CAMO: A Collaborative Ranking Method for Content Based Recommendation

Chengwei Wang,[1,3] **Tengfei Zhou,**[3] **Chen Chen,**[1,3] **Tianlei Hu,**[1,3,*] **Gang Chen**[2,3]
[1]The Key Laboratory of Big Data Intelligent Computing of Zhejiang Province, China
[2]CAD & CG State Key Lab, Zhejiang University, China
[3]College of Computer Science and Technology, Zhejiang University, China
{rr, zhoutengfei, cc33, htl, cg}@zju.edu.cn

Abstract

In real-world recommendation tasks, feedback data are usually sparse. Therefore, a recommender's performance is often determined by how much information that it can extract from textual contents. However, current methods do not make full use of the semantic information. They encode the textual contents either by "bag-of-words" technique or Recurrent Neural Network (RNN). The former neglects the order of words while the latter ignores the fact that textual contents can contain multiple topics. Besides, there exists a dilemma in designing a recommender. On the one hand, we shall use a sophisticated model to exploit every drop of information in item contents; on the other hand, we shall adopt a simple model to prevent itself from over-fitting when facing the sparse feedbacks. To fill the gaps, we propose a recommender named CAMO [1]. CAMO employs a multi-layer content encoder for simultaneously capturing the semantic information of multi-topic and word order. Moreover, CAMO makes use of adversarial training to prevent the complex encoder from over-fitting. Extensive empirical studies show that CAMO outperforms state-of-the-art methods in predicting users' preferences.

Introduction

Recommender System (RS) is one of the most important applications of artificial intelligence (Bennett and Lanning 2007; Linden, Smith, and York 2003; Su and Khoshgoftaar 2009). In e-commerce, RS is capable of selecting customized commodities for users and boosting vendors' profits (Zhu et al. 2014). In online media, RS is able to provide personalized playlists and increase users' engagement (Bonnin and Jannach 2013). In information retrieval, RS can be used to filter out users' unwanted links and improve their searching efficiency (Zhang, Yao, and Sun 2017).

RS aims to mine users' preferences from their feedbacks, including explicit ratings and implicit clicks (Rendle et al. 2009). Therefore, many methods use feedbacks as the sole data source to train a recommendation model (Mnih and Salakhutdinov 2007; Koren 2008; Koren, Bell, and Volinsky 2009). However, the feedbacks are often severely sparse in practical applications, which makes such methods produce inferior recommendation results. To enhance the performance, many researchers indicate that an item's content information is capable of boosting model's expressing power (Kim et al. 2016; Gupta and Varma 2017). Consequently, RSs with item content have higher accuracy than their vanilla counterparts (Wang and Blei 2011; Gopalan, Charlin, and Blei 2014).

In this paper, we focus on fusing RS with textual contents. Many Content-Based Recommendation (CBR) approaches have been proposed for exploiting a corpus of textual contents. Generally, CBR methods combine a Collaborative Filtering (CF) function with an encoder which extracts features from the text (Bansal, Belanger, and McCallum 2016; Chen et al. 2017; Xiaoyan Cai and Yang 2018). Note that these encoders dominate the performance of such methods since they have similar CF architectures. Though abundant literature have comprehensively studied the construction of content encoders (Zhang et al. 2017; Agarwal and Chen 2010; Volkovs, Yu, and Poutanen 2017), we find that there are still significant rooms for improvements.

Firstly, most existing methods fail to simultaneously capture topic semantics and word order information, since they belong to different semantic hierarchies. For topic-aware encoders, the content features are extracted by models such as Latent Dirichlet Allocation and autoencoder (Liu et al. 2017; Gopalan, Charlin, and Blei 2014; Wang, Wang, and Yeung 2015). Since these models are built upon the "bag-of-words" assumption, they are not sensitive to the order of words. For example, "I prefer apple to banana" and "I prefer banana to apple" are semantically equivalent to such methods, which induces serious bias in text understanding. To catch word order, many researchers use a Recurrent Neural Network (RNN) such as Gated Recurrent Unit (GRU) or Long Short-Term Memory (LSTM) to encode content text (Wang, Xingjian, and Yeung 2016; Bansal, Belanger, and McCallum 2016). However, they treat a document as one sequence of words, which fails to identify the high leveled topic information.

Secondly, a sophisticated encoder has poor generalization for unseen content. Empirical evidence given by (Almahairi et al. 2015) shows that replacing a simple encoder such as word embedding with a complex RNN can make recommendation accuracy drop seriously due to over-fitting. To re-

*Corresponding author.

[1]CAMO is short for Collaborative rAnking via dynaMic rOuting.

duce over-fitting, Wang, Xingjian, and Yeung (2016) injects noises into content corpus. Bansal, Belanger, and McCallum (2016) utilize an extra tag classification task to regularize the complex RNN model. The previous two strategies are not optimal. Injecting noises into content data inevitably contaminates the raw text, which reduces the quality of content data. Regularization via tag may be impractical in real-world tasks since valuable tags are hard to obtain.

To resolve these difficulties, we propose a CBR method called CAMO. Our main contributions are listed as follows.

- We design a novel neural architecture which extracts both topic and word order information. The encoder uses a multi-layer structure to mimic the semantic hierarchies. In the lower layer, a GRU is used to exploit word order. In the higher layer, a multi-head attention mechanism is employed to summarize topics.
- A variant of multi-head attention mechanism is devised. The proposed method uses a dynamic routing protocol to update the attention signal iteratively. Such protocol can allow attention-heads decide the pooling weights of lower-level states according to the predicted output. This allows attention-heads prune unrelated part and attend to active parts of a document.
- We utilize an adversarial training technique to enhance the generalization power of our content encoder. We view the training process as a minimax game between a content encoder and an opponent generator. The encoder tries to construct a content vector that minimizes the decoding error. The generator produces a document from the content vector and tries to maximize the distance between its output and the original document.

