Co-Attentive Multi-Task Learning for Explainable Recommendation

Zhongxia Chen[1,2], **Xiting Wang**[2*], **Xing Xie**[2], **Tong Wu**[3], **Guoqing Bu**[3], **Yining Wang**[3] and **Enhong Chen**[1]
[1]School of Computer Science and Technology, University of Science and Technology of China, China
[2]Microsoft Research Asia, China
[3]CFETS Information Technology (Shanghai) Co., Ltd., China
{czx87@mail., cheneh@}ustc.edu.cn, {xitwan, xing.xie}@microsoft.com,
{wutong, buguoqing_zh, wangyining_zh}@chinamoney.com.cn

Abstract

Despite widespread adoption, recommender systems remain mostly black boxes. Recently, providing explanations about why items are recommended has attracted increasing attention due to its capability to enhance user trust and satisfaction. In this paper, we propose a co-attentive multi-task learning model for explainable recommendation. Our model improves both prediction accuracy and explainability of recommendation by fully exploiting the correlations between the recommendation task and the explanation task. In particular, we design an encoder-selector-decoder architecture inspired by human's information-processing model in cognitive psychology. We also propose a hierarchical co-attentive selector to effectively model the cross knowledge transferred for both tasks. Our model not only enhances prediction accuracy of the recommendation task, but also generates linguistic explanations that are fluent, useful, and highly personalized. Experiments on three public datasets demonstrate the effectiveness of our model.

1 Introduction

Personalized recommendation has become a major technique for helping users handle huge amounts of online content. To improve user experience, it is essential that the recommendation model accurately predicts users' personal preferences for items. Except for **accuracy**, there is a growing interest in **explainability** [Alvarez-Melis and Jaakkola, 2018]. Evidence has shown that explanations about why the items are recommended can increase user trust, improve satisfaction, and persuade the users to buy or try an item [Rago *et al.*, 2018].

The recent attention on explainability has lead to the development of a series of explainable recommendation models. A fundamental question explainable recommendation aims to answer is how we balance accuracy and explainability. Most existing methods consider the two goals in separate steps or only focus on one of the goals, which limits their effectiveness. In particular, let us consider the two most widely used frameworks: post-hoc and embedded.

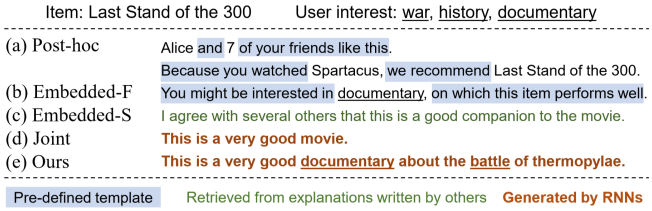


Figure 1: Explanations provided by different methods: (a) post-hoc methods [Sharma and Cosley, 2013; Peake and Wang, 2018]; (b) a feature-level embedded method [Zhang *et al.*, 2014]; (c) a sentence/review-level embedded method [Chen *et al.*, 2018]; (d) a joint model [Li *et al.*, 2017]; and (e) our model. Underlined words are related with user interest. Our model generates informative and personalized explanations that are relevant to the item.

Post-hoc methods explain a black-box model after it is trained [Vig *et al.*, 2009; Sharma and Cosley, 2013; Peake and Wang, 2018]. They consider accuracy and explainability in separate steps and provide explanations based on the model outputs. Typically, the rich information embedded inside the recommendation models are ignored and the explanations are selected from a set of pre-defined templates. As shown in Fig. 1(a), the templates are often readable and persuasive. However, the explanations may not reflect the model's actual reasoning and the diversity of the explanations are limited.

Embedded methods integrate the explanation process into the construction of the recommendation model [McAuley and Leskovec, 2013; Zhang *et al.*, 2014; He *et al.*, 2015; Chen *et al.*, 2018; Wang *et al.*, 2018a]. These methods focus on recommendation accuracy. Their explanations consist of features or sentences that are important for improving accuracy (Fig. 1(b)(c)). Since explainability is not included in the optimization goal, it is difficult to guarantee the quality of the explanations, e.g., consistency [Wang *et al.*, 2018b]. Moreover, the embedded methods are retrieval-based (i.e., selecting from original dataset). They may fail to provide a highly personalized explanation when data is sparse and may suffer from legal issues (e.g., copyright) in certain scenarios.

The purpose of this paper is to illustrate how to effectively optimize accuracy and explainability in a joint and unified framework. The key idea is that fully exploiting the correlations between the recommendation task and the explanation task potentially enables both tasks to be better off than when they are considered separately. While the idea seems

*Xiting Wang is the corresponding author

promising, achieving this objective is challenging. A simple multi-task learning framework that only shares latent representations such as user and item embeddings between the two tasks cannot achieve desirable results. As shown in Fig. 1(d), the generated explanations are usually quite general, i.e., they do not contain specific information about key features of the item. This is because 1) the shared representations are not explainable and fail to provide explicit constraints on the explanations and 2) user and item embeddings do not contain sufficient information about deep user-item interactions.

To address the aforementioned issues, we propose a **C**o-**A**ttentive **M**ulti-task **L**earning (**CAML**)[1] model for explainable recommendation, which enhances both accuracy and explainability of explainable recommendation by tightly coupling the recommendation task and the explanation task.

