Joint Text Embedding for Personalized Content-based Recommendation

Ting Chen[*]
University of California, Los Angeles
Los Angeles, CA
tingchen@cs.ucla.edu

Liangjie Hong
Etsy Inc.
New York City, NY
lhong@etsy.com

Yue Shi[†]
Yahoo! Research
Sunnyvale, CA
yueshi@acm.org

Yizhou Sun
University of California, Los Angeles
Los Angeles, CA
yzsun@cs.ucla.edu

ABSTRACT

Learning a good representation of text is key to many recommendation applications. Examples include news recommendation where texts to be recommended are constantly published everyday. However, most existing recommendation techniques, such as matrix factorization based methods, mainly rely on interaction histories to learn representations of items. While latent factors of items can be learned effectively from user interaction data, in many cases, such data is not available, especially for newly emerged items.

In this work, we aim to address the problem of personalized recommendation for completely new items with text information available. We cast the problem as a personalized text ranking problem and propose a general framework that combines text embedding with personalized recommendation. Users and textual content are embedded into latent feature space. The text embedding function can be learned end-to-end by predicting user interactions with items. To alleviate sparsity in interaction data, and leverage large amount of text data with little or no user interactions, we further propose a joint text embedding model that incorporates unsupervised text embedding with a combination module. Experimental results show that our model can significantly improve the effectiveness of recommendation systems on real-world datasets.

1 INTRODUCTION

Personalized recommendation has gained a lot of attention during the past few years [15, 25, 30]. Many models and algorithms have been proposed for personalized recommendation, among which, collaborative filtering techniques such as matrix factorization [14, 24] are shown to be most effective. For these approaches, historical behavior data is critical for learning latent factors of both users and items. However, in many scenarios, behavior data is not available or very sparse, which motivates us to incorporate content/text information for recommendation. In this work, we study the problem of content-based recommendation for *completely new* items/texts, where historical user behavior data is not available for new items at the time of recommendation. However, when it comes to text

article recommendations, it is not straightforward to incorporate text content into existing collaborative filtering models.

In order to understand content of new items/texts for better recommendation, a good representation based on textual information is essential. This issue is challenging and has not been satisfyingly solved yet. On one hand, traditional content-based [21] recommendation methods are usually based on simple text processing methods such as cosine similarity or logistic regression where both text and users are represented as bag-of-words. The limitations of such representation include the inability to encode similarity between words, as well as losing word order information [10, 19]. On the other hand, for collaborative filtering methods, although some of which has been extended to incorporate auxiliary information, text feature extraction functions are usually simple, and cannot leverage recent proposed representation learning techniques for text [5, 22, 27].

We address these issues with an approach that marries text embedding to personalized recommendation. In our proposed model, users and texts are simultaneously embedded into latent space where preferences can be depicted by simple dot product. While each user is directly associated with an embedding vector, text embedding requires an embedding function that maps a text sequence into a vector. Both user embedding and text embedding function can be trained end-to-end based on user-item interactions directly. With sophisticated neural networks (e.g., Convolutional Neural Networks) as text embedding function, high-level textual features can be better captured.

While end-to-end training of the embedding function delivers focused supervision for learning the task related representations. Interaction data is usually sparse, and there are still large amount of unlabeled data/corpora. Hence, we further propose a joint text embedding model to leverage unsupervised text embeddings that are pre-trained on large-scale unlabeled copora. To effectively fuse both types of information, a novel combination module is constructed and incorporated into the unified framework. Experimental results on two real-world data sets demonstrate the effectiveness of the proposed joint text embedding framework.

---

[*]Work done while the first author was an intern at Yahoo! Research.
[†]Now at Facebook.

2 PRELIMINARIES AND RELATED WORK

2.1 Problem Definition

We use $X = (x_1, \cdots, x_N)$ to denote the set of texts, the i-th text is represented by a sequence of words, i.e. $x_i = (w_1, \cdots, w_t)$. A matrix C is used to denote the historical interactions between users and texts, where C_{ij} indicates interaction between a user i and a text article j, such as click-or-not, like-or-not. [1]

Given text information X and historical interaction data C, our goal is to learn a model which can rank completely new texts for an existing user i based on this user's interests and the text content.

