Multi-Pointer Co-Attention Networks for Recommendation

Yi Tay
Nanyang Technological University
Singapore
ytay017@e.ntu.edu.sg

Luu Anh Tuan
Institute for Infocomm Research
Singapore
at.luu@i2r.a-star.edu.sg

Siu Cheung Hui
Nanyang Technological University
Singapore
asschui@ntu.edu.sg

ABSTRACT

Many recent state-of-the-art recommender systems such as D-ATT, TransNet and DeepCoNN exploit reviews for representation learning. This paper proposes a new neural architecture for recommendation with reviews. Our model operates on a multi-hierarchical paradigm and is based on the intuition that not all reviews are created equal, i.e., only a selected few are important. The importance, however, should be dynamically inferred depending on the current target. To this end, we propose a review-by-review pointer-based learning scheme that extracts important reviews from user and item reviews and subsequently matches them in a word-by-word fashion. This enables not only the most informative reviews to be utilized for prediction but also a deeper word-level interaction. Our pointer-based method operates with a gumbel-softmax based pointer mechanism that enables the incorporation of discrete vectors within differentiable neural architectures. Our pointer mechanism is co-attentive in nature, learning pointers which are co-dependent on user-item relationships. Finally, we propose a multi-pointer learning scheme that learns to combine multiple views of user-item interactions. We demonstrate the effectiveness of our proposed model via extensive experiments on **24** benchmark datasets from Amazon and Yelp. Empirical results show that our approach significantly outperforms existing state-of-the-art models, with up to 19% and 71% relative improvement when compared to TransNet and DeepCoNN respectively. We study the behavior of our multi-pointer learning mechanism, shedding light on '*evidence aggregation*' patterns in review-based recommender systems.

KEYWORDS

Deep Learning; Recommendation; Collaborative Filtering; Review-based Recommender Systems; Information Retrieval; Natural Language Processing

ACM Reference Format:
Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-Pointer Co-Attention Networks for Recommendation. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3220086

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220086

1 INTRODUCTION

On most e-commerce platforms today, the ability to write and share reviews is not only a central feature but is also a strongly encouraged act. Reviews are typically informative, pooling an extensive wealth of knowledge for prospective customers. However, the extensive utility of reviews do not only end at this point. Reviews are also powerful in capturing preferences of authors, given the rich semantic textual information that cannot be conveyed via implicit interaction data or purchase logs. As such, there have been immense interest in collaborative filtering systems that exploit review information for making better recommendations [2, 5, 8, 19, 39, 41].

Recent advances in deep learning has spurred on various innovative models that exploit reviews for recommendation [3, 26, 41]. The intuition is simple yet powerful, i.e., each user is represented as all reviews he (she) has written and an item is represented by all reviews that was written for it. All reviews are concatenated to form a single user (item) document. Subsequently, a convolutional encoder is employed to learn a single latent representation for the user (item). User and item embeddings are then matched using a parameterized function such as Factorization Machines [23]. This has shown to be highly performant [26], outperforming a wide range of traditionally strong baselines such as Matrix Factorization (MF). Models such as DeepCoNN [41], TransNets [3] and D-ATT [26] are recent state-of-the-art models that are heavily grounded in this paradigm.

Intuitively, this modeling paradigm leaves a lot to be desired. Firstly, the naive concatenation of reviews into a single document is unnatural, ad-hoc and noisy. In this formulation, reviews are treated indiscriminatingly irregardless of whether they are important or not. A user's bad review about a coffee shop should be mostly irrelevant when deciding if a Spa is a good match. Secondly, user and item representations are static irregardless of the target match. For example, when deciding if a coffee shop is a good match for a user, the user's past reviews about other coffee shops (and eateries) should be highly relevant. Conversely, reviews about spas and gyms should not count. Hence, a certain level of dynamism is necessary. Finally, the only accessible interaction between user and item is through a fixed dimensional representation, which is learned via excessive compression of large user-item review banks into low-dimensional vector representations. For richer modeling of user and item reviews, deeper and highly accessible interaction interfaces between user-item pairs should be mandatory.