The major contributions of this paper are:

First, we **design an encoder-selector-decoder architecture for multi-task learning**. Our architecture design is inspired by models in cognitive psychology. Cognitive scientists have shown that a human's cognitive process consists of three major sub-processes [Lang, 2000]. The three sub-processes correspond to our encoder, selector, and decoder. In explainable recommendation, the decoder is responsible for deciding the predicted rating (recommendation task) and generating the explanations (explanation task). The selector serves as the transferred cross knowledge for both tasks. The architecture tightly couples the two tasks in a natural way.

Second, we **propose a hierarchical co-attentive selector to effectively control the cross knowledge transfer for both tasks**. The selector models deep level interactions between the users and items. In particular, it identifies reviews and concepts (cross knowledge) that are important for the user-item pair based on co-attention. Compared with traditional methods such as REINFORCE [Williams, 1992], our model does not suffer from slow convergence and high variance because we use hierarchical multi-pointer networks [Vinyals *et al.*, 2015] for review and concept selection.

Finally, extensive experiments demonstrate that **our method improves both explainability and accuracy**. As shown in Fig. 1(e), our method is able to generate fluent explanations that are informative, highly personalized, and relevant with the item. Numerical experiments demonstrate that the explanations of CAML increases BLEU scores by 14.6% to 35.9%. Human evaluation shows that our explanations are more fluent and much more useful compared with the state-of-art generative model (**explainability**). Moreover, CAML consistently improves recommendation accuracy on three public datasets compared with 7 baselines (**accuracy**).

In the rest of the paper, we first introduce the problem definition and our model. We then describe the experiments and conclusion. The **related work** section is omitted because most existing works have been discussed in the introduction.

2 Problem Formulation

We formulate our problem as follows.

Input. The input of our model consists of the user set U, the item set V, the reviews, and the concepts in the reviews:

- Each **user** is represented by its ID $u \in U$ and each **item** is denoted by the item ID $v \in V$.
- The **reviews** of a user u are $(\mathcal{D}_{u,1}, ..., \mathcal{D}_{u,l_d})$, where l_d denotes the maximum number of reviews. Each review $\mathcal{D}_{u,i}$ is denoted by a set of words in the review. Similarly, $(\mathcal{D}_{v,1}, ..., \mathcal{D}_{v,l_d})$ represents the reviews of item v.
- The **concepts** are a subset of words that correspond to important explicit features mentioned in the review. For example, the review "*This is a great little comedy with a catchy song*" contains two concepts: *comedy* and *song*. We derive the concepts by utilizing Microsoft Concept Graph[2] [Wu *et al.*, 2012; Wang *et al.*, 2015], a widely used knowledge graph with about 18 million concepts or instances. We map the n-grams in the reviews to concepts or instances in the concept graph and filter concepts that are rarely used or not informative.

Output. Given a user-item pair (u, v), our model predicts:

- The **rating** r that reflects how much u likes v.
- The **linguistic explanation** $Y = (y_1, y_2, ..., y_T)$ that illustrates why user u likes or does not like item v. Here y_k denotes the k-th word in the explanation. During training, ground truth explanations can be set to reviews or tips the users write for the corresponding item [Li *et al.*, 2017].

3 Model Description

In this section, we introduce CAML, a **C**o-**A**ttentive **M**ulti-task **L**earning model for explainable recommendation. Our model improves both prediction accuracy and explainability by fully exploiting the correlations between the recommendation task and explanation task. To achieve these goals, we design an **encoder-selector-decoder** architecture. As shown in Fig. 2, our model consists of three components, each corresponds to one major sub-process of the information-processing model in cognitive psychology [Lang, 2000].

- In the **encoder**, we embed the words, reviews, and the implicit factors of users and items. The encoder corresponds to the sub-process of *encoding* in information processing.
- The **multi-pointer co-attention selector** identifies reviews and concepts that are important for both users and items by leveraging hierarchical multi-pointer co-attention. The identified concepts serve as the cross knowledge for the two tasks. This component corresponds to the *storage* sub-process in information processing, which distinguishes important pieces of encoded information and stores them properly through associations and links (i.e., networks).
- The **multi-task decoder** is responsible for predicting the ratings and generating the explanations based on the key concepts extracted by the selector. In information processing, this corresponds to the *retrieval* sub-process, which reactivates important information for decision making.

Next, we will introduce each component and how we jointly optimize them in an end-to-end framework.

3.1 Encoder

Word encoder. We use a look up layer to transform a word ID into a word embedding $\boldsymbol{\omega} \in \mathcal{R}^{l_w}$. Here, l_w is the dimensionality of word representations.

[1] Source code: https://github.com/3878anonymous/CAML.