2.2 Personalized Recommendation

Existing methods of personalized recommendation algorithms can be roughly categorized into there categories: (1) collaborative filtering methods, (2) content-based methods, and (3) hybrid methods.

Matrix factorization (MF) techniques [14, 24] is one of the most effective collaborative filtering (CF) methods. In MF, each user or item is associated with latent factor vectors u or v and the score between a pair of user and item is computed by their dot product, i.e. $s_{ij} = u_i^T v_j$. Since each item j is associated with latent factors v_j, an new item cannot be handled properly as the training of v_j depends on its interaction with users.

Content based methods [21] usually build model for user and content based on term weighting schemes like TF-IDF. And cosine similarity or logistic regression can be used to match between a pair of user and item. It is difficult to work with such representations to encode similarities between words, as well as word orders.

Hybrid methods can improve so-called "cold-start" issue by incorporating side information [5, 22, 27], or item content information [8, 30, 31]. However, most of these methods cannot deal with completely new items.

There are some work aiming at leveraging neural networks for better text recommendations, such as Collaborative Deep Learning [31], and others [1]. Compared to their work, 1) we treat the problem as a ranking problem instead of a rating prediction problem, thus pairwise loss functions are adopted; 2) our model provide a more general framework, enabling various text embedding functions, thus subsumes [1] as a special case; 3) our model incorporates unsupervised text embedding from large-scale unlabeled corpora.

2.3 Text Embedding

Recent advances in deep learning have demonstrated the importance of learning good representations for text and other types of data [2, 4, 12, 16, 18, 19]. Text embedding techniques aim at mapping text into vector representation that can be utilized for future predictive tasks. Such models have been proposed for addressing text classification/categorization problem [10, 12, 16]. Our task resembles a personalized text classification/ranking problem, in the sense that we try to classify/rank an article according to its interestingness w.r.t. a given user. Also, we utilize user behavior instead of labels of text as a supervised signal.

---

[1]We consider C as implicit feedback in this work, which means only positive interactions are provided, and non-interactions are treated as negative feedback implicitly.
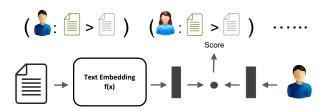

Figure 1: A supervised text embedding framework. Predicted score for a user-text interaction is fit into pairwise ranking objective shown on the top.

3 THE PROPOSED MODEL

In this section, we first introduce the supervised text embedding framework, which is trained in an end-to-end fashion for predicting user-item interactions. Then we propose a joint text embedding model by incorporating unsupervised text embedding with a combination function.

3.1 Supervised Text Embedding Framework

To simultaneously capture interests of users and semantics of texts, we embed both user and text into a common latent feature space, where dot product can be used to quantify their proximity.

Each user i is directly associated with an embedding vector u_i, which represents user's interests. For a text sequence x_j for the j-th item, it is mapped into a fixed-sized vector by an embedding function $f(x_j) \in \mathbb{R}^k$. The proximity score s_{ij} between the user and item pair (i, j) is computed by the dot product between their embeddings, as follows:

$$s_{ij} = u_i^T f(x_j)$$

Text embedding function $f(x)$. In our framework, the text embedding function is very flexible. It can be specified by *any* differentiable function that maps a text sequence into a fix-sized embedding vector. Many neural network structures can be applied, such as Convolutional Neural Networks, Recurrent Neural Networks, and etc. Here we introduce two such functions, *MoV* and *CNN*, while other extensions are straightforward.

Mean of Vectors (MoV). To represent a text sequence x of length T, we first embed each word in the text with an embedding vector w [18, 19], and then use the average of word embeddings to form the text/article embedding as follows:
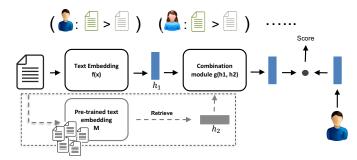
$$h = \frac{1}{t} \sum_{i=1}^{T} w_{x_i}$$

To better extract non-linear interactions among words, a densely-connected layer with non-linear activation can be applied. A single layer of such transformation is given by:
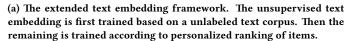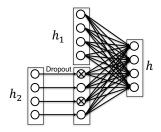
$$h' = \text{Relu}(Wh + b)$$

where $\text{Relu}(\cdot) = \max(0, \cdot)$ is Rectified Linear Unit.