Recall that reviews were fundamentally written independently, at different times, and for different products (or by different people). There should be no reason to squash everything into one long document if they can be modeled independently and then combined later. More importantly, a user may write hundreds of reviews over an extended period of time while an item may effortlessly receive a thousand of reviews. As such, existing modeling paradigms will

eventually hit a dead-end. Overall, this work proposes four major overhauls have to be made to existing models, i.e., (1) reviews should be modeled independently, (2) not all reviews are equally important and should be weighted differently, (3) the importance of each review should be dynamic and dependent on the target match and finally, (4) user and item reviews should interact not only through compressed vector representations but also at a deeper granularity, i.e., word-level.

To this end, we propose a *Multi-Pointer Co-Attention Network* (MPCN), a novel deep learning architecture that elegantly satisfies our key desiderata. Our model is multi-hierarchical in nature, i.e., each user is represented by n reviews of ℓ words each. Subsequently, all user and item reviews (for this particular instance pair) are matched to determine the most informative reviews. In order to do so, we design a novel pointer-based co-attention mechanism. The pointer mechanism extracts the named reviews for direct review-to-review matching. At this stage, another co-attention mechanism learns a fixed dimensional representation, by modeling the word-level interaction between these matched reviews. This forms the crux of our *review-by-review* modeling paradigm. Finally, we introduce a multi-pointer learning scheme that can be executed an arbitrary k times, extracting multiple multi-hierarchical interactions between user and item reviews.

1.1 Our Contributions

In summary, the prime contributions of this paper are as follows:

- We propose a state-of-the-art neural model for recommendation with reviews. Our proposed model exploits a novel pointer-based learning scheme. This enables not only noise-free but also deep word-level interaction between user and item.
- We conduct experiments on **24** benchmark datasets. Our proposed MPCN model outperforms all state-of-the-art baselines by a significant margin across all datasets. Our compared baselines are highly competitive, encompassing not only review-based models but also state-of-the-art interaction-only models. We outperform models such as Neural Matrix Factorization (NeuMF) [10], DeepCoNN [41], D-ATT [26] and TransNet [3]. Performance improvement over DeepCoNN, TransNets and D-ATT are up to 71%, 19% and 5% respectively.
- We investigate the inner workings of our proposed model and provide insight about how MPCN works under the hood. Additionally, analyzing the behavior of our pointer mechanism allows us to better understand the nature of the problem. Through analysis of our pointer mechanism, we show that different problem domains have different patterns of *'evidence aggregation'*, which might require different treatment and special care.

2 RELATED WORK

In this section, we identify and review several key areas which are highly relevant to our work.

2.1 Review-based Recommendation

The utility of exploiting reviews for recommendations have been extensively discussed and justified in many works [3, 13, 19, 26, 41]. This not only enables a mitigation of cold-start issues but also provides a richer semantic modeling of user and item characteristics. While relatively earlier works have mainly concentrated efforts on topic modeling and language modeling approaches [5, 19, 31], the recent shift towards deep learning models is prominent. The advantages of neural architectures are clear, i.e., not only do these models dispense with laborious feature engineering altogether, they are often highly competitive. In many recent works, Convolutional neural networks (CNN) act as automatic feature extractors, encoding a user (item) into a low-dimensional vector representation. User and item embeddings are then compared with a matching function.

An earlier neural model, the Deep Co-operative Neural Networks (DeepCoNN) [41] represents a user as all the reviews that he (she) has written. Likewise, an item is represented as all the reviews ever written (by other users) for it. User and item documents are then encoded with CNNs and passed into a Factorization Machine (FM) [23] for matching. It was later argued that DeepCoNN's competitive performance exploits the fact that test reviews were leaked (into the training set) [3]. As such, this reduces the recommendation problem to resemble a noisy adaptation of standard document-level sentiment analysis. To this end, [3] proposed TransNets, augmenting a DeepCoNN-like neural network with an additional multi-task learning scheme. More specifically, it learns to transform the penultimate hidden layer of DeepCoNN into a CNN-encoded representation of the test review. This signal was found to be useful, improving the performance on multiple benchmarks.