[2] https://concept.research.microsoft.com/

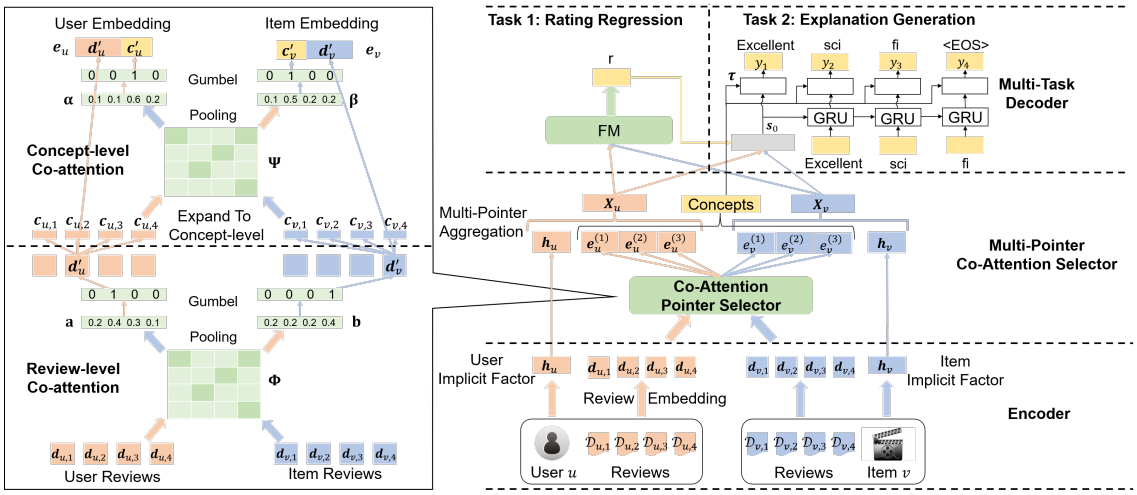


Figure 2: Framework of the proposed model for explainable recommendation.

Review encoder. We calculate an initial review embedding by following [Tay *et al.*, 2018]. Given review $\mathcal{D}_{u,i}$, the corresponding review embedding $\mathbf{d}_{u,i}$ can be calculated by $\mathbf{d}_{u,i} = \sum_{\boldsymbol{\omega} \in \mathcal{D}_{u,i}} \boldsymbol{\omega}$. Summing up the embeddings of related words to derive an initial review embedding balances both effectiveness and efficiency.

User/Item implicit factor encoder. Explicit factors such as reviews may not contain all information about a user or an item. To complement explicit factors, we introduce implicit user or item factors by following [Zhang *et al.*, 2014]. Specifically, we use a look up layer to transform user u (or item v) into implicit representation $\mathbf{h}_u \in \mathcal{R}^{l_u}$ (or $\mathbf{h}_v \in \mathcal{R}^{l_v}$).

3.2 Multi-Pointer Co-Attention Selector

We model the cross knowledge transferred for the two tasks by identifying (selecting) reviews and concepts that are important for user-item pairs. Our method is based on the multi-pointer co-attention networks [Tay *et al.*, 2018], which 1) are effective in modeling a deeper level of pairwise interactions and 2) have better convergence properties compared with alternatives such as REINFORCE [Williams, 1992]. We first show how we extend the networks to hierarchically select reviews and then concepts. Next, we discuss how we support selection and aggregation of multiple reviews and concepts.

Review-level co-attention pointer. Given user review embeddings $\mathbf{d}_{u,1}, ..., \mathbf{d}_{u,l_d}$ and item review embeddings $\mathbf{d}_{v,1}, ..., \mathbf{d}_{v,l_d}$, we first calculate co-attention weight matrix $\mathbf{\Phi} \in \mathcal{R}^{l_d \times l_d}$ among user-item review pairs. Specifically, the (i, j)-th entry of $\mathbf{\Phi}$ is computed by:

$$\phi_{i,j} = F(\mathbf{d}_{u,i})^{\mathrm{T}} \mathbf{W}_d F(\mathbf{d}_{v,j}) \tag{1}$$

Here, $\mathbf{W}_d \in \mathcal{R}^{l_w \times l_w}$ is the weight matrix and F is a feed-forward neural network with l_F layers.

Max pooling over the co-attention matrix selects reviews with the maximum correlations with all reviews:

$$a_i = \max_{j=1,...,l_d} \phi_{i,j} \quad \text{and} \quad b_j = \max_{i=1,...,l_d} \phi_{i,j} \tag{2}$$

To turn the unnormalized vectors $\mathbf{a} = (a_1, ..., a_{l_d})$ and $\mathbf{b} = (b_1, ..., b_{l_d})$ into a probability distribution, the most commonly used function is softmax. However, selecting reviews (i.e., applying the $\arg\max$ function) from the softmax distribution makes the model non-differentiable. To address this issue, the Straight-Through Gumbel-Softmax function [Jang *et al.*, 2017] is used for attention learning:

$$q_i = \frac{\exp(\frac{a_i + g_i}{\tau})}{\sum_{j=1}^{n_m} \exp(\frac{a_j + g_i}{\tau})} \tag{3}$$

where g_i is the Gumbel noise, τ is the temperature parameter which controls the smoothness of the vector $\mathbf{q}$. If τ is close to zero, $\mathbf{q}$ becomes close to a one-hot vector. In the forward pass, $\arg\max$ is leveraged to gain a pointer $\mathbf{z}$ from $\mathbf{q}$:

$$z_i = \begin{cases} 1, & i = \arg\max_j(q_j), \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

We denote this function as $Gumbel(\mathbf{a}) = \mathbf{z}$.

