BiGRU(d)$.

3.1.3. Attention layer

The attention mechanism in deep learning is essentially similar to the selective attention mechanism of human vision, and the core goal is to select more critical information from a large amount of information for the current task goal. In our task, to distinguish the contribution of different positions in a sequence of words, a self-attention mechanism is utilized to assign different weights to each feature vector automatically. When feeding a collection of feature vectors into the attention layer, this collection will be converted to a new l-dimensional vector. As for the collection H, we will get a fused vector,

$$v = \sum_i \alpha_i H_i \tag{4}$$

where α_i is the weight obtained by attention mechanism as follows:

$$\begin{aligned} x_i &= \mathbf{U}_1^T tanh(\mathbf{W}_1 H_i + \mathbf{b}_1) \\ \alpha_i &= \frac{\exp(x_i)}{\sum_{i=1}^{2n} \exp(x_i)} \end{aligned} \tag{5}$$

where $\mathbf{U}_1 \in \mathbf{R}^h$, $\mathbf{W}_1 \in \mathbf{R}^{h \times l}$, $\mathbf{b}_1 \in \mathbf{R}^h$ are initialized model parameters in the attention module. To express the calculation process conveniently, this process is written as: $v = ATT(H)$.

3.2. Neighborhood-aware deep model

3.2.1. Finding K-nearest-neighbors by vector space model

To find the nearest neighbors of target documents, we take the classical *vector space model*. The preprocessed documents are firstly represented as space vectors with word weighting schema of *term frequency–inverse document frequency*(TF–IDF). Then, similarities between the documents are calculated by *cosine* function. Finally, for each target document d, we choose the top-k similar documents to form the set of $\mathcal{N}_d$.

3.2.2. The layered architecture

The layer architecture of our proposed neighborhood-aware deep model for text tagging (NATT for short) is shown in Fig. 2. For the target document d and the neighboring documents $d' \in \mathcal{N}_d$, the same text-encoder is used to generate the feature vectors as: $v = ATT(BiGRU(d))$ or $v' = ATT(BiGRU(d'))$. Further, we use another attention layer to automatically assign weights in the fusion of feature vectors w.r.t $\mathcal{N}_d$ as follows:

$$\begin{aligned} v'' &= \sum_k \alpha_k v'_k, \\ x_k &= \mathbf{U}_2^T tanh(\mathbf{W}_2 v'_k + \mathbf{b}_2), \\ \alpha_k &= \frac{\exp(x_k)}{\sum_{k=1}^{K} \exp(x_k)}, \end{aligned} \tag{6}$$

where $\mathbf{U}_2 \in \mathbf{R}^h$, $\mathbf{W}_2 \in \mathbf{R}^{h \times l}$, $\mathbf{b}_2 \in \mathbf{R}^h$ are initialized model parameters in the attention module. In a single document, the attention mechanism can obtain the contribution of each word in the text, and calculate the weight of each word; for the neighborhood collection $\mathcal{N}_d$, the attention mechanism can also distinguish the contribution of each neighbor. Note that, similar strategies have been applied in collaborative filtering to aggregate the embeddings of items or components [21,22]. However, our work is totally different from these works on network structure, loss function and negative sampling. Additionally, we focus on text classification/tagging, while they focus on recommendation systems.

In the next, tag vectorization is performed by following TagSpace [4], where a tag t get its embedding vector u_t through a lookup table. After the above process, the document d and its neighbor set $\mathcal{N}_d$, the tag t are mapped to the same feature space. Finally, we estimate the matching degree between the candidate tags and the document d with its neighbors $\mathcal{N}_d$, and select the top-ranked tags for tagging d. The matching function is defined as follows:

$$f(d, \mathcal{N}_d, t) = v_d \cdot u_t = (v + v'') \cdot u_t \tag{7}$$

where $\cdot$ indicates the inner production of two vectors. It is worth noting that v and v'' are not equally weighted. They will be co-adapted with u_t in the process of model learning.

3.2.3. Loss function

To train the NATT model, we use the *pairwise hinge loss* as follows:

$$\begin{aligned} \min \sum_d \sum_{t^+ \in \mathcal{T}_d} \max\left\{0, m - f(d, \mathcal{N}_d, t^+) + f(d, \mathcal{N}_d, t^-)\right\}, \\ \text{subject to} \qquad t^- = \arg\max_{t^- \in \mathcal{F}_{dt}} f(d, \mathcal{N}_d, t^-). \end{aligned} \tag{8}$$

where m is a margin parameter, t^+ is a positive label and t^- is a negative label, $\mathcal{T}_d$ is the true label set of d.

For each positive pair $\langle d, t^+ \rangle$, we use the neighborhood-aware sampling strategy to establish the corresponding negative set of tags $\mathcal{F}_{dt}$. As for each positive document–tag pair, we generally sample dozens or even hundreds of negative tags in forming training data. However, many triples may make the function max{., .} be zero, this will lead to invalid updating of model parameters, and wastes a lot of training time. To deal this issue, the following trick is employed in each training epoch: we first calculate $f(d, \mathcal{N}_d, t^-)$ for each tag in $\mathcal{F}_{dt}$, and then select the tag with the largest value, a.k.a, $t^- = \arg\max_{t^- \in \mathcal{F}_{dt}} f(d, \mathcal{N}_d, t^-)$, to calculate loss function and update the model parameters. For example, for each pair $\langle d, t^+ \rangle$, we prepare n negative document–tag pairs in the sampling phase. In the training phase, we first evaluate which of the negative tags best matches the target document and neighboring documents, and then take this tag as the truly negative sample and count it in the loss function. Outwardly, it can reduce the computational overhead by n-1 times. Inwardly, it can avoid the situation, that the current value of function max{., .} is zero, as much as possible. This enables valid updates of model parameters every time, which would accelerate the convergence of model training.