DeepCoNN and TransNet are relatively simple model architectures. Another recently proposed model, the Dual Attention CNN model (D-ATT) [26] proposed augmenting CNNs with neural attention. The key idea of neural attention [1] is to emphasize important segments of documents by weighting each word by a learned *attention vector*. The final representation comprises a weighted linear combination of all input embeddings. Two variants of attention mechanism are proposed, i.e., local and global, both modeling different views of user-item review documents. However, these models are not without flaws. As mentioned, these models follow the same paradigm of representing user and item as a giant concatenated document of all their reviews which suffers inherent drawbacks such as (1) noise, (2) lack of dynamic target-dependent and (3) lack of interaction interfaces.

2.2 Text Matching and Co-Attentional Models

Our work is closely related to the problem domain of sequence pair modeling. A wide spectrum of models have been proposed for modeling relationships between two sequences, e.g., question-answer [6], premise-hypothesis [22, 24, 28] which are very similar to the user-item modeling problem at hand. In these fields, learning representations without fine-grained interaction modeling, i.e., absence of interaction interfaces in DeepCoNN, is known to be far outperformed by new models which utilize co-attentional mechanisms [6, 40]. Co-attentions learn pairwise attention between two sequences [37] (or modalities [17]), enabling pair-aware attention weights to be learned.

2.3 Recent Advances in Deep Learning

Notably, our network solely relies on attention mechanisms, and showcases the potential neural architectures that do not use convolutional and recurrent layers. This is inspired by the Transformer [29] architecture which uses multi-headed attention, concatenating outputs of each attention call. Consequently, our proposed model does not use any recurrent or convolution layers, and solely relies on attention.

Additionally, our work is characterized by the usage of pointers, which have been popularized by both Pointer Networks [30]. These networks learn to predict an output token which exists in the sequence itself, i.e., pointing to a token. The usage of pointers is primarily motivated for discrete problems and has also been widely adopted for answer span prediction in question-answering [33]. In these models, pointers are commonly applied at the last layer and have no issues since the model optimizes a loss function such as the cross entropy loss. However, our model requires the usage of pointers *within* the network (between layers). As such, a form of hard attention is required. Due to the non-differentiability of hard attention, it has been much less popular than the standard soft attention. In order to generate discrete pointers, we utilize the recent gumbel-softmax [12] trick which is capable of learning one-hot encoded vectors. Notably, the gumbel-softmax was recently adapted for automatic compositional learning of Gumbel TreeLSTMs [4], in which the gumbel-softmax is exploited to adaptively learn to make merging decisions.

2.4 Deep Learning for Recommendation

Factorization-based models [11, 21, 23] were popular standard machine learning baselines for interaction-based recommendation. Today, we see a shift into deep learning, in which neural models are claiming state-of-the-art performance [10, 27, 36, 38]. He et al. [10] proposed a neural framework that combines a generalized matrix factorization formulation with multi-layered perceptrons. He and Chua [9] proposed a neural adaptation of factorization machines. [35] proposed recurrent models for sequential modeling of interaction data. Tay et al. [27] proposed a translation-based framework that exploits neural attention for modeling user-item relations. Li et al. [15] proposed a encoder-decoder based framework for both rating prediction and generating tips. A separate class of deep models based on auto-encoders [16, 25, 32] has also been proposed for recommendation tasks.

3 OUR PROPOSED MODEL

In this section, we present a layer-by-layer description of our proposed model. The overall model architecture is illustrated in Figure 2.

3.1 Input Encoding

Our model accepts two input sequences, a (user) and b (item). Each input sequence is a list of reviews $\{r_1, r_2 \cdots r_{\ell_d}\}$ where ℓ_d is the maximum number of reviews.