Preliminaries and Notations

Consider the following typical CBR scenario for inferencing m users' preferences over n items. Suppose two kinds of data are available. One is the content corpus $C = \{doc_j\}_{j=1}^{n}$ where each document doc_j contains textual description of item j, the other is users' feedbacks $D = \{(i, \rho_i^+, \rho_i^-)\}_{i=1}^{n}$ where ρ_i^+ contains all items labeled with "relevant" by user i while ρ_i^- is comprised of all items labeled with "not relevant". The goal of CBR is to learn a preference score function $f(i, j, doc_j)$ which gives higher scores to items in ρ_i^+ and lower scores to items in ρ_i^-. Formally, the learning process can be expressed as follows (Christakopoulou and Banerjee 2015).

$$\min_{\theta_f} \sum_{i=1}^{m} \sum_{j \in \rho_i^+} \sum_{j' \in \rho_i^-} l(i, j, j') \triangleq L_{rank}(\theta_f) \tag{1}$$

where θ_f are the parameters of $f(i, j, doc_j)$ and $l(i, j, j') = -\log \sigma(f(i, j, doc_j) - f(i, j', doc_{j'}))$ with $\sigma(\cdot)$ being the sigmoid function. The preference score function can be modeled as

$$f(i, j, doc_j) = p_i^\top (q_j + g(doc_j)) \tag{2}$$

where p_i is the latent vector of user i, q_j is the latent vector of item j, and $g(\cdot)$ is a content encoder which maps doc_j into semantic vector. Equation (2) means that the final item representation is the sum of content vector $g(doc_j)$ and item-specific vector q_j (Bansal, Belanger, and McCallum 2016). We list the notations used in the paper in Tab. 1.

Table 1: Table of notations

Variable	Description
I_{topic}	Number of topics
I_{sent}	Number of sentences
I_{word}	Number of words
$p_i, \mathbf{P}$	User's latent vector and matrix
$q_i, \mathbf{Q}$	Item's latent vector and matrix
$\mathbf{W}_p$	Alignment matrix of topic space
$\mathbf{W}_{word}$	Embedding matrix of words
$\mathbf{W}_{dec}$	Decoding matrix of a GRU
$v^{doc}, g(doc_j)$	Semantic vector of a document
v_i^{topic}	Semantic vector of a topic
v^{sent}, v_i^{sent}	Semantic vector of a sentence
v^{word}, v_i^{word}	Semantic vector of a word
α_i^{attent}	Signal of a attention mechanism
h_i^{gru}	A hidden state of a GRU network
θ_{cont}	Set of the content encoder's parameters
θ_{dec}	Set of the generator's parameters
θ_{gru}	Set of a GRU's parameters
$\alpha_{p,i}$	Signal of Dynamic Routing
r	Number of Dynamic Routing iterations
$\hat{\partial}_x f(x)$	Stochastic gradient of function f

Methodology

Content Encoder

We use a three layer content network to encode semantic information of corpus into Euclidean vectors. The architecture of the content network is shown in Fig. 1. From top to bottom, a Doc2Vec layer aggregates sentence vectors into a topic-aware document vector. A Sent2Vec layer pools word vectors into sentence vectors. A Word2Vec layer embeds each word into a semantic space.

Doc2Vec Suppose that document doc is comprised of sentence vectors $\{v_i^{sent}\}_{i=1}^{I_{sent}}$ and topic vectors $\{v_i^{topic}\}_{i=1}^{I_{topic}}$. We make two assumptions. Assumption I: document vector v^{doc} is equal to the average of topic vectors v_p^{topic}

$$v^{doc} = \frac{1}{I_{topic}} \sum_{i=1}^{I_{topic}} v_i^{topic}. \tag{3}$$

Assumption II: v_p^{topic} is equal to the weighted mean of sentence vectors

$$v_p^{topic} = \sum_{i=1}^{I_{sent}} \alpha_{p,i} \mathbf{W}_p v_i^{sent} \tag{4}$$

where the weight $\alpha_{p,i}$ satisfies $\alpha_{p,i} \geq 0, \sum_i \alpha_{p,i} = 1$, and $\mathbf{W}_p$ is the alignment matrix. The weight $\alpha_{p,i}$ can be interpreted as the probability that topic p is involved in sentence