In the backward pass, the function approximates the gradients of $\mathbf{z}$ by using the gradients of $\mathbf{q}$ to ensure end-to-end model training [Jang *et al.*, 2017]. The review-level co-attentional user embedding $\mathbf{d}'_u$ and item embedding $\mathbf{d}'_v$ are:

$$\mathbf{d}'_u = (Gumbel(\mathbf{a}))^{\mathrm{T}} \mathbf{D}_u \ \text{ and } \ \mathbf{d}'_v = (Gumbel(\mathbf{b}))^{\mathrm{T}} \mathbf{D}_v \tag{5}$$

Here, the i-th row of $\mathbf{D}_u \in \mathcal{R}^{l_d \times l_w}$ (or $\mathbf{D}_v$) is $\mathbf{d}_{u,i}^{\mathrm{T}}$ (or $\mathbf{d}_{v,i}^{\mathrm{T}}$).

Concept-level co-attention pointer. Suppose that the selected user review contains concepts $\mathbf{c}_{u,1}, ..., \mathbf{c}_{u,l_c}$. Here $\mathbf{c}_{u,i}$ denotes the word embedding of the i-th selected concept. The concepts of the selected item review are $\mathbf{c}_{v,1}, ..., \mathbf{c}_{v,l_c}$. The concept co-attention matrix $\mathbf{\Psi} \in \mathcal{R}^{l_c \times l_c}$ is calculated by

$$\psi_{i,j} = F(\mathbf{c}_{u,i})^{\mathrm{T}} \mathbf{W}_c F(\mathbf{c}_{v,j}) \tag{6}$$

where $\mathbf{W}_c \in \mathcal{R}^{l_w \times l_w}$ is the weight matrix. Next, we calculate the weights of the concepts by aggregating $\mathbf{\Psi}$. We investigate different pooling strategies and find mean pooling to perform the best. Thus, mean pooling is used to calculate the weights of the concepts $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$:

$$\alpha_i = \frac{1}{l_c} \sum_{j=1,...,l_c} \psi_{i,j} \quad \text{and} \quad \beta_j = \frac{1}{l_c} \sum_{i=1,...,l_c} \psi_{i,j} \tag{7}$$

Here, α_i is the i-th entry of $\boldsymbol{\alpha}$ and β_j is the j-th entry of $\boldsymbol{\beta}$. The concept-level co-attentional embeddings are:

$$\mathbf{c}'_u = (Gumbel(\boldsymbol{\alpha}))^{\mathrm{T}}\mathbf{C}_u \text{ and } \mathbf{c}'_v = (Gumbel(\boldsymbol{\beta}))^{\mathrm{T}}\mathbf{C}_v \quad (8)$$

where the i-th row of $\mathbf{C}_u \in \mathcal{R}^{l_c \times l_w}$ (or $\mathbf{C}_v$) is $\mathbf{c}_{u,i}^{\mathrm{T}}$ (or $\mathbf{c}_{v,i}^{\mathrm{T}}$).

Multi-pointer aggregation. Since a user may consider multiple reviews and concepts when s/he provides ratings or reviews, we need to support selection and aggregation of multiple pointers. Consider the co-attentional user embedding $\mathbf{e}_u = [\mathbf{d}'_u; \mathbf{c}'_u]$ that combines review-level and concept-level user embedding. In the multi-pointer setting, we run our hierarchical co-attention pointer selector multiple times with different Gumbel noises to get multiple samples: $\{\mathbf{e}_u^{(1)}, ..., \mathbf{e}_u^{(n_p)}\}$. Similarly, we obtain a set of samples for the co-attentional item embedding: $\{\mathbf{e}_v^{(1)}, ..., \mathbf{e}_v^{(n_p)}\}$. We then use a non-linear layer to aggregate multiple samples:

$$\bar{\mathbf{e}}_u = \sigma(\mathbf{W}_p[\mathbf{e}_u^{(1)}, ..., \mathbf{e}_u^{(n_p)}] + \mathbf{b}_p) \quad (9)$$

$$\bar{\mathbf{e}}_v = \sigma(\mathbf{W}_p[\mathbf{e}_v^{(1)}, ..., \mathbf{e}_v^{(n_p)}] + \mathbf{b}_p) \quad (10)$$

where $\mathbf{W}_p \in \mathcal{R}^{l_p \times 2n_p l_w}$ and $\mathbf{b}_p \in \mathcal{R}^{l_p}$. The aggregated co-attentional user and item embeddings are used in both rating prediction and explanation generation.

3.3 Multi-Task Decoder

We propose a multi-task decoder that predicts ratings and generates explanations based on representations learned in the encoder and selector. We consider two types of user and item embeddings. The first type is the co-attentional embeddings $\bar{\mathbf{e}}_u$ and $\bar{\mathbf{e}}_v$, which are learned from explicit factors such as reviews. The second is implicit factor representations $\mathbf{h}_u$ and $\mathbf{h}_v$. To simultaneously capture an item's explicit and implicit factors, the user and item embeddings are expressed as $\mathbf{x}_u = [\bar{\mathbf{e}}_u; \mathbf{h}_u]$ and $\mathbf{x}_v = [\bar{\mathbf{e}}_v; \mathbf{h}_v]$.