3.1.1 Embedding Layer. Each review is a sequence of ℓ_w words which are represented as one-hot encoded vectors. For a and b, we pass all words into an embedding matrix $\mathbf{W}^{d \times |V|}$ where V is the vocabulary, retrieving a d dimensional vector for each word. Our network is hierarchical in nature, i.e., instead of representing all user (or item) reviews as one long document, we represent each user (or item) as a sequence of reviews. Each review is then hierarchically constructed from a sequence of words.

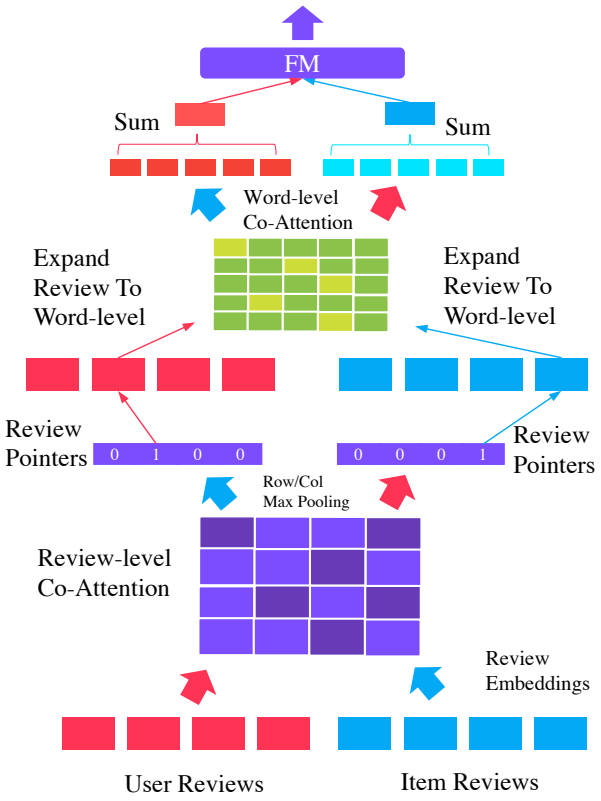


Figure 1: Illustration of proposed model architecture for Pointer-based Learning (*Best viewed in color*). This example illustrates a one pointer example. Review gating mechanism and multi-pointer learning is omitted for clarity.

3.1.2 Review Gating Mechanism. Each review is represented as a sum of its constituent word embeddings to form the vector $x \in \mathbb{R}^d$. Intuitively, not all reviews that a user writes and not all reviews written for a product is important. We design a first-stage filter, i.e., a review gating mechanism that accepts each review as an input and controls how much information passes through to the next level. Given the input $x \in \mathbb{R}^{\ell_r \times d}$, which represents either a or b.

$$\bar{x_i} = \sigma(\mathbf{W}_g x_i) + \mathbf{b}_g \odot tanh(\mathbf{W}_u x_i + b_u) \quad (1)$$

where $\odot$ is the Hadamard product and σ is the sigmoid activation function. x_i is the i-th review of sequence x. $\mathbf{W}_g, \mathbf{W}_u \in \mathbb{R}^{d \times d}$ and $b_g, b_u \in \mathbb{R}^n$ are parameters of this layer. While the purpose of the co-attention layer is to extract important reviews, we hypothesize that applying a pre-filter (gating mechanism) helps improve performance on certain datasets.

3.2 Review-level Co-Attention

In this layer, the aim is to select the most informative review from the review bank of each user and item respectively.

3.2.1 Affinity Matrix. Given a list of review embeddings from each user ($a \in \mathbb{R}^{\ell_r \times d}$) and item ($b \in \mathbb{R}^{\ell_r \times d}$) banks, we calculate an affinity matrix between them. This is described as follows:

$$s_{ij} = F(a_i)^\top \mathbf{M}\, F(b_j) \tag{2}$$

where $\mathbf{M}^{d \times d}$ and $S \in \mathbb{R}^{\ell_r \times \ell_r}$. $F(.)$ is a feed-forward neural network function with l layers. In practice, l is tuned amongst $[0, 2]$ where $l = 0$ reverts Equation (2) to the bilinear form.