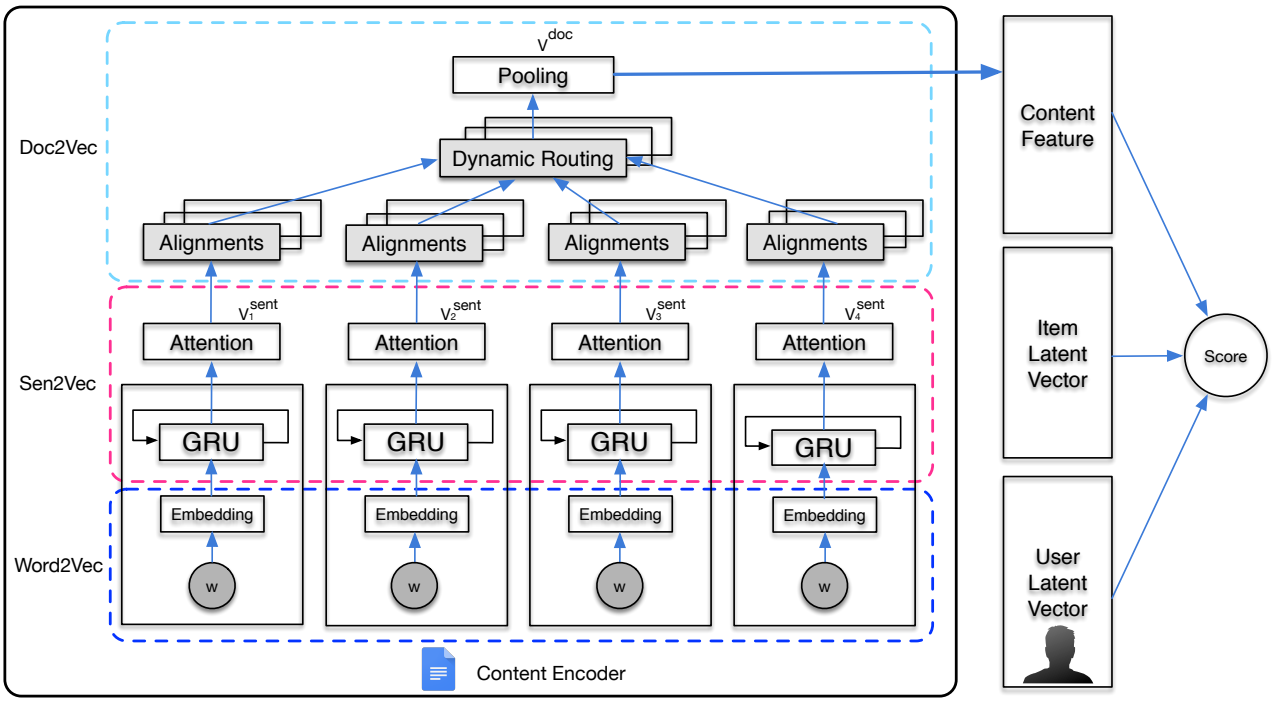

Figure 1: Architecture of CAMO.

i. Intuitively, the inner product of v_p^{topic} and $\mathbf{W}_p v_i^{sent}$ shall reflect the semantic relativeness of the two object. Thus, we formulate the probability $\alpha_{p,i}$ as follows.

$$\alpha_{p,i} \propto \exp((v_p^{topic})^\top \mathbf{W}_p v_i^{sent}). \tag{5}$$

Substitute equation (5) into (4), we obtain the following non-linear equation of topic vector.

$$v_p^{topic} = \frac{\sum_{i=1}^{I_{sent}} \exp((v_p^{topic})^\top \mathbf{W}_p v_i^{sent}) \mathbf{W}_p v_i^{sent}}{\sum_{i=1}^{I_{sent}} \exp((v_p^{topic})^\top \mathbf{W}_p v_i^{sent})}. \tag{6}$$

The above fixed point equation is of the form $x = f(x)$, and it can be numerically solved by iteratively calculating $x^{(t+1)} = f(x^{(t)})$ (Stoer and Bulirsch 2013). In this way, we get the recurrence for computing the topic vector

$$h_p^{(t)} = \begin{cases} 0 & t = 0 \\ \frac{\sum_i \exp(h_p^{(t-1)\top} \mathbf{W}_p v_i^{sent}) \mathbf{W}_p v_i^{sent}}{\sum_i \exp(h_p^{(t-1)\top} \mathbf{W}_p v_i^{sent})} & 0 < t \leq r \end{cases}. \tag{7}$$

We summarize the Doc2Vec protocol in Alg. 1.

Remark 1 *The Doc2Vec protocol can be simplified as the following formulation.*

$$\begin{cases} v^{doc} = \frac{1}{I_{topic}} \sum_{p=1}^{I_{topic}} v_p^{topic} \\ v_p^{topic} = \mathbf{V} \operatorname{softmax}(\mathbf{K}^\top \mathbf{W}_p^\top q_p), 1 \leq p \leq I_{topic} \end{cases} \tag{8}$$

where $\mathbf{K} = (v_1^{sent}, v_2^{sent}, \ldots, v_{I_{sent}}^{sent})$ *is the "key matrix",* $q_p = h_p^{(r-1)}$ *is the query vector and* $\mathbf{V} = \mathbf{W}_p \mathbf{K}$ *is the "value matrix". Therefore, the Doc2Vec protocol shares the same form of multi-head attention mechanism (Vaswani et al. 2017). The difference between them is that the proposed "query vector" is formed by a dynamic routing method.*

Algorithm 1 Doc2Vec($v_1^{sent}, v_2^{sent}, \ldots v_{I_{sent}}^{sent}$)

1: **for** $t \in \{0, 1, 2, \ldots r\}$ **do**
2: for all $1 \leq p \leq I_{topic}$: update hidden state $h_p^{(t)}$ via equation (7)
3: **end for**
4: for all $1 \leq p \leq I_{topic}$: $v_p^{topic} = h_p^{(r)}$
5: $v^{doc} = \frac{1}{I_{topic}} \sum_{p=1}^{I_{topic}} v_p^{topic}$
6: **return** v^{doc}

Sent2Vec Layer A sentence vector v^{sent} ($v^{sent} \in \{v_i^{sent}\}$) is generated by the following attention formulation.

