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 Chen2,3
1The Key Laboratory of Big Data Intelligent Computing of Zhejiang Province, China
2CAD & CG State Key Lab, Zhejiang University, China
3College 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. 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-
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 inferring $m$ 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 $D = \{(i, p_i^j, \rho_i^j)\}_{i=1}^m$ where $\rho_i^j$ contains all items labeled with “relevant” by user $i$ while $\rho_i$ 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 $\rho_i^j$ and lower scores to items in $\rho_i \setminus \rho_i^j$. Formally, the learning process can be expressed as follows (Christakopoulou and Banerjee 2015).

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

where $\theta_f$ 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

$$f(i,j,doc_j) = p_i^j (q_j + g(doc_j)) \quad (2)$$

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.

### 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 $\{v_{sent}^i\}_{i=1}^{I_{sent}}$ and topic vectors $\{v_{topic}^i\}_{i=1}^{I_{topic}}$. We make two assumptions. Assumption I: document vector $v_{doc}$ is equal to the average of topic vectors $v_{topic}^i$.

$$v_{doc} = \frac{1}{I_{topic}} \sum_{i=1}^{I_{topic}} v_{topic}^i \quad (3)$$

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

$$v_{topic}^i = \sum_{i=1}^{I_{sent}} \alpha_{p,i} W_p^{i,sent} \quad (4)$$

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

<table>
<thead>
<tr>
<th>Table 1: Table of notations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_{topic}$</td>
</tr>
<tr>
<td>$I_{sent}$</td>
</tr>
<tr>
<td>$I_{word}$</td>
</tr>
<tr>
<td>$p_i, \mathbf{P}$</td>
</tr>
<tr>
<td>$q_j, \mathbf{Q}$</td>
</tr>
<tr>
<td>$\mathbf{W}_p$</td>
</tr>
<tr>
<td>$\mathbf{W}_{word}$</td>
</tr>
<tr>
<td>$\mathbf{v}_{doc}, g(doc_j)$</td>
</tr>
<tr>
<td>$v_{topic}^i$</td>
</tr>
<tr>
<td>$v_{sent}^i, \mathbf{v}_{sent}$</td>
</tr>
<tr>
<td>$v_{word}, \mathbf{v}_{word}$</td>
</tr>
<tr>
<td>$\alpha_{i}^{\text{attent}}$</td>
</tr>
<tr>
<td>$\mathbf{h}_{\text{GRU}}$</td>
</tr>
<tr>
<td>$\theta_{\text{cont}}$</td>
</tr>
<tr>
<td>$\theta_{\text{dec}}$</td>
</tr>
<tr>
<td>$\theta_{\text{GRU}}$</td>
</tr>
<tr>
<td>$\alpha_{p,i}$</td>
</tr>
<tr>
<td>$r$</td>
</tr>
<tr>
<td>$\hat{\partial}_x f(x)$</td>
</tr>
</tbody>
</table>
Remark 1 We summarize the Doc2Vec protocol in Alg. 1. “query vector” is formed by a dynamic routing method. Therefore, the Doc2Vec protocol shares 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 $h_p^{(t)} = \begin{cases} 0 & t = 0 \\ \sum_i \exp(h_p^{(t-1)} v_i^{sent}) W_{p,v_i^{sent}}^\top & 0 < t \leq r \end{cases}$ (7)

We summarize the Doc2Vec protocol in Alg. 1.

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

$$\begin{align*}
\hat{v}_p^{doc} &= \frac{1}{I_{topic}} \sum_{i=1}^{I_{topic}} v_i^{topic} \\
v_p^{topic} &= V \text{softmax}(K^\top W_p^\top q_p), 1 \leq p \leq I_{topic} \\
K &= (v_1^{sent}, v_2^{sent}, \ldots, v_I^{sent}) \text{ is the “key matrix”,}
q_p &= h_p^{(t-1)} \text{ is the query vector and } V = W_p K \text{ is the “value matrix”. Therefore, the Doc2Vec protocol shares the same form of multi-head attention mechanism (Vaswani et al. 2017). The difference between them is that the proposed “query vector” is formed by a dynamic routing method.}
\end{align*}$$

Algorithm 1 Doc2Vec $v_i^{sent}, v_2^{sent}, \ldots, v_I^{sent}$

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: $v_p^{doc} = \frac{1}{I_{topic}} \sum_{i=1}^{I_{topic}} v_i^{topic}$
6: return $v_p^{doc}$

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

$$v_i^{sent} = \sum_{i=1}^{I_{word}} \alpha_i^{atten} h_i^{gru}$$

where $h_i^{gru}$ is the hidden state of a GRU and $\alpha_i^{atten}$ 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.