Convolutional Neural Networks (CNN). Although *MoW* model is simple and relatively efficient, since text sequence is treated as a bag of words, orderings among words are ignored. As demonstrated in [10], ordering information of words can be helpful. To

(a) The extended text embedding framework. The unsupervised text embedding is first trained based on a unlabeled text corpus. Then the remaining is trained according to personalized ranking of items.

(b) Combination function $g(h_1, h_2)$ explicitly combines two text embedding vectors.

Figure 2: The joint text embeding framework.

address the issue, Convolutional Neural Networks is adopted for text embedding.

In CNN, instead of averaging over all word embeddings, it maintains several filters of given size(s). Each filter will slide over the whole text sequence. Additionally, at each position, an activation is computed by a dot product between the filter and local embedding vectors. To be more specific, we use $D = w_{x_1} \oplus w_{x_2} \oplus \cdots \oplus w_{x_T} \in \mathbb{R}^{T \times k}$ to denote the concatenation of word vectors for the text. To apply convolution on text sequence D, we compute the j-th entry from applying i-th filter according to:

$$c_j^i = \text{Relu}(W^j \cdot D_{i:i+s-1} + b^j).$$

Here $W^j \in \mathbb{R}^{s \times k}$ is the j-th filter of size s, and b^j is the bias term. The output c of convolution layer can be downsized by pooling operator, such as taking max over all temporal dimensions of c, so a fixed sized vector h can be produced. Due to the page limit, we refer the reader to [12] for more clear detailed descriptions.

Objective Function and Training. To learn the user embedding and text embedding function, the output scores for each pair of user and item are used to predict their interactions. For a given user, we want to rank his/her interested articles higher than those he/she is not. So for each user i, a pair of a positive item p and a negative item n are both sampled, and similar to [23], the score difference between positive and negative items is maximized, leading to a pairwise ranking loss function as follows:

$$\mathcal{L} = \mathbb{E}_{(i,p,n) \sim \mathcal{D}} \left[-\log \sigma(s_{ip} - s_{in}) \right] \tag{1}$$

where p is a positive item for user i, and n is a negative item for user i. Each triplet (i, p, n) is drawn from some predefined data distribution $\mathcal{D}$. And σ is sigmoid function. The objective can be optimized by Stochastic Gradient Descent (SGD). In order to get triplets $\{(i, p, n)\}$ for training, positive interactions $\{(i, p)\}$ are first sampled, and then negative items are sampled according to some predefined distribution (e.g. item frequency).

The framework is demonstrated in Figure 1. We name the above proposed framework TER, short for Text Embedding for content-based Rcommendation.

3.2 Incorporating Unsupervised Text Embedding

There are two challenges faced by the supervised text embedding framework proposed above: 1) user-item interaction data may be sparse, and 2) there are many texts with little to none user interactions. These issues can lead to over-fitting. To alleviate sparsity in interaction data and leverage a large amount of text data with little to none user interactions, we propose to incorporate unsupervised text embedding with a new combination function. The overall framework with joint text embedding is summarized in Figure 2a.

Different from the supervised model, a pre-trained text embedding module is added, so each text is first mapped into two embedding vectors: h_1 from text embedding function $f(x)$ and h_2 from pre-trained embedding matrix M. Then to generate a cohesive text embedding vector for the item, we propose a combination function $g(h_1, h_2)$ to explicitly combine h_1 and h_2. Below we introduce these two additional components in detail.

Unsupervised Text Embedding Matrix M. Unlike supervised text embedding, which requires user interactions for training mapping function $f(x)$. The unsupervised text embedding can be pre-trained with only text articles themselves, requiring no additional labels. To leverage a large-scale text corpus, we adopt Paragraph Vector [16] in our framework.