$$v^{sent} = \sum_{i=1}^{I_{word}} \alpha_i^{attent} h_i^{gru} \tag{9}$$

where h_i^{gru} is the hidden state of a GRU and α_i^{attent} is the attention signal for pooling the GRU's hidden states. GRU is used to capture the order information of words, its hidden states are given by the following recurrence.

$$h_i^{gru} = \begin{cases} 0 & i = 0 \\ \text{GRUCell}(h_{i-1}^{gru}, v_i^{word}) & i > 0 \end{cases} \tag{10}$$

where the transition function $h' = \text{GRUCell}(h, v)$ (Chung et al. 2014) can be formulated as follows.

$$\begin{cases} z = \sigma(\mathbf{W}_z v + \mathbf{U}_z h + b_z) \\ r = \sigma(\mathbf{W}_r v + \mathbf{U}_r h + b_r) \\ h' = (1 - z) * h \\ \quad\quad + z * (\tanh(\mathbf{W}_h v + \mathbf{U}_h (r * h) + b_h)) \end{cases}$$

with matrix $\mathbf{W}_z, \mathbf{W}_r, \mathbf{W}_h$, matrix $\mathbf{U}_z, \mathbf{U}_r, \mathbf{U}_h$, and vector b_z, b_r, b_h being the GRU's parameters. The attention signal $\alpha^{attent} = (\alpha_1^{attent}, \ldots, \alpha_{I_{word}}^{attent})^\top$ is given by the following equation

$$\alpha^{attent} = \text{softmax}(\mathbf{H}_{GRU}^\top h_{I_{word}}^{gru}) \tag{11}$$

where matrix $\mathbf{H}_{GRU}$ contains the hidden states of GRU cells.

Word2Vec Layer Let w be a word belonging to the vocabulary of articles. The vector representation v^{word} of word w is given by

$$v^{word} = \mathbf{W}_{word}(w, :) \tag{12}$$

where $\mathbf{W}_{word}$ is the embedding matrix.

Model Training

Complex models are notorious for their over-fitting in the task of personal recommendation. Such phenomenon will make the proposed model perform inferiorly on the unrecorded data. According to the framework of multi-task learning, constructing an auxiliary learning task for a personal ranking task can greatly reduce the risk of over-fitting (Ruder 2017). Therefore, training a decoder to map document vectors $\{v^{doc}\}$ back into corpus $\{doc\}$ is a natural choice.

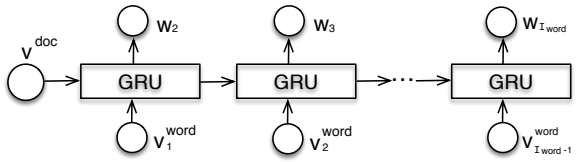

Figure 2: The generator produces texts from v^{doc}.

The Decoding Task Let θ_{cont} denote the parameters of the content network (including the word embedding matrix $\mathbf{W}_{word}$, topic alignment matrix $\mathbf{W}_p$, and so on). Let θ_{dec} denote the parameters for the generator of the document vector. For a document $doc = \{w_i\}$, we define the transition probability of texts as the following softmax distribution

$$p(w_{i+1}|w_i, \theta_{cont}, \theta_{dec}) = \text{softmax}(\mathbf{W}_{dec} h_i^{dec}) \tag{13}$$

where $\mathbf{W}_{dec}$ is a matrix and hidden states $\{h_i^{dec}\}$ are generated from a GRU with hidden states given by:

$$h_i^{dec} = \begin{cases} v^{doc} & i = 0 \\ \text{GRUCell}(h_{i-1}^{dec}, v_i^{word}) & i > 0 \end{cases}. \tag{14}$$

where v_i^{word} is the word vector of w_i. For the corpus $C = \{doc\}$, the negative log-likelihood L_{dec} can be composed as follows.

$$L_{dec}(\theta_{cont}, \theta_{dec}) = -\sum_{doc} \sum_{w_i \in doc} \log p(w_{i+1}^{doc}|w_i^{doc}, \theta_{cont}, \theta_{dec}).$$

One can learn the parameters of the decoding task via minimize $L_{dec}(\cdot, \cdot)$.

Algorithm 2 Alternate Gradient Method

Require: $\mathbf{P}^{(0)}, \mathbf{Q}^{(0)}, \theta_{cont}^{(0)}, \mathbf{W}_{dec}^{(0)}, \theta'$ and $\theta_{gru}^{(0)}$.

1: **for** $t \in \{0, 2, \ldots T\}$ **do**

2: Compute stochastic gradients

$$\begin{aligned} g_{\mathbf{P}} &= \hat{\partial}_{\mathbf{P}} L_{rank}(\mathbf{P}^{(t)}, \mathbf{Q}^{(t)}, \theta_{cont}^{(t)}) \\ g_{\mathbf{Q}} &= \hat{\partial}_{\mathbf{Q}} L_{rank}(\mathbf{P}^{(t)}, \mathbf{Q}^{(t)}, \theta_{cont}^{(t)}) \\ g_{\theta_{cont}} &= \hat{\partial}_{\theta_{cont}} L_{rank}(\mathbf{P}^{(t)}, \mathbf{Q}^{(t)}, \theta_{cont}^{(t)}) \\ g'_{\theta_{cont}} &= \hat{\partial}_{\theta_{cont}} \lambda L_{dec}(\theta_{cont}^{(t)}, \mathbf{W}_{dec}^{(t)}, \theta_{gru}^{(t)}) \\ g'_{\mathbf{W}_{dec}} &= \hat{\partial}_{\mathbf{W}_{dec}} \lambda L_{dec}(\theta_{cont}^{(t)}, \mathbf{W}_{dec}^{(t)}, \theta_{gru}^{(t)}) \\ g'_{\theta_{gru}} &= \hat{\partial}_{\theta_{gru}} \lambda L_{dec}(\theta_{cont}^{(t)}, \mathbf{W}_{dec}^{(t)}, \theta_{gru}^{(t)}) \end{aligned}$$