Rating Prediction. Factorization machine (FM) [Rendle, 2010] is used as the rating prediction method to model pairwise interactions between different features:

$$r = f(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n}\sum_{j=i+1}^{n} \langle \mathbf{m}_i, \mathbf{m}_j \rangle x_i x_j \quad (11)$$

Here, $x_i \in \mathcal{R}$ is the i-th entry of $\mathbf{x} = [\mathbf{x}_u; \mathbf{x}_v]$, $w_0 \in \mathcal{R}$ is the bias, $w_i \in \mathcal{R}$ and $\mathbf{m}_i \in \mathcal{R}^k$ are parameters to be learned. The loss function of our rating prediction task is formulated as:

$$\mathcal{L}_r = \frac{1}{2|\Omega|} \sum_{(u,v)\in\Omega} (r - r_*)^2 \quad (12)$$

where Ω represents the training set and r_* is the corresponding ground truth rating.

Explanation Generation. We generate explanation Y based on user embedding $\mathbf{x}_u$, item embedding $\mathbf{x}_v$, predicted rating r, and concepts chosen in the selector. We first show how we use Gated Recurrent Unit (GRU) [Chung *et al.*, 2014] to translate $\mathbf{x}_u$, $\mathbf{x}_v$, and r into a sequence of words. Then, we introduce two related losses used in the training process.

GRU. GRU demonstrates high capability in text generation tasks. To incorporate $\mathbf{x}_u$, $\mathbf{x}_v$, and r into GRU, we compute the initial hidden state $\mathbf{s}_0$ by:

$$\mathbf{s}_0 = \tanh(\mathbf{W}_u\mathbf{x}_u + \mathbf{W}_v\mathbf{x}_v + \mathbf{W}_r\hat{\mathbf{r}} + \mathbf{b}_s) \quad (13)$$

Here, $\hat{\mathbf{r}}$ is a vector representation of r. $\hat{\mathbf{r}}$ is calculated by rounding r into an integer (e.g., $3.14 \rightarrow 3$) and converting it to a one-hot vector. $\mathbf{W}_u$, $\mathbf{W}_v$, $\mathbf{W}_r$, and $\mathbf{b}_s$ are parameters to be learned. The hidden state $\mathbf{s}_t$ at time t is calculated recursively:

$$\mathbf{s}_t = \mathrm{GRU}(\mathbf{s}_{t-1}, \boldsymbol{\omega}_{y_t}) \quad (14)$$

where $\boldsymbol{\omega}_{y_t}$ is the embedding of the word y_t generated at time t. This hidden state is fed into the output layer to generate the output word distribution $\mathbf{o}_t$ as time t:

$$\mathbf{o}_t = \zeta(\mathbf{W}_o\mathbf{s}_{t-1} + \mathbf{b}_o) \quad (15)$$

where $\mathbf{W}_o \in \mathcal{R}^{|\mathcal{V}|\times l_t}$, $\mathbf{b}_o \in \mathcal{R}^{|\mathcal{V}|}$, ζ is the softmax function and $|\mathcal{V}|$ is the vocabulary size. During testing, beam search is used to find the best explanation $Y = (y_1, ..., y_T)$ with the maximum log-likelihood $\sum_{t=1}^{T} \log o_{t,y_t}$.

Concept relevance loss $\mathcal{L}_c$. During training, $\mathcal{L}_c$ is used to increase the probability that the selected concepts appear in Y. Let us denote the selected concept vector as $\boldsymbol{\tau} \in \mathcal{R}^{|\mathcal{V}|}$. τ_k is 1 if the k-th word is a concept and has been selected by at least one pointer and is 0 otherwise. $\mathcal{L}_c$ is computed by:

$$\mathcal{L}_c = \frac{1}{|\Omega|} \sum_{(u,v)\in\Omega} \sum_{t=1}^{T} (\max_k \ (-\tau_k \log o_{t,k})) \quad (16)$$

Negative log-likelihood loss $\mathcal{L}_n$. $\mathcal{L}_n$ is widely used to ensure that the generated words are similar to the ground truth ones. Let y_{t*} denote the ground truth word at time t, we have

$$\mathcal{L}_n = \frac{1}{|\Omega|} \sum_{(u,v)\in\Omega} \sum_{t=1}^{T} (-\log o_{t,y_{t*}}) \quad (17)$$

3.4 Joint Learning

Different types of loss functions are linearly combined to jointly learn two tasks in an end-to-end manner:

$$\mathcal{L} = \mathcal{L}_r + \lambda_c\mathcal{L}_c + \lambda_n\mathcal{L}_n + \lambda_l\|\Theta\|_2^2 \quad (18)$$

where λ_c, λ_n, and λ_l are weights that balance the importance of different losses. Θ contains all model parameters. We choose Adam [Kingma and Ba, 2015] as the training optimizer because it empirically outperforms other stochastic optimization methods.

4 Experiments

4.1 Experimental Settings

Datasets. Three publicly available datasets from different domains are used in our evaluation:

- **Electronics** is the part of Amazon dataset[3] that focuses on electronic products. We use the 5-core version where users and items have no fewer than 5 reviews.

	Electronics	Movies&TV	Yelp-2016
Users	192,403	123,960	677,379
Items	63,001	50,052	84,693
Reviews	1,688,117	1,697,533	2,530,843
Concepts	652	791	1,004

Table 1: Statistics of three public datasets.