For each layer, model parameters can be upgraded with the back-propagation algorithm which iteratively computes gradients using the chain rule. Many gradient descent optimization algorithms, such as Adam, Adagrad, Adadelta and RMSprop can be used [26]. Moreover, a *mini-batch* gradient descent, which processes a small subset of the training set in each iteration, is necessary to speed up training on the large-scale datasets.

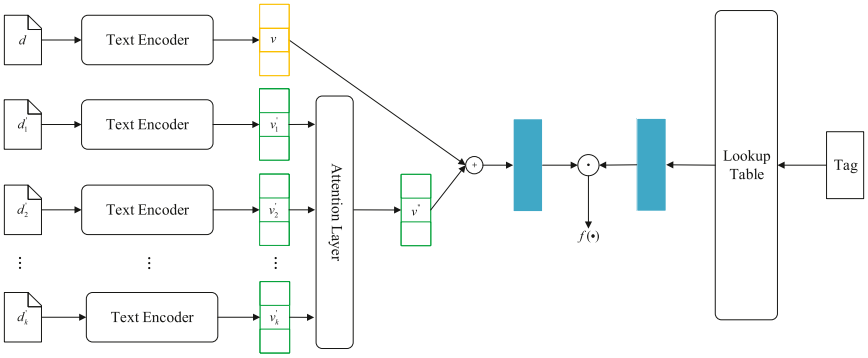


Fig. 2. Framework of the NATT model.

Table 1
Statistics of the datasets.

Dataset	#Documents	#Tags	#Document–tag pairs	#Tags per document	Density
ProgrammableWeb	15,507	508	50,224	3.24	0.637%
CiteULike-t	25,495	1052	120,682	4.73	0.445%
Eurlex-sm	11,547	203	24,777	2.15	1.057%
Reuter	10,377	119	13,122	1.26	1.063%

Algorithm 1 *Neighborhood-aware negative sampling*

Input: d, $\mathcal{N}_d$, $\mathcal{T}_d$;
Output: The list of triples $< d, t, \mathcal{F}_{dt} >$;
  Initialize a list: $L \leftarrow \varnothing$;
  Initialize a collection: $\Omega \leftarrow \mathcal{T}_d$;
  for each document $d' \in \mathcal{N}_d$ **do**
    $\Omega \leftarrow \Omega \cup \mathcal{T}_{d'}$;
  end for
  for each positive tag $t \in \mathcal{T}_d$ **do**
    Randomly sample n tags as the negative samples of t: $\mathcal{F}_{dt}$;
    for each $t' \in \mathcal{F}_{dt}$ **do**
      if $t' \in \Omega$ **then**
        Remove t' from $\mathcal{F}_{dt}$;
        Randomly sample another tag t' until $t' \notin \Omega$;
        Add t' to $\mathcal{F}_{dt}$;
      end if
    end for
    Add triple $< d, t, \mathcal{F}_{dt} >$ into L;
  end for
  return L;

3.2.4. Neighborhood-aware negative sampling

Without considering the neighborhood effect, we can use random sampling to complete the model training, as long as the negative labels do not intersect with the positive one. However, when the neighboring documents are introduced, their positive labels may become the negative labels of the target document, which leads to conflicts. We use a simple strategy to avoid this risk, named as *neighborhood-aware negative sampling*, as shown in Algorithm 1.

The sampling algorithm is carried out in each iteration to optimize the model parameters randomly. Generally, we can sample $n \geq 20$ negative tags for a positive document–tag pair to meet the performance requirements.

4. Experiments

In this section, we conducted extensive experiments aiming to answer the following research questions:

1. RQ1 What is the performance when comparing our NATT model with the state-of-the-art methods?
2. RQ2 How the impact of training epochs is on the prediction performance w.r.t the NATT model?
3. RQ3 How the impact of the number of nearest neighbors k is on the prediction performance?

4.1. Datasets

For our experiments, we choose four datasets to cover different domains of textual documents. Statistics of the datasets are shown in Table 1. Tag frequency's distribution on the four datasets all follows the power-law distribution and thus there is no distortion as shown in Fig. 3.

ProgrammableWeb. The ProgrammableWeb dataset is collected by us from ProgrammableWeb,[1] which is the largest online registry of Web APIs and has documented over 20,000 open Web APIs and thousands of applications. The refined dataset includes 15,507 documents of Web APIs, 508 pre-defined tags and 50,224 document–tag pairs.

CiteULike. The CiteULike-t dataset[2] were collected from CiteULike and Google Scholar by Wang et al. [27]. There are abstracts, titles, and social tags for each academic article. The dataset is very sparse on the document–tag relationship, and contains many duplicate and low-quality tags. For these reasons, we preprocess the dataset by removing low-frequent tags, identifying duplicate tags and finally keep up to 10 labels for each document.

Eurlex-sm. The Eur-lex dataset[3] is a collection of documents about European Union law which provides an excellent opportunity to study text classification techniques [28]. The most important categorization is provided by the EUROVOC descriptors, which form a topic hierarchy regarding different aspects of European law. We use a subset (named Eurlex-sm) which consists of 11,547 documents, 203 tags and 24,777 document–tag pairs.