3.2.2 Pooling Function. By taking the row and column wise maximum of the matrix s and using them to weight the original list of reviews a and b, we are able to derive the standard co-attention mechanism. This is described as follows:

$$a' = (G(\max_{col}(s)))^\top a \quad \text{and} \quad b' = (G(\max_{row}(s)))^\top b \tag{3}$$

There are different choices for the pooling operation. Max pooling is used here because, intuitively it selects the review which has the maximum influence (or affinity) with all reviews from its partner. This type of co-attention is known to be extractive, characterized by its usage of max pooling.

Note that we apply the function $G(.)$ to $\max_{col}(s)$ and $\max_{row}(s)$. In most applications, $G(.)$ would correspond to the standard softmax function, which converts the input vector into a probability distribution. The vectors a', b' would then be the *co-attentional vector representations*. However, in our case, we desire further operations on the selected reviews and therefore do not make use of these representations. Instead, $G(.)$ has to return a one-hot encoded vector, pointing to the selected reviews which forms the real objective behind this co-attentional layer. However, the Softmax function returns a continuous vector, which is unsuitable for our use-case. The usage of discrete vectors in neural architectures is known to be difficult, as the arg max operation is non-differentiable. Hence, we leverage a recent advance, the Gumbel-Max trick, for learning to point. The next section describes this mechanism.

3.3 Review Pointers

We leverage a recent advance, the Gumbel-Softmax [12], for incorporating discrete random variables in the network.

3.3.1 Gumbel-Max. In this section, we describe Gumbel-Max [18], which facilitates the key mechanism of our MPCN model. Gumbel-Max enables discrete random variables (e.g., one-hot vectors) to be utilized within an end-to-end neural network architecture. Consider a k-dimensional categorical distribution where class probabilities $p_1, \cdots p_k$ are defined in terms of unnormalized log probabilities $\pi_1, \cdots \pi_k$:

$$p_i = \frac{\exp(\log(\pi_i))}{\sum_{j=1}^{k} \exp(\log(\pi_j))} \tag{4}$$

A one-hot sample $z = (z_1, \cdots z_k) \in \mathbb{R}^k$ from the distribution can be drawn by using the following:

$$z_i = \begin{cases} 1\,, i = \arg\max_j(\log(\pi_j) + g_j) \\ 0\,, \text{otherwise} \end{cases} \tag{5}$$

$$g_i = -\log(-\log(u_i)) \qquad u_i \sim Uniform(0, 1) \tag{6}$$

where g_i is the *Gumbel noise* which perturbs each $\log(\pi_i)$ term such that the arg max operation is equivalent to drawing a sample weighted by $p_i, \cdots p_k$.

3.3.2 Straight-Through Gumbel-Softmax. In the Gumbel-Softmax, the key difference is that the arg max function is replaced by the differentiable softmax function which is described as follows:

$$y_i = \frac{\exp(\frac{\log(\pi_i)+g_i}{\tau})}{\sum_{j=1}^{k} \exp(\frac{\log(\pi_j)+g_i}{\tau})} \tag{7}$$

where τ, the temperature parameter, controls the extend of how much the output approaches a one hot vector. More concretely, as τ approaches zero, the output of the Gumbel-Softmax distribution becomes cold, i.e., becomes closer to a one-hot vector. In the straight-through (ST) adaptation, the forward-pass is discretized by converting the vector output to a one-hot vector via arg max:

$$y_i = \begin{cases} 1\,, i = \arg\max_j(y_j) \\ 0\,, \text{otherwise} \end{cases} \tag{8}$$

However, the backward pass maintains the flow of continuous gradients which allows the model to be trained end-to-end. This is useful as we only want important reviews to be selected (i.e., hard selection) to be considered in computation of the loss function. Notably, alternatives such as REINFORCE [34] exist. However, it is known to suffer from high variance and slow convergence [12].