[3] http://jmcauley.ucsd.edu/data/amazon/

Datasets	Criteria	Retrieval			Generative	Ours			Improvement (%)	
		LexRank	NARRE	RLRec	NRT	CAML-G	CAML-C	CAML	$\Delta_{Retrieval}$	$\Delta_{Generative}$
Electronics	BLEU	1.44	1.45	1.45	1.33	1.79	1.92	**1.97**	+35.9	+48.1
	ROUGE-1	14.22	15.19	11.12	17.39	18.64	19.00	**19.26**	+26.8	+10.8
	ROUGE-2	3.60	3.29	1.60	3.50	3.63	3.78	**3.81**	+5.8	+8.9
	ROUGE-L	13.70	13.28	9.70	15.71	16.37	16.63	**16.75**	+22.3	+6.6
	ROUGE-SU4	4.38	5.25	3.13	5.97	6.26	6.43	**6.47**	+23.2	+8.4
Movies&TV	BLEU	1.78	1.75	1.73	1.60	1.94	**2.04**	**2.04**	+14.6	+27.5
	ROUGE-1	15.68	15.31	11.61	18.09	18.86	19.14	**19.32**	+23.2	+6.8
	ROUGE-2	2.45	3.62	3.84	4.30	4.48	4.43	**4.58**	+19.3	+6.5
	ROUGE-L	12.46	12.99	10.06	16.00	16.41	16.48	**16.69**	+28.5	+4.3
	ROUGE-SU4	5.24	5.79	4.98	6.29	6.52	6.57	**6.71**	+15.9	+6.7
Yelp	BLEU	0.97	1.13	1.13	1.31	1.50	1.57	**1.58**	+39.8	+20.6
	ROUGE-1	11.06	10.55	9.28	13.31	13.93	14.15	**14.24**	+28.8	+7.0
	ROUGE-2	2.42	2.66	1.93	3.05	3.42	3.47	**3.50**	+31.6	+14.8
	ROUGE-L	10.15	9.30	8.18	12.13	12.70	12.84	**12.90**	+27.1	+6.3
	ROUGE-SU4	3.58	3.85	3.10	4.60	4.92	5.00	**5.03**	+30.6	+9.3

Table 2: Evaluation results for explanation generation. CAML-G and CAML-C are two variants of our method. $\Delta_{Retrieval}$ and $\Delta_{Generative}$ denote the relative improvement of CAML over the most competitive retrieval baseline and the generative baseline.

- **Movies&TV** is also from the Amazon 5-core dataset. This dataset focuses on movies and TVs.
- **Yelp** contains restaurant reviews from Yelp Challenge 2016[4]. Compared with the first two datasets, the Yelp dataset is larger and much sparser.

For each dataset, we keep the top 40,000 most frequent words. We filter both rare concepts (occurring in less than 0.5% of reviews) and domain-dependent frequent concepts (e.g., *movie* and *film* in dataset Movies&TV). The statistics of the datasets are shown in Table 1.

Baselines. To evaluate **explainability**, we compare CAML with both retrieval methods and a generative method.

- **Retrieval**. Retrieval methods select linguistic explanations from review comments in the training set. Three retrieval methods are considered. The first is **Lexrank** [Erkan and Radev, 2004], a stochastic graph-based method for computing relative importance of textual units. The other baselines are **NARRE** [Chen *et al.*, 2018] and **RLRec** [Wang *et al.*, 2018b], which obtain the important sentences or reviews by employing the attention mechanism.
- **Generative**. Generative methods create explanations by using RNNs. The state-of-art generative method is **NRT** [Li *et al.*, 2017], which generates an explanation based on rating and the word-level distribution of the review.

To evaluate the **accuracy** of rating prediction, we compare CAML with two groups of baselines:

- **CF**. The first group consists of Collaborative Filtering (CF) methods. These methods require only the observed ratings for prediction. In particular, we consider three popular CF methods: **PMF** [Mnih and Salakhutdinov, 2008], **NMF** [Lee and Seung, 2001], and **SVD++** [Koren, 2008].
- **Neural**. The second group contains neural models that leverage review comments for rating prediction. Four state-of-the-art methods are considered: **MPCN** [Tay *et al.*, 2018], **NARRE**, **RLRec**, and **NRT**.

Evaluation criteria. To measure **explainability**, we leverage **BLEU** [Papineni *et al.*, 2002] and **ROUGE** [Lin, 2004], which are widely used in machine translation and natural language generation to evaluate the similarity between ground truth texts and generated texts [Li *et al.*, 2016; Li *et al.*, 2017]. Following [Li *et al.*, 2017], we set the ground truth explanations as the first sentences in the reviews. Li et al. also consider tips (short comments provided by users) as ground truth explanations. We ignore tips because most of them are not informative (e.g., "*This is a good movie*"). Note that the ground truth reviews ($\mathcal{D}_{u,v}$) will be excluded from the inputs ($\mathcal{D}_u$ and $\mathcal{D}_v$) to prevent leak of information. We use F-measures of ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-SU4 to evaluate the explanations in different granularities. Larger BLEU and ROUGE scores indicate better explainability.