Reuter. The Reuter-21578 dataset[4] was a collection of Reuters newswire in 1987 where documents were assembled and indexed with categories by personnel from Reuters Ltd. It is widely used in the research field of text classification [29]. The refined dataset

1 https://www.programmableweb.com/.
2 http://www.datatang.com/data/45466.
3 http://www.ke.tu-darmstadt.de/resources/eurlex.
4 https://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection.

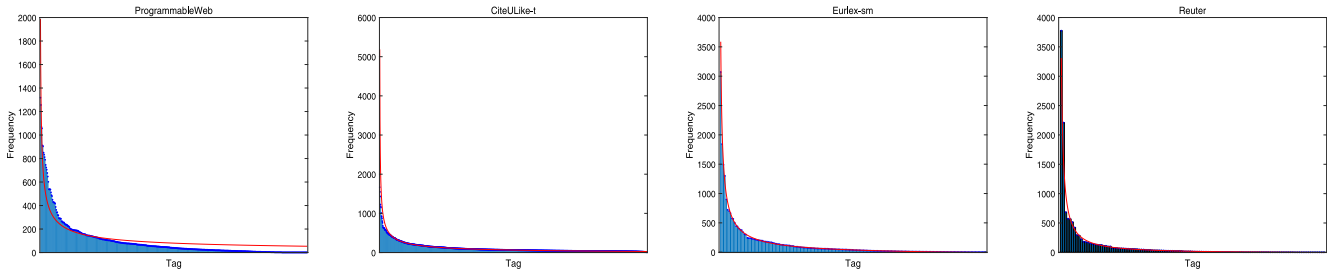


Fig. 3. The distributions of tag frequency on the four datasets.

includes 10,377 documents, 119 tags and 13,122 document–tag pairs.

For the performance evaluation, we utilize traditional offline evaluation. As for each dataset, we randomly divide the set of documents into the training set and the test set using a default ratio: 90%:10%, without considering temporal factors. The documents with tags in the training set are used for training, and their neighboring documents are selected within the same training set. The documents with tags in the test sets are used for test while their neighboring documents are selected within the corresponding training set. Additionally, when performing an evaluation against the test set, cold-start tags those were not emerging in the training set will be abandoned.

For the corpus consists of all documents in each dataset, we then use *TfidfVectorizer* from scikit-learn[5] to extract preliminary features (by removing stop words, discarding the words of that document frequency are greater than 0.5, and restricting the max features by 8000). After the sequence padding, the processed documents will be fed into the embedding layer of the neural network in the form of the word sequence.

4.2. Evaluation metrics

The evaluation metrics for our experiments are *Precision@N*, *Recall@N*, *nDCG@N*, and *HD@N*. The larger the value, the better the classification effect. *P@N* is the ratio of the number of relevant tags in the recommendation list to *N*. The specific formula can be defined as:

$$P@N = \frac{|\{relevant\ tag\} \cap \{top - N\ tag\}|}{N}. \tag{9}$$

R@N is the ratio of the number of relevant tags in the recommendation list to the number of relevant tags in the test set, and defined as:

$$R@N = \frac{|\{relevant\ tag\} \cap \{top - N\ tag\}|}{|\{relevant\ tag\}|}. \tag{10}$$

nDCG@N is defined as:

$$DCG@N = \sum_{i=1}^{N} \frac{2^{rel(i)} - 1}{log_2(i+1)}, \tag{11}$$

$$IDCG@N = \sum_{i=1}^{C} \frac{1}{log_2(i+1)}, \tag{12}$$

$$nDCG@N = \frac{DCG@N}{IDCG@N}, \tag{13}$$

where $rel(i) = 1$ if the ith tag is related to the target document, otherwise $rel(i) = 0$. Using a graded relevance scale of tags, *DCG* measures the usefulness of a tag based on its position in the recommendation list. The gain is accumulated from the top of the recommendation list to the bottom, with the gain of each result discounted at lower ranks. *IDCG@N* is the ideal *DCG* at the position *N*.

HD@N is a measure to evaluate the aggregate diversity of recommendation lists [30]. Given two recommendation lists of tags, it defines as:

$$HD@N = 1 - \frac{overlap@N}{N}. \tag{14}$$

where *overlap@N* indicates the number of tags shared in two recommendation lists. We calculate *HD@N* based on all the pairs of recommendation lists. The higher value of *HD@N* means that the algorithm can use more diverse tags to label documents, rather than focusing on a few popular tags.

4.3. Evaluation methods

The baseline methods used in our experiment are CNN-KIM, XML-CNN, BGRU-Att, BLSTM-2DCNN, TagSpace, ConvMF and ItemKNN, which cover different principles of text tagging methods.

CNN-KIM. It exploits a convolutional neural network trained on top of pre-trained word vectors for classification tasks [8]. In our experiments, the size of the context window is [3, 4, 5] and the number of convolution kernels is 128 and the *softmax* function is used for the task of multi-label classification.

XML-CNN. It is the first attempt at applying deep learning to the extreme multi-label text classification [9], with a tailored convolutional neural network for multi-label classification in particular. In our experiments, the size of the context window is [2, 4, 8] and the number of convolution kernels is 128.

BGRU-Att. [11] proposed a hierarchical attention network for document classification, it attends differentially to more and less important content when constructing the document representation and feeding it into a softmax layer for classification. In our experiments, the attention mechanism is only applied at word-level and thus its structure of the text-encoder is exactly the same as ours.