3.3.3 Learning to Point. In order to compute a review pointer (for user and item), we set $G(.)$ in Equation (3) to use the Gumbel Softmax. However, since we are interested only in the pointer (to be used in subsequent layers), the pointer is then calculated as:

$$p_a = (Gumbel(\max_{col}(s))) \quad \text{and} \quad p_b = (Gumbel(\max_{row}(s))) \tag{9}$$

By applying p_a to a, we select the $p_a - th$ review of user a and $p_b - th$ review of item b. The selected reviews are then passed into the next layer where rich interactions are extracted between these reviews.

3.4 Word-level Co-Attention

The review-level co-attention smooths over word information as it compresses each review into a single embedding. However, the design of the model allows the most informative reviews to be extracted by the use of pointers. These reviews can then be compared and modeled at word-level. This allows a user-item comparison of finer granularity which facilitates richer interactions as compared to simply composing the two review embeddings. Let $\bar{a}, \bar{b}$ be the selected reviews using the pointer learning scheme. Similar to the review-level co-attention, we compute a similarity matrix between $\bar{a}$ and $\bar{b}$. The key difference is that the affinity matrix is computed word-by-word and not review-by-review.

$$w_{ij} = F(\bar{a}_i)^\top \mathbf{M}_w\, F(\bar{b}_j) \tag{10}$$

where $\mathbf{M}_w \in \mathbb{R}^{d \times d}$ and $w \in \mathbb{R}^{\ell_w \times \ell_w}$. Next, to compute the *co-attentional* representation of reviews $\bar{a}, \bar{b}$, we take the mean pooling.

$$\bar{a}' = (S(avg_{col}(w)))^\top \bar{a} \quad \text{and} \quad \bar{b}' = (S(avg_{row}(w)))^\top \bar{b} \tag{11}$$

where $S(.)$ is the standard Softmax function and $F(.)$ is a standard feed-forward neural network with l layers. The rationale for using the average pooling operator here is also intuitive. At the review-level, a large maximum affinity score of a review with respect to all *opposite* reviews (even when a low average affinity socre) warrants it being extracted, i.e., a strong signal needs to be further investigated. However, at a word-level, max-pooling may be biased towards identical words and may have a great inclination to act as a word matching operator. Hence, we want to maintain a stable co-attentive extractor. Our early empirical experiments also justify this design. Finally $\bar{a}'$ and $\bar{b}'$ are the output representations.

Note that, implementation of co-attention layers (review-level and word-level) is equivalent to only two simple MATMUL operations (in Tensorflow). As such, scalability is not really a concern in our approach since this is quite efficiently optimized on GPUs.

3.5 Multi-Pointer Learning

While our objective is to eliminate noisy reviews by the usage of hard pointers, there might be insufficient information if we point to only a single pair of reviews. Hence, we devise a multi-pointer composition mechanism. The key idea is to use multiple pointers where the number of pointers n_p is a user-defined hyperparameter. Our model runs the Review-level Co-Attention n_p times, with each generating a unique pointer. Each of the n_p review pairs is then modeled with the Word-level Co-Attention mechanism. The overall output is a list of vectors $\{\bar{a}'_1, \cdots \bar{a}'_{n_p}\}$ and $\{\bar{b}'_1, \cdots \bar{b}'_{n_p}\}$. Additionally, we also found it useful to include the sum embedding of all words belonging to the user (item). This mainly helps in robustness, in case where user and item do not have any matching signals that were found by our pointer mechanism. We explore three ways to compose these embeddings.