3: Perform stochastic gradient descent with stepsizes η_t, δ_t

$$\begin{aligned} \mathbf{P}^{(t+1)} &= \mathbf{P}^{(t)} - \eta_t g_{\mathbf{P}} \\ \mathbf{Q}^{(t+1)} &= \mathbf{Q}^{(t)} - \eta_t g_{\mathbf{Q}} \\ \theta_{cont}^{(t+1)} &= \theta_{cont}^{(t)} - \eta_t g_{\theta_{cont}} - \delta_t g'_{\theta_{cont}} \\ \mathbf{W}_{dec}^{(t+1)} &= \mathbf{W}_{dec}^{(t)} - \delta_t g'_{\mathbf{W}_{dec}} \end{aligned}$$

4: Perform stochastic gradient ascent with stepsize δ_t

$$\theta_{gru}^{(t+1)} = \theta_{gru}^{(t)} + \delta_t g'_{\theta_{gru}}$$

5: {Project θ_{gru} onto the feasible set}

6: **if** $\|\theta_{gru}^{(t+1)} - \theta'\| > \gamma$ **then**

7: $\theta_{gru}^{(t+1)} = \theta' + \gamma(\theta_{gru}^{(t+1)} - \theta')/\|\theta_{gru}^{(t+1)} - \theta'\|$

8: **end if**

9: **end for**

Adversarial Training Intuitively, multi-task learning of collaborative ranking and textual decoding can be expressed as follows.

$$\min_{\mathbf{P}, \mathbf{Q}, \theta_{cont}, \theta_{dec}} L_{rank}(\mathbf{P}, \mathbf{Q}, \theta_{cont}) + \lambda L_{dec}(\theta_{cont}, \theta_{dec}). \tag{15}$$

However, we find the encoder estimated by equation (15) performs worse than a simple Word2Vec model. Almahairi et al. (2015) also point out this issue. This shows that expression (15) is not suitable for estimating model parameters, since a small $L_{dec}(\cdot)$ may result from the strong fitting power of GRU other than the generalization power of the learned feature v^{doc}. To rule out the optimistic bias induced by GRU, we modify the composite loss as follows.

$$\begin{aligned} \min_{\mathbf{P}, \mathbf{Q}, \theta_{cont}, \mathbf{W}_{dec}} \{ & L_{rank}(\mathbf{P}, \mathbf{Q}, \theta_{cont}) \\ & + \lambda \max_{\theta_{gru}, \|\theta_{gru} - \theta'\| \leq \gamma} L_{dec}(\theta_{cont}, \mathbf{W}_{dec}, \theta_{gru}) \} \end{aligned} \tag{16}$$

where θ_{gru} is the parameters of the GRU, θ' is a randomly generated vector and γ is a tuning parameter. Our intuition is that if the decoding loss is still small for an adversarial constructed GRU, one can infer that even a bad decoder can not distort the generalization power of v^{doc}. The minimax problem (16) can be solved by performing alternate gradient method and we summarize it in Alg. 2.

Experiments

We conduct extensive experiments on benchmark datasets. We use the following state-of-the-art methods as baselines. Note that we do not compare with matrix factorization models such as BPR, WRMF, and FM since it has shown that CML and CDL outperform these models (Hsieh et al. 2017; Wang, Wang, and Yeung 2015).

- CML + Skip-Thought (CMLST). We combine CML (Hsieh et al. 2017), a metric based collaborative ranking method, with text encoder Skip-Thought (Kiros et al. 2015) to model both users' feedbacks and content texts.
- CRAE (Wang, Xingjian, and Yeung 2016). CRAE contains a Bayesian matrix factorization model for modeling users' feedbacks and a Bayesian RNN for encoding content corpus.
- CDL (Wang, Wang, and Yeung 2015). CDL is a joint probabilistic graphics model that integrates the matrix factorization model with probabilistic autoencoder.
- CDAE (Wu et al. 2016). CDAE is a denoising autoencoder for users' feedbacks.

The codes for CML, Skip-Thought, CDL, and CDAE are downloaded from the homepage of the correspondent author. The codes for the proposed CAMO and CRAE are implemented using TensorFlow (Abadi et al. 2016). All the compared methods are run on the same machine with i7-5930K CPU, 64GB RAM, and one TITAN Xp GPU.

Datasets

The benchmark datasets include: CiteULike (Wang and Blei 2011), MovPlot1M and MovPlot10M (Liu et al. 2017). CiteULike contains users' bookmarks of scientific articles and paper abstracts. MovPlot1M and MovPlot10M are extensions of MovieLens1M and MovieLens10M datasets. Both contain movie ratings and corpus of movie plots. We treat every rated movie as "relevant". For each "relevant" user-item pair in all datasets, we sample NS items without bookmarks (or ratings) as "not relevant". NS is dubbed negative sampling ratio. The statistics of the datasets are listed in Tab. 2. We randomly select 80% of the observed data as training set and evaluate the models with the remaining 20%.

Table 2: Statistics of the benchmark datasets where #users, #items and #feedbacks are the number of users, items and feedbacks respectively, "avg. words" is average words per document.