BLSTM-2DCNN. It explores *2D max-pooling* operation and *2D convolution* over Bidirectional LSTM to sample more meaningful information of the text [10]. The parameter setting is same as the original paper: the units of LSTM is 300, the number of convolutional filters is 100 and the size of the context window is [3, 3], the 2D pooling size is [2, 2].

TagSpace. It also exploits a convolutional neural network trained on top of pre-trained word vectors to learn the representation of documents. In our experiments, the size of the context window is set to 5, the number of convolution kernels is set to 128 and the margin value is set to 0.1, as close as possible to the original setting [4].

[5] https://scikit-learn.org/.

Table 2
Top-5 performance comparisons of selected methods. The best values for our proposed models and the baseline models are marked respectively in **bold** and underline.

Methods	ProgrammableWeb				CiteULike-t			
	P@5	R@5	NDCG@5	HD@5	P@5	R@5	NDCG@5	HD@5
CNN-KIM	0.1861	0.3394	0.5922	0.9107	0.1517	0.2136	0.4518	0.9408
XML-CNN	0.2511	0.4378	0.6311	0.9160	0.2186	0.2923	0.5274	0.9641
BGRU-Att	0.2352	0.4156	0.6085	0.9199	0.2123	0.2806	0.5170	0.9661
BLSTM-2DCNN	0.2702	0.4694	0.6575	0.8805	0.1968	0.2611	0.4836	0.9428
TagSpace	0.2422	0.4164	0.6177	0.9466	0.1650	0.2140	0.4243	0.9600
ConvMF	0.2746	0.4719	0.6602	0.8699	0.1916	0.2462	0.4557	0.8985
ItemKNN	0.3637	0.6258	0.8208	0.9195	0.3274	0.4556	0.7203	0.9623
NATT-0	0.3056	0.5247	0.7464	**0.9431**	0.2684	0.3685	0.6331	**0.9686**
NATT	**0.3726**	**0.6377**	**0.8076**	0.9304	**0.3414**	**0.4706**	**0.7248**	0.9580
Methods	Eurlex-sm				Reuter			
	P@5	R@5	NDCG@5	HD@5	P@5	R@5	NDCG@5	HD@5
CNN-KIM	0.2423	0.6647	0.8712	0.8427	0.1964	0.8915	0.9004	0.7879
XML-CNN	0.3518	0.8717	0.8855	0.8733	0.2008	0.8828	0.8482	0.7048
BGRU-Att	0.3623	0.8607	0.9086	0.8628	0.2056	0.9026	0.8753	0.6949
BLSTM-2DCNN	0.3647	0.8966	0.9171	0.8158	0.2128	0.9273	0.8914	0.7122
TagSpace	0.3461	0.8531	0.8970	0.8898	0.2064	0.9045	0.8781	0.8118
ConvMF	0.3467	0.8569	0.8593	0.7918	0.2183	0.9472	0.8973	0.6015
ItemKNN	0.3664	0.9018	0.9119	0.8077	0.2209	0.9697	0.9285	0.6836
NATT-0	0.3698	0.9079	**0.9402**	**0.8893**	0.2131	0.9260	0.8978	**0.7931**
NATT	**0.3722**	**0.9140**	0.8855	0.7375	**0.2251**	**0.9701**	**0.9424**	0.7649

Table 3
Top-10 performance comparisons of selected methods. The best values for our proposed models and the baseline models are marked respectively in **bold** and underline.

Methods	ProgrammableWeb				CiteULike-t			
	P@10	R@10	NDCG@10	HD@10	P@10	R@10	NDCG@10	HD@10
CNN-KIM	0.1134	0.4018	0.6021	0.8686	0.0950	0.2659	0.4727	0.9097
XML-CNN	0.1596	0.5424	0.6409	0.8795	0.1499	0.3842	0.5419	0.9447
BGRU-Att	0.1515	0.5170	0.6216	0.8720	0.1468	0.3745	0.5316	0.9431
BLSTM-2DCNN	0.1698	0.5731	0.6678	0.8106	0.1351	0.3480	0.5020	0.9062
TagSpace	0.1479	0.4974	0.6273	0.9277	0.1104	0.2790	0.4424	0.9387
ConvMF	0.1786	0.5969	0.6680	0.8125	0.1399	0.3538	0.4841	0.8669
ItemKNN	0.2216	0.7393	0.8151	0.8601	0.2139	0.5655	0.7207	0.9336
NATT-0	0.1835	0.6138	0.7457	**0.9114**	0.1683	0.4440	0.6379	**0.9534**
NATT	**0.2248**	**0.7464**	**0.8021**	0.9034	**0.2216**	**0.5822**	**0.7207**	0.9374
Methods	Eurlex-sm				Reuter			
	P@10	R@10	NDCG@10	HD@10	P@10	R@10	NDCG@10	HD@10
CNN-KIM	0.1299	0.7020	0.8680	0.7868	0.1015	0.9145	0.9062	0.7047
XML-CNN	0.1896	0.9247	0.8857	0.8297	0.1067	0.9196	0.8552	0.6114
BGRU-Att	0.1931	0.9353	0.9066	0.8006	0.1092	0.9382	0.8825	0.6605
BLSTM-2DCNN	0.1933	0.9388	0.9167	0.7303	0.1116	0.9579	0.8985	0.6392
TagSpace	0.1853	0.9007	0.8943	0.8492	0.1082	0.9350	0.8856	0.7876
ConvMF	0.1897	0.9206	0.8592	0.7229	0.1144	0.9730	0.9012	0.5988
ItemKNN	0.1975	0.9548	0.9096	0.7646	0.1136	0.9846	0.9310	0.5621
NATT-0	0.1927	0.9363	**0.9381**	**0.8588**	0.1108	0.9500	0.9028	0.7415
NATT	**0.1984**	**0.9615**	0.8848	0.7136	**0.1161**	**0.9863**	**0.9438**	**0.7437**