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

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

$$\begin{align*}
z &= \sigma(W_z v + U_z h + b_z) \\
r &= \sigma(W_r v + U_r h + b_r) \\
h' &= (1 - z) * h + z * (\tanh(W_h v + U_h (r * h) + b_h))
\end{align*}$$
with matrix $W_z, W_r, W_h$, matrix $U_z, U_r, U_h$, and vector $b_z, b_r, b_h$ being the GRU’s parameters. The attention signal $\alpha^{\text{atten}} = (\alpha_1^{\text{atten}}, \ldots, \alpha_{|\text{word}|}^{\text{atten}})^T$ is given by the following equation

$$
\alpha^{\text{atten}} = \text{softmax}(H_{\text{GRU}}^W h_{\text{word}}^{\text{gru}}) \quad (11)
$$

where matrix $H_{\text{GRU}}$ contains the hidden states of GRU cells.

**Word2Vec Layer** Let $w$ be a word belonging to the vocabulary of articles. The vector representation $v_{\text{word}}$ of word $w$ is given by

$$
v_{\text{word}} = W_{\text{word}}(w,:) \quad (12)
$$

where $W_{\text{word}}$ 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 $\{v^{\text{doc}}\}$ back into corpus $\{\text{doc}\}$ is a natural choice.

**Algorithm 2 Alternate Gradient Method**

**Require:** $P^{(0)}, Q^{(0)}, \theta^{(0)}_{\text{cont}}, W_{\text{dec}}^{(0)}, \theta^{(0)}_{\text{gru}}$

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

2. Compute stochastic gradients

   $$
g_P = \partial P_{\text{rank}}(P^{(t)}, Q^{(t)}, \theta^{(t)}_{\text{cont}})$$

   $$
g_Q = \partial Q_{\text{rank}}(P^{(t)}, Q^{(t)}, \theta^{(t)}_{\text{cont}})$$

   $$
g_{\theta_{\text{cont}}} = \partial \theta_{\text{cont}} L_{\text{rank}}(P^{(t)}, Q^{(t)}, \theta^{(t)}_{\text{cont}})$$

   $$
g_{\theta_{\text{gru}}} = \partial \theta_{\text{gru}} L_{\text{dec}}(\theta^{(t)}_{\text{cont}}, W_{\text{dec}}^{(t)}, \theta^{(t)}_{\text{gru}})$$

3. Perform stochastic gradient descent with stepsizes $\eta_t, \delta_t$

   $$
P^{(t+1)} = P^{(t)} - \eta_t g_P$$

   $$
Q^{(t+1)} = Q^{(t)} - \eta_t g_Q$$

   $$
\theta^{(t+1)}_{\text{cont}} = \theta^{(t)}_{\text{cont}} - \eta_t g_{\theta_{\text{cont}}} - \delta_t g_{\theta_{\text{gru}}}$$

   $$
W_{\text{dec}}^{(t+1)} = W_{\text{dec}}^{(t)} - \delta_t g_{W_{\text{dec}}}$$

4. Perform stochastic gradient ascent with stepsize $\delta_t$

   $$
\theta_{\text{gru}}^{(t+1)} = \theta_{\text{gru}}^{(t)} + \delta_t g_{\theta_{\text{gru}}}$$

5. if $\|\theta_{\text{gru}}^{(t)} - \theta^{(t)}_{\text{gru}}\| > \gamma$ then

6. $\theta_{\text{gru}}^{(t+1)} = \theta_{\text{gru}}^{(t)} + \gamma (\theta_{\text{gru}}^{(t+1)} - \theta^{(t)}_{\text{gru}}) / \| \theta_{\text{gru}}^{(t+1)} - \theta^{(t)}_{\text{gru}} \|

7. end if

8. end for

**Adversarial Training**

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

$$
\min_{P, Q, \theta_{\text{cont}}, \theta_{\text{dec}}} L_{\text{rank}}(P^{(0)}, Q^{(0)}, \theta^{(0)}_{\text{cont}}) + \lambda L_{\text{dec}}(\theta^{(0)}_{\text{cont}}, \theta^{(0)}_{\text{dec}}).
$$

(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 $L_{\text{dec}}(\cdot)$ may result from the strong fitting power of GRU other than the generalization power of the learned feature $v^{\text{doc}}$. To rule out the optimistic bias induced by GRU, we modify the composite loss as follows.

$$
\min_{P, Q, \theta_{\text{cont}}, W_{\text{dec}}} \frac{1}{T} \sum_{t=0}^{T} L_{\text{rank}}(P, Q, \theta_{\text{cont}}) + \lambda \max_{\theta_{\text{gru}}, \|\theta_{\text{gru}} - \theta'\| \leq \gamma} L_{\text{dec}}(\theta_{\text{cont}}, W_{\text{dec}}, \theta_{\text{gru}}).
$$