Case 1. User interest: **horror**, **night**, **fun**	
NRT	I'll admit it.
CAML	I am a huge fan of **horror** movies, and this is one of my favorite movies.
Truth	Remember when **horror** movies were **fun**?
Case 2. User interest: **humor**, **scenery**, **main character**	
NRT	I love this series.
CAML	If you like british **humor**, you will love this series.
Truth	This is very british **humor**.
Case 3. User interest: **story line**, **cartoon**, **worth**	
NRT	I enjoyed this series as much as the first one.
CAML	I enjoyed this movie, the **animation** was great and the **story line** was very good.
Truth	Great **price** and the **animation** was cool.

Table 3: Explanations generated by NRT and CAML. "Truth" represents the ground truth explanation. **Bolded words** are concepts.

The **accuracy** of rating prediction is evaluated by employing Root Mean Square Error (**RMSE**) [Chen *et al.*, 2018], which measures the deviation between the predicted ratings r and the ground truth ratings r_*.

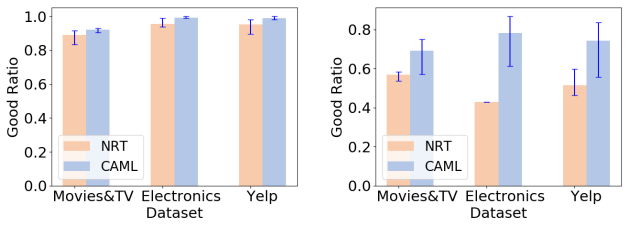
Implementation details. We implement CAML by using Tensorflow[5]. The models are trained on NVIDIA Tesla P100. We randomly choose 80% of samples as the training data, 10% as validation and 10% for testing on each dataset. We initialize the hyper-parameters for the baselines by following the corresponding paper and carefully tune them to ensure that they achieve the optimal performance. The initial learning rate is set to 10^{-3}. The number of pointers is tested in

[4]https://www.yelp.com/dataset/challenge

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

Datasets	CF			Neural Models				Ours			Improvement (%)	
	PMF	NMF	SVD++	MPCN	NARRE	RLRec	NRT	CAML-R	CAML-C	CAML	Δ_{CF}	Δ_{Neural}
Electronics	2.065	1.170	1.105	1.105	1.103	1.102	1.091	**1.085**	1.086	**1.085**	+1.8	+0.6
Movies&TV	1.250	1.089	1.013	1.001	0.999	1.012	0.990	0.990	**0.987**	**0.987**	+2.6	+0.3
Yelp	1.829	1.290	1.193	1.193	1.190	1.220	1.186	1.180	**1.173**	**1.173**	+1.7	+1.1

Table 4: Numerical results for rating prediction (RMSE). CAML-R and CAML-C are two variants of our model. Δ_{CF} and Δ_{Neural} represent the relative improvement of CAML over the most competitive CF baseline and neural baseline.



(a) Fluency (b) Usefulness

Figure 3: Human evaluation results on explanation quality.

[1, 2, 3, 4, 5] for all datasets. For model regularization, the dropout rate is set to 0.2 and the L2 regularization is a fixed number 10^{-6}. We set the rating prediction and the explanation generation as the same weight 1.0, and tune λ_c among [0.01, 0.05, 0.1, 0.2, 0.5].

A more comprehensive description of experimental settings can be found in the supplementary material[6].

4.2 Explainability Study

Overall performance. Table 2 shows that CAML consistently outperforms baselines in terms of BLEU and ROUGE scores on different datasets. Take BLEU as an example, CAML achieves 14.6% to 39.8% improvement over the most competitive retrieval method and obtains 20.6% to 48.1% improvement over the state-of-art generative method. This illustrates the effectiveness of our encoder-selector-decoder architecture and the multi-pointer co-attention selector. Our method consistently outperforms NRT because it learns deep user-item interactions and provides explicit constraints on the explanations to improve the explanation quality.

Human evaluation *w.r.t.* fluency and usefulness. With the help of a vendor company, we hire three experienced assessors. We sample 100 test cases from the three datasets and ask the assessors to evaluate if the generated sentences are fluent and useful for helping them decide whether they will try the item or not. The assessors do not know each other and are not aware of which explanation comes from which method (the order of the explanations are randomly shuffled). Fig. 3 shows our model outperforms the state-of-art generative method NRT in terms of both fluency and usefulness on all three datasets. Here, good ratio denotes the percentage of sentences that are considered fluent or useful. The Kappa scores for fluency and usefulness are 0.514 and 0.465, respectively, showing moderate agreements among assessors.

Case analysis. Table 3 shows examples of explanations generated by NRT and our method (Movies&TV dataset). For each user in the three cases, we display three concepts that s/he most frequently mentioned in the reviews (training set) to reveal her/his interest. This table shows that our model can select the concepts that fit the user interests, and generate explanations that contain one or multiple appropriate concepts. Compared with NRT, our explanations are informative, useful and similar with the ground truth explanations.

[6] The supplemental material is available at https://github.com/3878anonymous/CAML/blob/master/supplemental.pdf

4.3 Accuracy of Rating Prediction

The evaluation results of recommendation accuracy is shown in Table 4. In general, neural models outperform traditional CF methods. This is because that the neural models learn more effective representations of users and items through deeper architectures. Moreover, all neural models leverage additional information (i.e., reviews).