ConvMF. It utilizes a component of convolutional neural network same as in CNN-KIM to learn the representation vector of documents [18]. For training the model presented as in Eq. (1), we let $\lambda_u = 10$, $\lambda_v = 0.1$, $\lambda_w = 0$. The size of the context window is [3, 4, 5] and the number of convolution kernels is 128.

ItemKNN. It is a non-parametric method used for classification and benefits from the neighborhood effect. We use $k = 50$, which is the optimal setting on all datasets, to determine the k-nearest-neighbors of documents.

NATT-0. It is a naive version of NATT without considering the neighborhood effect. It is principally similar to TagSpace, except that the CNN used to encode documents are replaced by our text-encoder.

NATT. It is our neighborhood-aware model that considers neighborhood effect both on document modeling and negative sampling.

For a fair comparison, we also use the same setting of word embedding for all the neural models except ItemKNN, namely, the embedding layer is initialized using *pre-trained* word vectors of GloVe[6] with a dimensional size of 300. Besides, the embedding size of tags is set to 300 for ConvMF, TagSpace and NATT. As for the gradient descent algorithm, *Adam* is choosed as the optimizer. In addition, all the neural models are implemented in Tensorflow and tested on a computer configured with Intel Quad-Core i7 processor, 32GB memory and Ubuntu 16.04 with GPU support of Nvidia GTX 1080Ti.

4.4. Experimental results

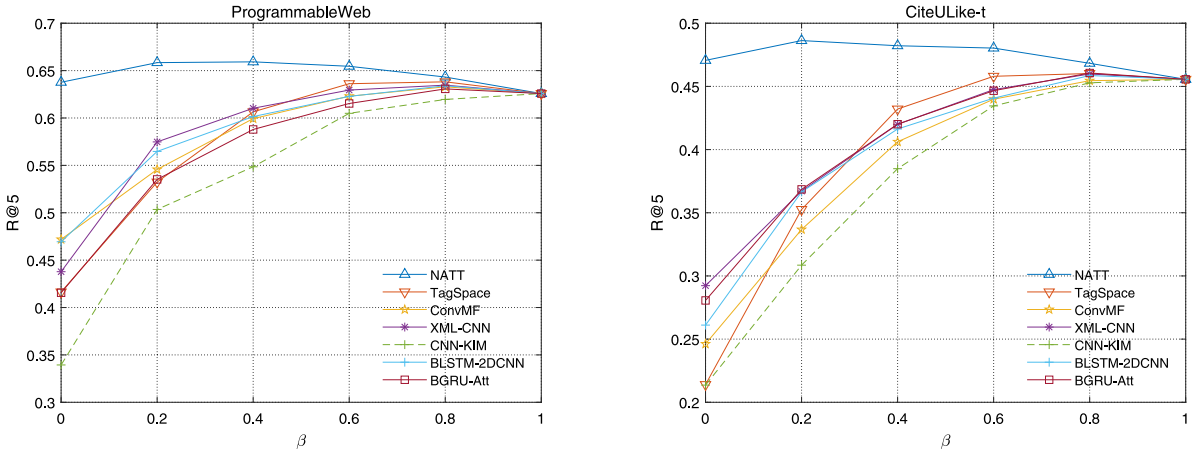
4.4.1. Performance comparison (RQ1)

To answer RQ1, we conduct intensive experiments against the four datasets. The results are presented in Tables 2 and 3.

[6] https://nlp.stanford.edu/projects/glove/.

Table 4
Case study of tag predictions on the Reuter dataset where tags get hit are marked in bold font.

Target text	Methods	Top 5 tags
Israel will tender overnight for 33,000 long tons of U.S. sorghum and/or 22,000 long tons of U.S. corn for April shipment, private export sources said.	BGRU-Att	oilseed, soybean, **grain**, **corn**, wheat
	NATT-0	**grain**, **corn**, barley, crude, wheat
	NATT	**sorghum**, **grain**, wheat, ship, soybean
	Groundtruth	**sorghum**, **grain**, **corn**
The Commodity Credit Corporation (CCC) announced 1.5 mln dlrs in credit guarantees previously earmarked to cover sales of dry edible beans to Honduras have been switched to cover sales of white corn, the U.S. Agriculture Department said. ...	BGRU-Att	oilseed, soybean, **grain**, **corn**, rice
	NATT-0	**corn**, **grain**, wheat, rice, oilseed
	NATT	earn, **grain**, **corn**, acq, cotton
	Groundtruth	**grain**, **corn**
Thai natural rubber exports rose to 763,331 tonnes in 1986 from 689,964 a year earlier, the private Board of Trade said. Japan, the biggest buyer, imported 384,622 tonnes of Thairubber in 1986, up from 348,855 the previous year, it said. ...	BGRU-Att	trade, oilseed, soybean, grain, sugar
	NATT-0	**rubber**, trade, coconut, tea, coffee
	NATT	**rubber**, grain, corn, rapeseed, wheat
	Groundtruth	**rubber**



(a) (b)

Fig. 4. Top-n performance on merging ItemKNN.