Given a set of text articles, Paragraph Vector associates each word i with a word embedding vector w_i and each document d with a document embedding vector v_d. To learn both types of embedding vectors simultaneously, a prediction task is formed: for each word occurrence, we firstly hide the word, and then model is asked to predict the exact word given neighboring word embeddings and the document embedding. The probability of the word is given as follows:

$$P(token = i) = \frac{\exp(v_d^T w_i)}{\exp(\sum_j v_d^T w_j)}$$

As introduced in [16], the model is trained by maximum likelihood with negative sampling.

After training Paragraph Vector on the whole corpus, which includes text articles that have no related user interaction associated. We obtain a pre-trained text embedding module with embedding

matrix M, where each row M_i is an unsupervised text embedding vector for the i-th text article.

Combination Function $g(h_1, h_2)$. To combine both text embedding vectors, i.e. h_1 from text embedding function $f(x)$, and h_2 from pre-trained embedding matrix M, we introduce a combination function $g(h_1, h_2) \in \mathbb{R}^k$ where k is the user-defined output dimension. Since the relation between two text embedding vectors h_1 and h_2 can be complicated and non-linear, in order to combine them effectively, we specify the combination function $g(\cdot)$ with a small neural network: Firstly a concatenation of the two vectors are formed, i.e. $h = [h_1, h_2]$, and then it is further transformed by a densely-connected layer with non-linear activations, i.e. $h' = \text{Relu}(Wh + b)$.

Although unsupervised text embeddings can provide useful text features [16], they might not be directly relevant to the task. So to control the degree of trust for unsupervised text embeddings, we introduce dropout [28] into unsupervised text vectors, i.e. h_2, which randomly select entries and set them to zero. On one hand, when setting the dropout to zero, the whole embedding vector is utilized; on the other hand, when setting the dropout to one, the whole text vector is set to zero, hence it is equivalent to use none of pre-trained embeddings. When the dropout rate is between zero and one, it can be seen as a trade-off for the unsupervised module. Figure 2b illustrates the combination module.

Training of the Joint Model. The training procedure is separated into two stages. At the first stage, a unsupervised text embedding matrix M is trained using unlabeled texts. At the second stage, similar to the supervised framework, the training objective is also pairwise ranking objective in Eq. 1. The parameters in the second stage involve both user embeddings and parameters in $f(x)$ and $g(h_1, h_2)$. Finally we name the extended model TER+.

4 EXPERIMENTS

In this section, we present our empirical studies on two real-world text recommendation data sets.

4.1 Data Collections

Two real-world data sets are used. The first data set CiteULike, containing user-bookmarking-article behavior data from CiteU-Like.org, was provided in [30]. It contains 5,551 users, 16,980 items, and 204,986 interactions. The second data set is Yahoo! News Feed[2]. We randomly sampled 10,000 users (with at least 10 click behaviors) and their clicked news to form the data set, which contains 58,579 items, and 515,503 interactions. Since CiteULike and News data sets have both title and abstract/summary, for each data set, we create following two data sets: one contains only title information (i.e. short text), and the other contains both title and summary/abstract (i.e. long text). The average lengths of short text in CiteULike and News are 9 and 11 respectively, and that of long text are 194 and 89 respectively.

To ensure items at the test time are *completely new*, we first select a portion (20%) of items to form the pool of test items. All user interactions with those test items are held-out during training, only the remaining user-item interactions are used as training data. For unsupervised text embedding pre-training, we also include many

Table 1: Data statistics for user, items and their interactions.

	# of user	# of item	# of interaction
Citeulike	5,551	16,980	204,986
News	10,000	58,579	515,503

Table 2: Data statistics for text content.

	voc. size	max	min	mean	median
Citeulike (title)	4,777	15	2	9	9
Citeulike (title&abs.)	23,011	300	22	194	186
News (title)	16,589	20	1	11	11
News (title&sum.)	41,537	200	2	89	90

texts that have no user interaction data. More specifically, for CiteULike data set, additional 339,150 papers from DBLP (a superset of CiteULike) are included; and for news data set, additional 3,935,228 news articles are also included.