(16)

where $\theta_{\text{gru}}$ is the parameters of the GRU, $\theta'$ is a randomly generated vector and $\gamma$ 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 $v^{\text{doc}}$. 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 \( N_5 \) items without bookmarks (or ratings) as “not relevant”. \( N_5 \) 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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#users</th>
<th>#items</th>
<th>#feedbacks</th>
<th>avg. words</th>
</tr>
</thead>
<tbody>
<tr>
<td>CiteULike</td>
<td>5551</td>
<td>16980</td>
<td>210504</td>
<td>204.9</td>
</tr>
<tr>
<td>MovPlot1M</td>
<td>6040</td>
<td>3861</td>
<td>996045</td>
<td>82.19</td>
</tr>
<tr>
<td>MovPlot10M</td>
<td>13975</td>
<td>10681</td>
<td>1962580</td>
<td>84.66</td>
</tr>
</tbody>
</table>

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 \( N_5 = 6 \). 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 \( \lambda \) 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 \( d_{doc} \) 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. \( v_{doc} = \frac{1}{I_{word}} \sum_{i=1}^{I_{word}} w_i \). 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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>MAP</th>
<th>AUC</th>
<th>Pre@5</th>
<th>Pre@10</th>
<th>Rec@5</th>
<th>Rec@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>CiteULike</td>
<td>0.1340</td>
<td>0.9040</td>
<td>0.1496</td>
<td>0.1218</td>
<td>0.1131</td>
<td>0.1724</td>
</tr>
<tr>
<td>Word2Vec</td>
<td>0.1589</td>
<td>0.9283</td>
<td>0.1730</td>
<td>0.1374</td>
<td>0.1317</td>
<td>0.1930</td>
</tr>
<tr>
<td>CAMO</td>
<td>0.1623</td>
<td>0.9540</td>
<td>0.1779</td>
<td>0.1414</td>
<td>0.1357</td>
<td>0.2019</td>
</tr>
<tr>
<td>MovPlot1M</td>
<td>0.2145</td>
<td>0.9168</td>
<td>0.3973</td>
<td>0.3418</td>
<td>0.0892</td>
<td>0.1425</td>
</tr>
<tr>
<td>Word2Vec</td>
<td>0.2199</td>
<td>0.9215</td>
<td>0.4059</td>
<td>0.3468</td>
<td>0.0911</td>
<td>0.1474</td>
</tr>
<tr>
<td>Sent2Vec</td>
<td>0.2439</td>
<td>0.9296</td>
<td>0.4310</td>
<td>0.3701</td>
<td>0.0995</td>
<td>0.1629</td>
</tr>
<tr>
<td>CAMO</td>
<td>0.2516</td>
<td>0.9737</td>
<td>0.3948</td>
<td>0.3360</td>
<td>0.1122</td>
<td>0.1985</td>
</tr>
</tbody>
</table>

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.

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

(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

<table>
<thead>
<tr>
<th>CAMO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Scalable collaborative filtering using cluster based smoothing</td>
</tr>
<tr>
<td>2. Analysis of recommendation algorithms for e-commerce</td>
</tr>
<tr>
<td>3. Applying collaborative filtering techniques to movie search for</td>
</tr>
<tr>
<td>better ranking and browsing</td>
</tr>
<tr>
<td>4. Slope one predictors for online rating based collaborative</td>
</tr>
<tr>
<td>filtering</td>
</tr>
<tr>
<td>5. Collaborative filtering by personality diagnosis a hybrid</td>
</tr>
<tr>
<td>memory and model based approach</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CMLST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Empirical analysis of predictive algorithms for collaborative</td>
</tr>
<tr>
<td>filtering</td>
</tr>
<tr>
<td>2. Item based collaborative filtering recommendation algorithms</td>
</tr>
<tr>
<td>3. Improving recommendation lists through topic diversification</td>
</tr>
<tr>
<td>4. Recommender systems</td>
</tr>
<tr>
<td>5. Toward the next generation of recommender systems a survey of</td>
</tr>
<tr>
<td>the state-of-the-art and possible extensions</td>
</tr>
</tbody>
</table>

(b) Queried paper: Towards real-time community detection in large networks

<table>
<thead>
<tr>
<th>CAMO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Random hypergraphs and their applications</td>
</tr>
<tr>
<td>2. A comparative study of social network models network evolution</td>
</tr>
<tr>
<td>models and nodal attribute models</td>
</tr>
<tr>
<td>3. Line graphs link partitions and overlapping communities</td>
</tr>
<tr>
<td>4. Statistical significance of communities in networks</td>
</tr>
<tr>
<td>5. Emergence of communities in weighted networks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CMLST</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Line graphs link partitions and overlapping communities</td>
</tr>
<tr>
<td>2. Statistical significance of communities in networks</td>
</tr>
<tr>
<td>3. Random hypergraphs and their applications</td>
</tr>
<tr>
<td>4. Adaptive networks coevolution of disease and topology</td>
</tr>
<tr>
<td>5. The energy landscape of social balance</td>
</tr>
</tbody>
</table>

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