The overall performance of CNN-KIM is lower since it is a naive model. TagSpace performs significantly better than CNN-KIM as it is based on collaborative representation learning. As for XML-CNN, it exploits a specially tailored convolutional neural network for multilabel classification, and thus outperforms CNN-KIM. BGRU-Att exploits bidirectional RNN and self-attention mechanism to extract features from the text, it outperforms CNN-KIM and achieves comparable results to XML-CNN. Since BLSTM-2DCNN utilizes 2D convolution and pooling over bidirectional LSTM to enhance the text encoding, it outperforms XML-CNN, BGRU-Att and TagSpace. With respect to ConvMF, it fully exploits the text-tag matrix and textual information to learn the representation of textual document and tags, and thus achieves a comparable performance with XML-CNN, BGRU-Att and BLSTM-2DCNN in most cases.

For NATT-0, although it is significantly better than most of the baselines, it cannot compete with ItemKNN due to the lack of considering neighborhood effect. By contrast, NATT achieves much better results leveraging on the neighborhood effect. It beats CNN-KIM, XML-CNN, BGRU-Att, BLSTM-2DCNN, TagSpace and ConvMF on all the datasets. In terms of P@5, R@5 and NDCG@5, it achieves at least 35.7%, 35.1%, 22.3% gains in the ProgrammableWeb dataset, at least 56.1%, 61.0%, 36.6% gains in the CiteULike-t dataset, at least 2.0%, 2.0%, −3.5% gains in the Eurlex-sm dataset, and at least 3.1%, 2.4%, 5.0% gains in the Reuter dataset. In terms of P@10, R@10 and NDCG@10, the trend is basically the same. We also notice that when the dataset is more sparse and per document has more tags, the better NATT performs, comparing the datasets of ProgrammabelWeb and CiteULike-t with the datasets of Eurlex-sm and Reuter.

In HD@{5,10}, NATT-0 and TagSpace are the two best players. NATT is also competitive, but the performance may be decreased, due to the fact that popular tags tend to be promoted by neighbors in comparison with NATT-0. The behind reason is perhaps that tags are embedded as distributed representations and tightly coupled with text encoder in these models (while ConvMF is loosely coupled), which helps capture semantic divergences among tags and reduce tag ambiguities. In contrast, multi-label classification models based on activation functions (e.g., *softmax*), do not have this advantage, as highly-frequent or popular tags will be assigned to with high probability when candidate tags are strongly relevant to each other.

To illustrate this point, we further carry out a case study on the Reuter dataset which has shorter texts and less tags. The results are presented in Table 4. In particular, we compare BGRU-Att with NATT/NATT-0 as they are built on the same text-encoder. The difference is that BGRU-Att makes classification with a softmax layer. For the first two cases in Table 4, the documents are all reports on agricultural products trade. Tags proposed by the selected model are strongly relevant to each other and semantic divergences among them is small. BGRU-Att and NATT/NATT-0 achieve the same accuracies on top-5 recommendations. However, BGRU-Att prefers popular tags. Both *oilseed* and *soybean*, which occur hundreds of times in the dataset, rank in the first two places. For the third case, BGRU-Att shows the same symptoms and fails in predicting tags. On the contrary, NATT/NATT-0 gives accurate and diverse answers, where *rubber* only occurs three times in the dataset. Totally, NATT/NATT-0 can suggest novel tags rather than popular tags, thus improving the performance of long-tail recommendation.

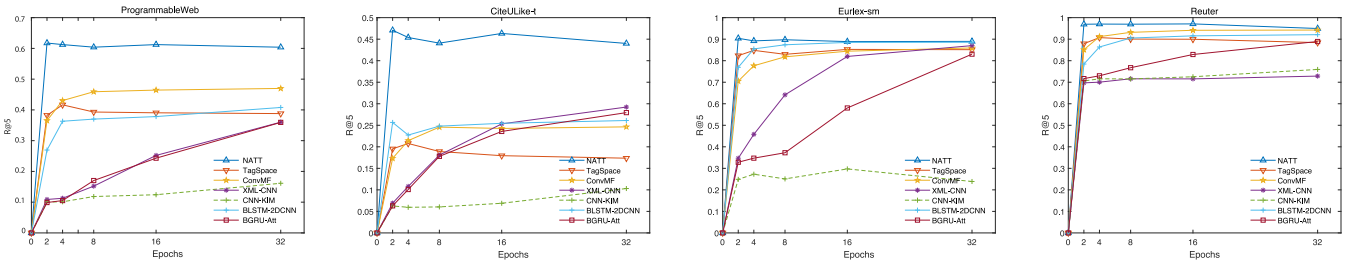


Fig. 5. Top-n performance on training epochs.

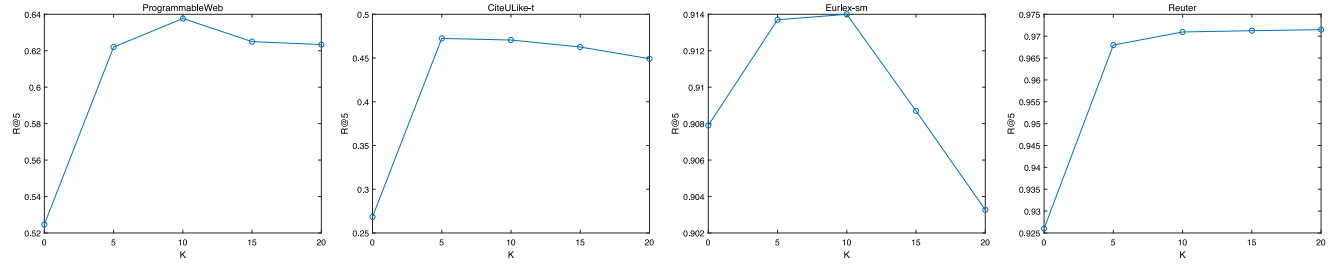


Fig. 6. Impact of the number of nearest neighbors w.r.t NATT.