Dataset	#users	#items	#feedbacks	avg. words
CiteULike	5551	16980	210504	204.9
MovPlot1M	6040	3861	996045	82.19
MovPlot10M	13975	10681	1962580	84.66

Default Hyperparameter Settings

We set all the dimension hyperparameters (e.g. dimensions of the user, item and document latent vectors) of the considered models to the same value d. We call d model's dimension and set $d = 256$ for all compared methods by default. We set negative sampling ratio $NS = 6$. The default parameters for CAMO are set as follows: the number of topics I_{topic} is set to 10, the number of dynamic routing iterations r is set to 14, and the regularization parameter λ is set to 0.2. Other parameters of baseline methods are set to their default values.

Performance of Each Neural Layer

In this section, we study the influence of semantic vectors constructed by different layers of the proposed encoder. Let doc be a document containing words $\{w_i\}_{i=1}^{I_{word}}$. To show the impact of Word2Vec layer, we evaluate a Word2Vec variant of CAMO. The variant is constructed by replacing CAMO's content vectors with the mean of word vectors, i.e. $v^{doc} = \frac{1}{I_{word}} \sum_i v_i^{word}$. Similarly, we build a Sent2Vec variant of CAMO using the average of sentence vectors as content features. The experimental results are shown in Tab. 3. From the table, we can see that accuracy of Sent2Vec variant is much better than Word2Vec. This can be explained by that Sent2Vec captures the semantic information of word order, while Word2Vec does not. The table also shows that CAMO is more accurate than its Sent2Vec variant. This is because Doc2Vec layer encodes extra topic semantic information to content features.

Table 3: Performance of CAMO's different neural layers Pre@5, Pre@10, Rec@5 and Rec@10 are Precision@5, Precision@10, Recall@5 and Recall@10 respectively. MAP is short for mean average precision. AUC is abbreviated for Area Under the ROC.

CiteULike	MAP	AUC	Pre@5	Pre@10	Rec@5	Rec@10
Word2Vec	0.1340	0.9404	0.1496	0.1218	0.1131	0.1724
Sent2Vec	0.1589	0.9283	0.1730	0.1374	0.1317	0.1930
CAMO	**0.1623**	**0.9540**	**0.1779**	**0.1414**	**0.1357**	**0.2019**
MovPlot1M	MAP	AUC	Pre@5	Pre@10	Rec@5	Rec@10
Word2Vec	0.2145	0.9168	0.3973	0.3418	0.0892	0.1425
Sent2Vec	0.2199	0.9215	0.4059	0.3468	0.0911	0.1474
CAMO	**0.2439**	**0.9296**	**0.4310**	**0.3701**	**0.0995**	**0.1629**
MovPlot10M	MAP	AUC	Pre@5	Pre@10	Rec@5	Rec@10
Word2Vec	0.2360	0.9168	0.3696	0.3174	0.1118	0.1818
Sent2Vec	0.2468	0.9724	0.3852	0.3208	0.1181	0.1917
CAMO	**0.2516**	**0.9737**	**0.3948**	**0.3360**	**0.1216**	**0.1985**

Quantitative Comparison

In this section, we compare the accuracy of the considered methods. To reduce randomness, we run each method five times and report the average. We use Area Under the ROC Curve (AUC) and Mean Average Precision (MAP) to measure the ranking accuracies of the compared methods. We report their accuracies on the three benchmark datasets in Tab. 4. From the table, we can see that CAMO

outperforms the baselines both on AUC and MAP. Moreover, users usually check a few top-ranked items in real-world applications, top-K evaluation metrics are important for studying the recommendation performance. Thus, we measure both Precision@K and Recall@K on benchmark datasets. The results are reported in Fig. 3 where (a1-c1) record Precision@K curves and (a2-c2) contain Recall@K curves. From Fig. 3 we can see that under different K, the proposed CAMO has consistently better results in terms of both criterions. We find that model dimension d and negative sampling ratio NS have a huge impact on the model's performance, and hence we analyze different settings of these parameters on testing accuracy. To visualize the impact of them, we report Recall@50 scores of CAMO, CMLST, and CRAE in Fig. 4 and Fig. 5. Both Fig. 4 and Fig. 5 show that CAMO have better Recall@50 than other methods consistently. In summary, all the above quantitative empirical results show the superior performance of CAMO. We attribute the promising empirical results to the encoding and generalization power of the multi-layer content network of CAMO.

Table 4: Performance of the compared methods.

Method	**CiteULike**		**MovPlot1M**		**MovPlot10M**	
	AUC	MAP	AUC	MAP	AUC	MAP
CMLST	0.9354	0.1217	0.9214	0.1881	0.9370	0.1753
CRAE	0.9416	0.0580	0.9157	0.1856	0.9682	0.1585
CDL	0.9120	0.1124	0.9188	0.1432	0.9365	0.1547
CDAE	0.9348	0.1211	0.9210	0.1630	0.9366	0.1712
CAMO	**0.9540**	**0.1623**	**0.9296**	**0.2439**	**0.9737**	**0.2516**

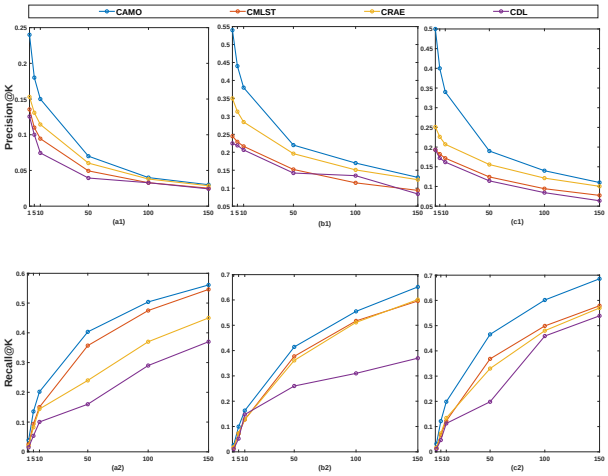

Figure 3: Precision@K and Recall@K of compared methods under different choices of K with respect to benchmark datasets.