Our model consistently achieves the highest accuracy on all three datasets and outperforms NRT by 0.9% on average. This demonstrates the effectiveness of our encoder-selector-decoder architecture and our approach for modeling explicit user-item interactions through hierarchical co-attention.

4.4 Ablation Analysis

Effectiveness of multi-task learning. We compare our multi-task learning model with two variants that consider only one task. The first variant, CAML-G considers only explanation generation and removes the rating prediction loss $\mathcal{L}_r$ from Eq. (18). The second variant, CAML-R takes into account only rating prediction and removes explanation generation losses $\mathcal{L}_c$ and $\mathcal{L}_n$ from Eq. (18). Tables 2 and 4 show that our multi-task learning model achieves better performances than the single-task models. This suggests that sharing information between two tasks is beneficial.

Effectiveness of concept selection. To study the effectiveness of involving concepts, we evaluate the performance of CAML-C. In CAML-C, we remove concept-level co-attention pointer, set $\mathbf{e}_u$ to $\mathbf{d}'_u$, set $\mathbf{e}_v$ to $\mathbf{d}'_v$, and remove the concept relevant loss $\mathcal{L}_c$ from Eq. (18). From Tables 2 and 4 we can see that CAML-C consistently performs worse than or equal to CAML in terms of both explainability and accuracy. The results show the effectiveness of concept selection, especially for the explanation generation task.

5 Conclusion

In this paper, we propose a co-attentive multi-task learning model for explainable recommendation, which fully exploits the correlations between the recommendation task and the explanation task to improve both accuracy and explainability. Specifically, we propose an encoder-selector-decoder architecture that is inspired by the cognitive process of human decision making. A hierarchical co-attentive selector is designed to effectively model the cross knowledge. Experiments show that our approach outperforms state-of-the-art baselines on both the accuracy of rating prediction and the quality of generated explanations.

References

[Alvarez-Melis and Jaakkola, 2018] David Alvarez-Melis and Tommi S. Jaakkola. Towards robust interpretability with self-explaining neural networks. In *NIPS*, pages 7786–7795, 2018.

[Chen *et al.*, 2018] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. Neural attentional rating regression with review-level explanations. In *WWW*, pages 1583–1592, 2018.

[Chung *et al.*, 2014] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[Erkan and Radev, 2004] Günes Erkan and Dragomir R Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479, 2004.

[He *et al.*, 2015] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. Trirank: Review-aware explainable recommendation by modeling aspects. In *CIKM*, pages 1661–1670, 2015.

[Jang *et al.*, 2017] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *SIGKDD*, pages 426–434, 2008.

[Lang, 2000] Annie Lang. The limited capacity model of mediated message processing. *Journal of communication*, 50(1):46–70, 2000.

[Lee and Seung, 2001] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2001.

[Li *et al.*, 2016] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. In *NAACL HLT*, pages 110–119, 2016.

[Li *et al.*, 2017] Piji Li, Zihao Wang, Zhaochun Ren, Lidong Bing, and Wai Lam. Neural rating regression with abstractive tips generation for recommendation. In *SIGIR*, pages 345–354, 2017.

[Lin, 2004] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*, 2004.

[McAuley and Leskovec, 2013] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of ACM conference on Recommender systems*, pages 165–172, 2013.

[Mnih and Salakhutdinov, 2008] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264, 2008.

[Papineni *et al.*, 2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318, 2002.

[Peake and Wang, 2018] Georgina Peake and Jun Wang. Explanation mining: Post hoc interpretability of latent factor models for recommendation systems. In *SIGKDD*, pages 2060–2069, 2018.

[Rago *et al.*, 2018] Antonio Rago, Oana Cocarascu, and Francesca Toni. Argumentation-based recommendations: Fantastic explanations and how to find them. In *IJCAI*, pages 1949–1955, 2018.

[Rendle, 2010] Steffen Rendle. Factorization machines. In *ICDM*, pages 995–1000, 2010.

[Sharma and Cosley, 2013] Amit Sharma and Dan Cosley. Do social explanations work?: Studying and modeling the effects of social explanations in recommender systems. In *WWW*, pages 1133–1144, 2013.

[Tay *et al.*, 2018] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Multi-pointer co-attention networks for recommendation. *SIGKDD*, 2018.

[Vig *et al.*, 2009] Jesse Vig, Shilad Sen, and John Riedl. Tagsplanations: Explaining recommendations using tags. In *International Conference on Intelligent User Interfaces*, pages 47–56, 2009.

[Vinyals *et al.*, 2015] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *NIPS*, pages 2692–2700, 2015.

[Wang *et al.*, 2015] Zhongyuan Wang, Haixun Wang, Ji-Rong Wen, and Yanghua Xiao. An inference approach to basic level of categorization. In *CIKM*, pages 653–662, 2015.

[Wang *et al.*, 2018a] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. Tem: Tree-enhanced embedding model for explainable recommendation. In *WWW*, pages 1543–1552, 2018.

[Wang *et al.*, 2018b] Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. A reinforcement learning framework for explainable recommendation. In *ICDM*, 2018.

[Williams, 1992] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[Wu *et al.*, 2012] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. Probase: A probabilistic taxonomy for text understanding. In *SIGMOD*, pages 481–492, 2012.

[Zhang *et al.*, 2014] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*, pages 83–92, 2014.