It is worth emphasizing that ItemKNN performs outstandingly in most cases, which is consistent with the recent observation in the research of top-n item recommendation [13], where authors fully compare the prediction performance of ItemKNN with those of the state-of-the-art deep recommendation models. It further confirms the advantage of neighborhood effect in the task of tag recommendation. However, this result does not hinder the significance of our work. On the one hand, NATT is competitive in comparison with ItemKNN. NATT works better than ItemKNN on the datasets of ProgrammableWeb and CiteULike-t, and achieves close results on the datasets of Eurlex-sm and Reuter. On the other hand, NATT can complement with ItemKNN. To illustrate this point, we further blend ItemKNN with the remaining models. Specifically, the results of different prediction functions are normalized first as $r'(d,t)$, then weighted with the normalized prediction score of ItemKNN, $r(d,t)$, to generate a hybrid prediction rule as follows:

$$r(d,t) = \beta * r(d,t) + (1-\beta) * r'(d,t) \tag{15}$$

where β is the weighting factor to balance the effect of two methods. In this way, the performance benefits of the fusion method can be observed.

The experimental results are shown in Fig. 4 where we changed β from 0 to 1 with a step of 0.2. We found that CNN-KIM, XML-CNN, TagSpace, BGRU-Att, BLSTM-2DCNN and ConvMF get little benefits when combined with ItemKNN. For the NATT model, this combination can bring at least 2% revenue on the metric of R@5 when β is taken a value of around 0.2. Considering that the average number of tags per document is no more than 5 on the four datasets, R@5 is more suitable to measure the overall performance of recommendations. This implies that NATT can complement with ItemKNN, while other methods are suppressed by ItemKNN.

4.4.2. Performance on training epochs (RQ2)

In this section, we observe the performance changes of different models in the iterative training process and estimate the training efficiency by the way. The results are shown in Fig. 5 where we only consider the metric of R@5 as it is more meaningful in performance comparison.

We find that the proposed NATT model usually only needs 2–4 iterations to get the best results compared with other models. ConvMF, CNN-KIM, XML-CNN, BGRU-Att and BLSTM-2DCNN require multiple times of iterations to get the best results. In particular, compared with TagSpace, the NATT model still significantly reduces the number of training iterations. There are two reasons for this. One lies in using neighborhood information to enhance the feature representation of target documents which is equivalent to a data-enhanced training process. It is conducive to the rapid optimization of model parameters. Second, for each triple $\langle d, t, \mathcal{F}_{dt} \rangle$, we only choose the most relevant label in $\mathcal{F}_{dt}$ to update the model parameters.

4.4.3. Impact of the number of nearest neighbors (RQ3)

In this section, we observe the influence of the number of nearest neighbors on the performance of NATT. Fig. 6 shows the results of the four datasets. It can be seen that when the number of nearest neighbors is $k = 10$, the best result can be obtained. Furthermore, increasing the number of neighbors will damage performance. Compared with the optimal configuration of $k = 50$ in ItemKNN, our method significantly reduces the demand for the number of neighbors. This helps to reduce the computational overhead of NATT, considering that the component of text encoding is time-consuming.

5. Conclusions and future works

We have proposed NATT which exploits the neighborhood effect of documents on both text encoding and negative sampling for text tagging. Experimental results reflect that NATT is effective in the task of top-n recommendation of tags. Also, NATT shows its merits regarding computational efficiency, as it requires less training epochs and a smaller number of nearest neighbors whenever achieving the best results.

Unlike the existing works which focus on developing deep neural networks for text encoding, we concentrate on establishing a neighborhood-aware framework to promote the effectiveness of tag predictions. If a stronger component can improve the

effect of text encoding, for example, the BERT model [31] and the capsule networks [32], more gains in performance can be expected. Also, it would be interesting to improve the strategy of negative sampling and the effectiveness of tag embedding by considering co-occurrence or other relationships of tags [33]. We leave these works for the future.

CRediT authorship contribution statement

Shaowei Qin: Conceptualization, Methodology, Software, Data curation, Investigation, Formal analysis, Writing - original draft. **Hao Wu:** Supervision, Resources, Conceptualization, Methodology, Investigation, Project administration, Funding acquisition, Writing - original draft. **Rencan Nie:** Methodology, Writing - review & editing. **Jun He:** Resources, Writing - review & editing.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61962061, 61562090, U1802271), partially supported by the Yunnan Provincial Foundation for Leaders of Disciplines in Science and Technology, Top Young Talents of "Ten Thousand Plan" in Yunnan Province, China, the Program for Excellent Young Talents of Yunnan University, China, the Project of Innovative Research Team of Yunnan Province, China (2018HC019).