The detailed data set statistics are shown in Table 1 and 2.

4.2 Comparing Methods and Settings

We compare following methods in experiments:

- Cosine similarity matching [21], which is based on similarities of TF-IDFs between candidate and user's historical items.
- Regularized multi-task logistic regression [7], which can be seen as one-layer linear text model.
- CDL (Collaborative Deep Learning) [31], which simultaneously trains auto-encoder for encoding text content, and matrix factorization for encoding user behavior.
- Content Pre-trained, which first pre-trains text embeddings by Paragraph Vector, and then used as fixed item features for matrix factorization.
- TER. This is our proposed supervised framework. Note that two variants of text embedding function $f(x)$ are compared: MoV and CNN.
- TER+. This is the joint text embedding framework. Both text embedding functions, MoV and CNN, are compared.

Parameter Settings: For CDL, both TER and TER+, we set the dimensions of both user embedding and final text embedding vector to 50 for fair comparisons. For CNN, we use 50 filters with filter size of 3. Regularization is added using both weight decay on user embedding and dropout on item embedding. We use Adam [13] with learning rate of 0.001. For both baselines and our model, we tune the parameters with grid search.

Evaluation Metrics: We adopt MAP (Mean Average Precision) and average AUC for evaluation. First, for each interaction between a user and a test item in the test set, we sample 10 negative samples from a test item-pool to form the candidate set. Then, AP and AUC are computed based on the rankings given by the model and the final MAP and average AUC are averaged over all users.

4.3 Performance Comparison

Table 3 shows MAP and AUC results of different methods on four data sets. As shown in the results, our methods (both TER and

	CiteULike (title)	CiteULike (title&abs.)	News (title)	News (title&sum.)
Cosine	0.5535 / 0.8194	0.7116 / 0.9162	0.3526 / 0.6950	0.4580 / 0.7721
Multitask	0.6129 / 0.8441	0.7355 / 0.9258	0.4051 / 0.7085	0.4560 / 0.7760
Content Pretrained	0.6250 / 0.8961	0.7310 / 0.9372	0.4512 / 0.8145	0.4778 / 0.8352
CDL	0.6182 / 0.8839	0.7484 / 0.9410	0.3549 / 0.7648	0.4477 / 0.8060
TER (MoV)	0.6789 / 0.9201	0.7476 / 0.9432	0.497 / 0.8294	0.5272 / 0.8515
TER (CNN)	0.6908 / 0.9264	0.7519 / 0.9458	0.5069 / 0.8470	0.5227 / 0.8580
TER+ (MoV)	**0.7073 / 0.9309**	**0.7641 / 0.9485**	0.5020 / 0.8462	0.5294 / **0.8628**
TER+ (CNN)	0.6990 / 0.9274	**0.7620 / 0.9478**	**0.5149 / 0.8541**	0.5353 / **0.8626**

Table 3: Performance of different methods on four data sets. Two metrics are reported, namely MAP (left to slash), and AUC (right to slash). For both metrics, the larger the better.

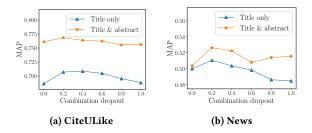**(a) CiteULike**   **(b) News**

Figure 3: Mean Average Precision of different dropout rate for utilizing unsupervised text embedding.

TER+) consistently beat other baselines and achieve state-of-the-art performance. Other several important observations can also be made from the results: 1) representation learning or embeddings methods (our methods, pre-trained method and CDL) can achieve better results compared to traditional TF-IDF based methods, 2) the joint supervised and unsupervised text embedding can achieve better results compared to supervised or unsupervised text embedding alone, and 3) the advantage of our model on short texts is more significant compared to longer one. We also observe that Mov outperforms CNN in some cases (e.g. in CiteULike data sets), we conjecture this is due to that words in CiteUlike may be more indicative w.r.t. user interests so simpler embedding functions can already well capture the semantics.