- **Concatenation** - All pointer outputs are concatenated, e.g., $[\bar{a}'_1; \cdots \bar{a}'_{n_p}]$. This has implications to the subsequent layers, especially if n_p is large. Hence we consider the next two alternatives.
- **Additive Composition** - All pointer outputs are summed, e.g., $sum(\bar{a}'_1, \cdots \bar{a}'_{n_p})$.
- **Neural Network** - All pointer outputs are concatenated and passed through a single non-linear layer with ReLU (σ_r) activations. e.g., $\sigma_r(W([\bar{a}'_1; \cdots \bar{a}'_{n_p}]) + b)$ which maps the concatenated vector into a d dimensional vector.

Note that this is applied to $\bar{b}'$ as well but omitted for brevity. Let the output of this layer be a_f and b_f. In our experiments, we tune amongst the three above-mentioned schemes. More details are provided in the ablation study.

3.6 Prediction Layer

This layer accepts a_f, b_f as an input. The concatenation of $[a_f; b_f]$ is passed into a factorization machine (FM) [23]. FM accepts a real-valued feature vector and models the pairwise interactions between features using factorized parameters. The FM function is defined as follows:

$$F(x) = w_0 + \sum_{i=1}^{n} w_i \, x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle \, x_i \, x_j \qquad (12)$$

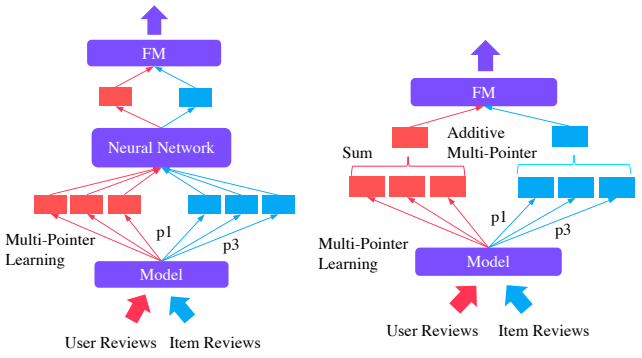


Figure 2: Illustration of Neural Network (*left*) and Additive (*right*) based Multi-Pointer Learning.

where $x \in \mathbb{R}^k$ is a real-valued input feature vector. $\langle ., . \rangle$ is the dot product. The parameters $\{v_1 \ldots v_n\}$ are factorized parameters (vectors of $v \in \mathbb{R}^k$) used to model pairwise interactions (x_i, x_j). w_0 is the global bias and $\sum_{i=1}^{n} w_i \, x_i$ represents a linear regression component. The output of $F(x)$ is a scalar, representing the strength of the user-item interaction. The network is trained end-to-end by minimizing the standard mean squared error loss following [26].

4 EMPIRICAL EVALUATION

In this section, we present our experimental setup and empirical evaluation. Our experiments are designed to answer the following research questions (RQs):

(1) **RQ1** - Does our proposed approach outperform state-of-the-art models such as D-ATT and DeepCoNN? How much is the relative improvement?
(2) **RQ2** - What are the impacts of some of the design / architectural choices of MPCN?
(3) **RQ3** - What are the effects of the key hyperparameters of our model (e.g., number of pointers, etc.) on model performance?
(4) **RQ4** - Are we able to derive any insight about how MPCN works by analyzing the behavior of the pointer layer?

4.1 Datasets

We utilize datasets from two different sources which are described as follows:

(1) **Yelp Dataset Challenge** - Yelp is an online review platform for businesses such as restaurants, bars, spas, etc. We use the dataset from the latest challenge[1].
(2) **Amazon Product Reviews** - Amazon is a well-known E-commerce platform. Users are able to write reviews for products they have purchased. We use **23** datasets from the Amazon Product Review corpus[2] [7, 20].