Qualitative Comparison

To show CAMO's topic-aware extracting ability, we visualize the content feature of movies for Mov1MPlot dataset. We randomly sample 300 or 400 movies, and plot the corresponding features extracted by CAMO via t-SNE algorithm

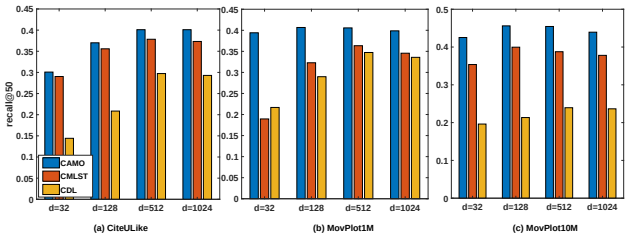

Figure 4: Recall@50 of compared methods under different choices of dimension with respect to benchmark datasets.

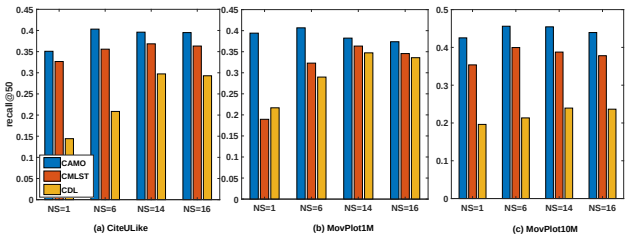

Figure 5: Recall@50 of compared methods under different choices of NS with respect to benchmark datasets.

(Maaten and Hinton 2008). In comparison, we also plot content features extracted by CMLST. The visualized results are displayed in Fig. 6. From the figure, we can see that movies are well clustered by CAMO while CMLST seem to confuse horror movies with drama. The better identification power of CAMO results from its topic-aware neural architecture.

To show the embedding capability of CAMO, we randomly choose two article from CiteULike and display the top-5 similar papers. The similarity is defined by Euclidean distance. For comparison, the results of CMLST are also recorded. We exhibit the experimental results in Tab. 5. It shows that when queried with paper "Amazon.com Recommendations Item-to-Item Collaborative Filtering", CAMO identifies both recommendation and e-commerce topics, while CMLST ignores the latter. We can also observe that when queried with paper "Towards Real-time Community Detection in Large Networks", CMLST even contains one irrelevant paper named "Adaptive networks coevolution of disease and topology", while all results of CAMO are relevant. These experimental studies demonstrate CAMO's embedding ability.

Conclusion

We propose a content-based recommender named CAMO. CAMO employs a multi-layer content encoder to transform textual corpus into content features. The lower layer of the content encoder contains a GRU which extracts semantic information of word order. The higher layer uses a dynamic routing protocol to ensemble features into topic vectors. To prevent the complex content encoder from over-fitting, we use adversarial training framework to enhance CAMO's generalization ability. Empirical studies show that CAMO outperforms state-of-the-art methods.

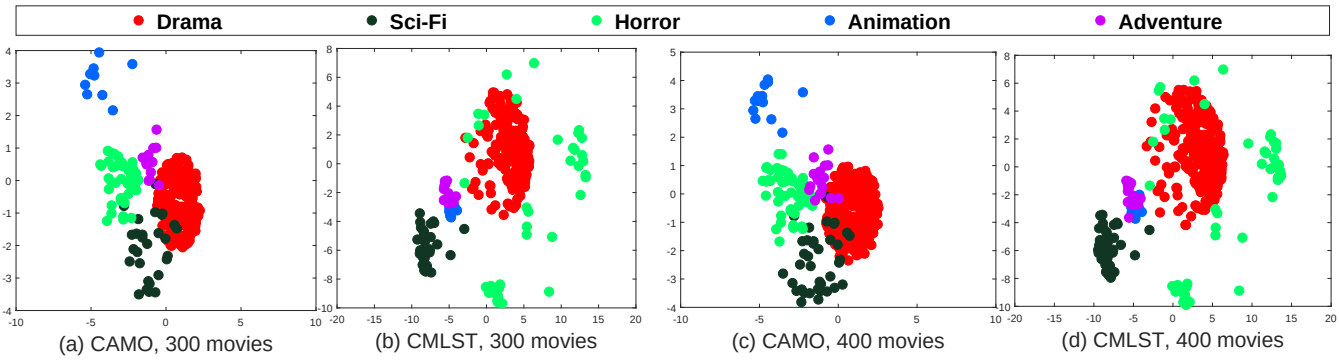

Figure 6: Comparison of t-SNE plots.

Table 5: Top-5 similar papers found by using the extracted document vector.