Figure 3 shows performances of different dropout rate for pre-trained text embedding vector h_2 in combination function $g(h_1, h_2)$. We observe that, as dropout rate increases, most of the curves go up and then go down. The peak occurs mostly around 0.2 to 0.4, both the 0 and 1 two extreme points have worse results. This further confirms the effectiveness of incorporating unsupervised text embedding, and also show that certain level of noise injected into pre-trained text embedding can improve performance.

4.4 Case Studies

To further understand the proposed model, we conduct several case studies looking into the layout or nearest neighbors of words and articles in the embedding space.

To visualize the text embedding learned from different models, we firstly choose top conferences in five domains (ML, DM, CV, HCI, BIO), and then randomly select articles that are published in those conferences. We apply TSNE [17] to visualize 2d map for these articles, and color them according to their domains of publication. The results are shown in Figure 4 where we found that our combined model can best distinguish papers from different domains.

Table 4 shows similar words for given queried words, i.e. "neural" and "learning", in CiteULike data set. From the result we clearly see the distinction between meanings of word learned from both methods. For example, the nearest word "neural" learned in unsupervised text embedding (articles with and without user like behavior) is mostly related to artificial neural networks, but in supervised text embedding, it is mostly related to neuroscience, which is more close to biology. This is because that in the CiteULike data set, there exist a lot of biologists, so the word embedding learned from supervised text embedding is likely to be dominated by the neuroscience perspective. However, by incorporating the unsupervised text embedding learned from a larger corpus, more meanings of the words can be recovered.

Table 5 shows the similar articles given a randomly selected queried article. We find that although unsupervised text embedding can provide some similar articles, the proposed framework (both TER and TER+) can better capture the similarity of articles.

5 RELATED WORK

Our work is related to both personalized recommendation and text embedding and understanding.

Collaborative filtering [15] has been one of the most effective methods in recommender system. Methods like matrix factorization [14, 24] are wide adopted, and recently some methods based on neural networks are also explored [26, 31, 33]. Content based methods are proposed [2, 21], but has not been well developed to exploit deep semantics of content information. Hybrid methods can improve so-called "cold-start" issue by incorporating side information [5, 22, 27], or item content information [8, 30, 31]. In our case, although we have historical data about users' interactions with items, but at the time of recommendation we are considering the items that have never been seen before, which cannot be handle directly by most existing matrix factorization based methods. Our model is similar to CDL [31], but with following differences: (1) we treat the problem as ranking instead of rating prediction problem, (2) we provide a general framework which allows flexible choice
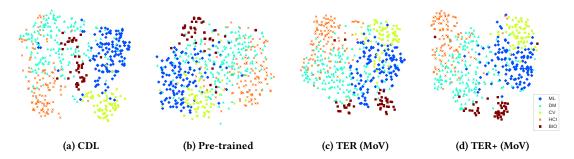
| (a) CDL | (b) Pre-trained | (c) TER (MoV) | (d) TER+ (MoV) |

Figure 4: Article embeddings for papers of different domains.

Unsupervised	recurrent feedforward artificial feed multilayer trained neuron chaotic parameters
Supervised	cortex motor spike hippocampal attention sensory train parietal perceptual

(a) Most similar words to "neural".

Unsupervised	training styles learners experts contexts activities reinforcement traditional concept
Supervised	measuring rehabilitation courses multimodal elearning special review sense instruction

(b) Most similar words to "learning".

Table 4: Most similar words according to the learned word embedding.

	Pretrained	TER+ (MoV)
1	Portraits of complex networks	Navigability of complex networks
2	*Exploring the diversity of complex metabolic networks*	Portraits of complex networks
3	Synchronization in complex networks	Hierarchical organization in complex networks
4	*Lethality and centrality in protein networks*	Scale free networks
5	*Exploring the new world of the genome with dna microarrays*	*Structure of growing social networks*

(a) Queried paper: Exploring complex networks.