References

[1] M. Gupta, R. Li, Z. Yin, J. Han, Survey on social tagging techniques, ACM Sigkdd Explor. Newsl. 12 (1) (2010) 58–72.
[2] F. Herrera, F. Charte, A.J. Rivera, M.J. Del Jesus, Multilabel classification, in: Multilabel Classification, Springer, 2016, pp. 17–31.
[3] P. Cremonesi, Y. Koren, R. Turrin, Performance of recommender algorithms on top-n recommendation tasks, in: Proceedings of the Fourth ACM Conference on Recommender Systems, ACM, 2010, pp. 39–46.
[4] J. Weston, S. Chopra, K. Adams, # tagspace: Semantic embeddings from hashtags, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1822–1827.
[5] S. Zhang, L. Yao, A. Sun, Y. Tay, Deep learning based recommender system: A survey and new perspectives, ACM Comput. Surv. 52 (1) (2019) 5:1–5:38.
[6] H. Wang, N. Wang, D.-Y. Yeung, Collaborative deep learning for recommender systems, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD'15), ACM, 2015, pp. 1235–1244.
[7] R. Johnson, T. Zhang, Effective use of word order for text Categorization with convolutional neural networks, in: NAACL HLT 2015, the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015, 2015, pp. 103–112.
[8] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, a Meeting of SIGDAT, a Special Interest Group of the ACL, 2014, pp. 1746–1751.
[9] J. Liu, W.-C. Chang, Y. Wu, Y. Yang, Deep learning for extreme multi-label text classification, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2017, pp. 115–124.
[10] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, B. Xu, Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling, in: COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan, 2016, pp. 3485–3495.
[11] Z. Yang, D. Yang, C. Dyer, X. He, A.J. Smola, E.H. Hovy, Hierarchical attention networks for document classification, in: NAACL HLT 2016, the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016, 2016, pp. 1480–1489.
[12] E. Grave, T. Mikolov, A. Joulin, P. Bojanowski, Bag of tricks for efficient text classification, in: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers, 2017, pp. 427–431.
[13] M.F. Dacrema, P. Cremonesi, D. Jannach, Are we really making much progress? a worrying analysis of recent neural recommendation approaches, in: Proceedings of the 13th ACM Conference on Recommender Systems, ACM, 2019, pp. 101–109.
[14] N. Bhatia, Vandana, Survey of nearest neighbor techniques, 2010, CoRR abs/1007.0085, URL: http://arxiv.org/abs/1007.0085.
[15] S. Tan, Neighbor-weighted k-nearest neighbor for unbalanced text corpus, Expert Syst. Appl. 28 (4) (2005) 667–671.
[16] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, in: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.
[17] G. Wang, C. Li, W. Wang, Y. Zhang, D. Shen, X. Zhang, R. Henao, L. Carin, Joint embedding of words and labels for text classification, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers, 2018, pp. 2321–2331.
[18] D. Kim, C. Park, J. Oh, S. Lee, H. Yu, Convolutional matrix factorization for document context-aware recommendation, in: Proceedings of the 10th ACM Conference on Recommender Systems (RecSys'16), ACM, New York, NY, USA, 2016, pp. 233–240.
[19] H. Wu, Z. Zhang, K. Yue, B. Zhang, J. He, L. Sun, Dual-regularized matrix factorization with deep neural networks for recommender systems, Knowl.-Based Syst. 145 (2018) 46–58.
[20] B.M. Sarwar, G. Karypis, J.A. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001, 2001, pp. 285–295.
[21] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, T. Chua, Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017, 2017, pp. 335–344.
[22] L. Chen, Y. Liu, X. He, L. Gao, Z. Zheng, Matching user with item set: Collaborative bundle recommendation with deep attention network, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019, 2019, pp. 2095–2101.
[23] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, B. Xu, Attention-based bidirectional long short-term memory networks for relation classification, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers, 2016.
[24] D. Dong, X. Zheng, R. Zhang, Y. Wang, Recurrent collaborative filtering for unifying general and sequential recommender, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden, 2018, pp. 3350–3356.
[25] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014, arXiv preprint arXiv:1412.3555.
[26] S. Ruder, An overview of gradient descent optimization algorithms, 2016, CoRR abs/1609.04747, URL: http://arxiv.org/abs/1609.04747.
[27] H. Wang, B. Chen, W. Li, Collaborative topic regression with social regularization for tag recommendation, in: IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013, 2013, pp. 2719–2725.
[28] E. Loza Mencía, J. Fürnkranz, An evaluation of efficient multilabel classification algorithms for large-scale problems in the legal domain, in: Montemagni, Simonetta and Tiscornia, Daniela and Francesconi, Enrico and Peters, Wim (eds.), Proceedings of the LREC 2008 Workshop on Semantic Processing of Legal Texts, Marrakech, Morocco, 2008, pp. 23–32.
[29] C. Apté, F. Damerau, S.M. Weiss, Automated learning of decision rules for text Categorization, ACM Trans. Inf. Syst. 12 (3) (1994) 233–251.
[30] X. Wang, H. Wu, C. Hsu, Mashup-oriented API recommendation via random walk on knowledge graph, IEEE Access 7 (2019) 7651–7662.
[31] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), ACL, Minneapolis, Minnesota, 2019, pp. 4171–4186.
[32] R. Aly, S. Remus, C. Biemann, Hierarchical multi-label classification of text with capsule networks, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, 2019, pp. 323–330.
[33] H. Wu, Q. Zhou, R. Nie, J. Cao, Effective metric learning with co-occurrence embedding for collaborative recommendations, Neural Netw. 124 (2020) 308–318.