(a) Queried paper: Amazon.com recommendations item-to-item collaborative filtering
CAMO
1. Scalable collaborative filtering using cluster based smoothing
2. Analysis of recommendation algorithms for e-commerce
3. Applying collaborative filtering techniques to movie search for better ranking and browsing
4. Slope one predictors for online rating based collaborative filtering
5. Collaborative filtering by personality diagnosis a hybrid memory and model based approach
CMLST
1. Empirical analysis of predictive algorithms for collaborative filtering
2. Item based collaborative filtering recommendation algorithms
3. Improving recommendation lists through topic diversification
4. Recommender systems
5. Toward the next generation of recommender systems a survey of the state-of-the-art and possible extensions
(b) Queried paper:Towards real-time community detection in large networks
CAMO
1. Random hypergraphs and their applications
2. A comparative study of social network models network evolution models and nodal attribute models
3. Line graphs link partitions and overlapping communities
4. Statistical significance of communities in networks
5. Emergence of communities in weighted networks
CMLST
1. Line graphs link partitions and overlapping communities
2. Statistical significance of communities in networks
3. Random hypergraphs and their applications
4. Adaptive networks coevolution of disease and topology
5. The energy landscape of social balance

In the future, we would like to try the following two strategies to improve CAMO. Firstly, we would like to incorporate other kinds of content information such as images and videos into the model. Secondly, we will investigate how to use the probabilistic methods to enhance the robustness of CAMO.

Acknowledgments

This research is supported by National Key Research and Development Program (2017YFB1201001).

References

Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. 2016. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, 265–283.

Agarwal, D., and Chen, B.-C. 2010. flda: matrix factorization through latent dirichlet allocation. In *ACM international conference on Web search and data mining*.

Almahairi, A.; Kastner, K.; Cho, K.; and Courville, A. 2015. Learning distributed representations from reviews for collaborative filtering. In *Proceedings of the 9th ACM Conference on Recommender Systems*, 147–154. ACM.

Bansal, T.; Belanger, D.; and McCallum, A. 2016. Ask the gru: Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, 107–114. ACM.

Bennett, J., and Lanning, S. 2007. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, 35.

Bonnin, G., and Jannach, D. 2013. A comparison of playlist generation strategies for music recommendation and a new baseline scheme. In *Workshops at the twenty-seventh AAAI conference on artificial intelligence*.

Chen, T.; Sun, Y.; Shi, Y.; and Hong, L. 2017. On sampling strategies for neural network-based collaborative filtering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 767–776. ACM.

Christakopoulou, K., and Banerjee, A. 2015. Collaborative ranking with a push at the top. In *Proceedings of the 24th International Conference on World Wide Web*, 205–215.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Gopalan, P. K.; Charlin, L.; and Blei, D. 2014. Content-based recommendations with poisson factorization. In *Advances in Neural Information Processing Systems*, 3176–3184.

Gupta, S., and Varma, V. 2017. Scientific article recommendation by using distributed representations of text and graph. In *Proceedings of the 26th International Conference on World Wide Web Companion*, 1267–1268. International World Wide Web Conferences Steering Committee.

Hsieh, C.-K.; Yang, L.; Cui, Y.; Lin, T.-Y.; Belongie, S.; and Estrin, D. 2017. Collaborative metric learning. In *Proceedings of the 26th International Conference on World Wide Web*, 193–201. International World Wide Web Conferences Steering Committee.

Kim, D.; Park, C.; Oh, J.; Lee, S.; and Yu, H. 2016. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, 233–240. ACM.

Kiros, R.; Zhu, Y.; Salakhutdinov, R. R.; Zemel, R.; Urtasun, R.; Torralba, A.; and Fidler, S. 2015. Skip-thought vectors.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer* 42(8):30–37.

Koren, Y. 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*, 426–434. ACM.

Linden, G.; Smith, B.; and York, J. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE* 7(1):76–80.

Liu, C.; Jin, T.; Hoi, S. C.; Zhao, P.; and Sun, J. 2017. Collaborative topic regression for online recommender systems: an online and bayesian approach. *Machine Learning* 106(5):651–670.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *Journal of machine learning research* 9(Nov):2579–2605.

Mnih, A., and Salakhutdinov, R. 2007. Probabilistic matrix factorization. In *Advances in neural information processing systems*, 1257–1264.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, 452–461. AUAI Press.

Ruder, S. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Stoer, J., and Bulirsch, R. 2013. *Introduction to numerical analysis*, volume 12. Springer Science & Business Media.

Su, X., and Khoshgoftaar, T. M. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence* 2009:4.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*, 5998–6008.

Volkovs, M.; Yu, G.; and Poutanen, T. 2017. Dropoutnet: Addressing cold start in recommender systems. In *Advances in Neural Information Processing Systems*, 4957–4966.

Wang, C., and Blei, D. M. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 448–456. ACM.

Wang, H.; Wang, N.; and Yeung, D.-Y. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1235–1244. ACM.

Wang, H.; Xingjian, S.; and Yeung, D.-Y. 2016. Collaborative recurrent autoencoder: recommend while learning to fill in the blanks. In *Advances in Neural Information Processing Systems*, 415–423.

Wu, Y.; DuBois, C.; Zheng, A. X.; and Ester, M. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, 153–162. ACM.

Xiaoyan Cai, J. H., and Yang, L. 2018. Generative adversarial network based heterogeneous bibliographic network representation for personalized citation recommendation. In *AAAI*.

Zhang, Y.; Ai, Q.; Chen, X.; and Croft, W. B. 2017. Joint representation learning for top-n recommendation with heterogeneous information sources. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 1449–1458. ACM.

Zhang, S.; Yao, L.; and Sun, A. 2017. Deep learning based recommender system: A survey and new perspectives. *arXiv preprint arXiv:1707.07435*.

Zhu, T.; Harrington, P.; Li, J.; and Tang, L. 2014. Bundle recommendation in ecommerce. In *SIGIR*, 657–666. ACM.