	Pretrained	Combined
1	Communities in networks	Semiotic dynamics in online social communities
2	*Scientific computing in the cloud*	Communities in cyberspace
3	Communities and technologies	Audience , structure and authority in the weblog community
4	*Cloud computing*	Communities and technologies
5	*Computing with membranes*	Conversations in the blogosphere an analysis from the bo...

(b) Queried paper: Mapping weblog communities.

Table 5: Most similar articles found based on different text embedding. We use *italics* to highlight those "inaccurate" results.

of text embedding function $f(x)$, and (3) our model can explicitly incorporate unsupervised text embedding.

To understand text data, both supervised and unsupervised methods are proposed. Supervised methods are usually guided by text labels, such as sentiment labels or category labels. Different from traditional text classification, which train SVM or logistic regression classifiers based on n-gram features [9, 20], recent work take advantage of distributed representation brought by embedding methods, which include CNN [6, 12, 32], RNN [29] and others [11]. Those methods cannot be directly applied for recommendation as only a global classification/ranking model is provided. Also, instead of using labels as in existing supervised text embedding methods,

we utilize user item interactions as supervision to learn the text embedding function. There are also unsupervised text embedding techniques [16, 18, 19], which do not require labels but cannot adapt to the task of interest.

We further generalize the proposed model and develop efficient training techniques in [3].

6 CONCLUSIONS

In this work, we tackle the problem of content-based recommendation for completely new texts. An novel joint text embedding based framework is proposed, in which user embedding and text embedding function are learned end-to-end based on interactions

between users and items. The text embedding function is flexible, and can be specified by deep neural networks. Both supervised and unsupervised text embeddings are fused together by an combination module as part of a unified model. Empirical evaluations based on real-world data sets demonstrate that our model can achieve state-of-the-art results for recommending new texts. As for the future work, it is interesting to explore other ways of incorporating unsupervised text embeddings.

ACKNOWLEDGEMENTS

REFERENCES

[1] Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the GRU: Multi-task Learning for Deep Text Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 107–114.
[2] Ting Chen and Yizhou Sun. 2017. Task-Guided and Path-Augmented Heterogeneous Network Embedding for Author Identification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 295–304.
[3] Ting Chen, Yizhou Sun, Yue Shi, and Liangjie Hong. 2017. On Sampling Strategies for Neural Network-based Collaborative Filtering. In *Proceedings of the 23th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.
[4] Ting Chen, Lu-An Tang, Yizhou Sun, Zhengzhang Chen, and Kai Zhang. 2016. Entity Embedding-based Anomaly Detection for Heterogeneous Categorical Events. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16)*. Miami.
[5] Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. 2012. SVDFeature: a toolkit for feature-based collaborative filtering. *Journal of Machine Learning Research* 13, Dec (2012), 3619–3622.
[6] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12, Aug (2011), 2493–2537.
[7] Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 109–117.
[8] Prem K Gopalan, Laurent Charlin, and David Blei. 2014. Content-based recommendations with poisson factorization. In *Advances in Neural Information Processing Systems*. 3176–3184.
[9] Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*. Springer, 137–142.
[10] Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058* (2014).
[11] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of Tricks for Efficient Text Classification. *arXiv preprint arXiv:1607.01759* (2016).
[12] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).
[13] Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: A Method for Stochastic Optimization. (2015).
[14] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 426–434.
[15] Yehuda Koren, Robert Bell, Chris Volinsky, and others. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
[16] Quoc V Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents.. In *ICML*, Vol. 14. 1188–1196.
[17] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.
[18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
[19] T Mikolov and J Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems* (2013).
[20] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, 79–86.

[21] Michael J Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The adaptive web*. Springer, 325–341.
[22] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.
[23] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.
[24] Ruslan Salakhutdinov and Andriy Mnih. 2011. Probabilistic matrix factorization. In *NIPS*, Vol. 20. 1–8.
[25] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*. ACM, 791–798.
[26] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 111–112.
[27] Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 650–658.
[28] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.
[29] Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 1422–1432.
[30] Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 448–456.
[31] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1235–1244.
[32] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*. 649–657.
[33] Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. 2016. A Neural Autoregressive Approach to Collaborative Filtering. *arXiv preprint arXiv:1605.09477* (2016).