In total, we provide model comparisons over **24** benchmark datasets. For all datasets, we split interactions into training, development and testing sets. We utilize a time-based split, i.e., the last item of each user is added to the test set while the penultimate is used for development. For Amazon, the datasets are preprocessed in a

[1] https://www.yelp.com/dataset/challenge
[2] http://jmcauley.ucsd.edu/data/amazon/

5-core fashion (i.e., each user and item have at least 5 reviews to be included). Since the datasets can be found in the official webpage, we do not restate their statistics to save space. For Yelp, we use a 20-core setting, providing a comparison on a denser dataset. We tokenize the reviews using NLTK and retain words that appear at least 10 times in the vocabulary. We would like to emphasize that, when building user and item representations using their respective reviews, all reviews belonging to interactions from the test and development sets **were not included**. This is to prevent the problem from reverting to a sentiment analysis task, albeit noisier [3].

4.2 Compared Methods

We compare against a series of competitive baselines.

- **Matrix Factorization** (MF) is a standard and well-known baseline for CF. It represents the user and item rating with the inner product, i.e., $p^\top q$.
- **Factorization Machines** (FM) [23] are general purpose machine learning algorithms that use factorized parameters to model pairwise interaction within a real-valued feature vector. We concatenate the user-item latent embedding together and pass it through the FM model.
- **Multi-layered Perceptrons** (MLP) are strong neural baselines for CF and were used as a baseline in [10]. We use the same pyramidal scheme of 3 layers.
- **Neural Matrix Factorization** (NeuMF) [10] is the state-of-the-art model for interaction-only CF. It casts the MF model within a neural framework and combines the output with multi-layered perceptrons.
- **Deep Co-Operative Neural Networks** (DeepCoNN) [41] is a review-based convolutional recommendation model. It trains convolutional representations of user and item and passes the concatenated embedding into a FM model.
- **TransNet** [3] is an improved adaptation of DeepCoNN which incorporates transform layers and an additional training step that enforces the transformed representation to be similar to the embedding of the actual target review.
- **Dual Attention CNN Model** (D-ATT) [26] is a recently proposed state-of-the-art CNN-based model that uses reviews for recommendation. This model is characterized by its usage of two forms of attentions (local and global). A final user (item) representation is learned by concatenating representations learned from both local and global attentions. The dot product between user and item representations is then used to estimate the rating score.

Given our already extensive comparisons against the state-of-the-art models, we omit comparisons with models such as HFT [19], Collaborative Topic Regression [31], Collaborative Deep Learning (CDL) [32] and ConvMF [13] since they have been outperformed by the recently proposed DeepCoNN or D-ATT model.

4.3 Experimental Setup

The evaluation metric is the well-known (and standard) mean-squared error which measures the square error between the rating prediction and ground truth. We implement all models ourselves in Tensorflow. All models are trained with Adam [14] with an initial learning rate of 10^{-3}. We train all models for a maximum of 20 epochs with early stopping (i.e., if model performance does not improve for 5 epochs) and report the test result from the best performing model on the development set. We found that models tend to converge before 20 epochs. However, an exception is that the MF baseline requires many more epochs to converge. As such, we train the MF model till convergence. For interaction only models, the embedding size is set to 50. For TransNet and DeepCoNN, the number of filters is set to 50 and the filter size is 3. For D-ATT, the global attention layer uses filter sizes of $[2, 3, 4]$. The word embedding layer is also set to 50 dimensions. We regularize models with a dropout rate of 0.2 and a fixed L2 regularization of 10^{-6}. Dropout is applied after all fully-connected and convolutional layers. We use two transform layers in the TransNet model. All word embeddings are learned from scratch as we found that using pretrained embeddings consistently degrades performance across all datasets (and models). The maximum document length is set to 600 words (20 reviews of 30 tokens each) which we empirically found to be a reasonable length-specific performance bound. We assign a special delimiter token to separate reviews within a user (item) document for DeepCoNN, TransNet and D-ATT. If FM is used, the number of factors is set to 10. For our proposed model, the number of pointers p is tuned amongst $\{1, 3, 5, 8, 10\}$. On most datasets, the optimal performance is reached with $2 - 3$ pointers.

4.4 Experimental Results

Table 1 reports the results of our experiments. Firstly, we observe that our proposed MPCN is the top performing model on all 24 benchmark datasets. This ascertains the effectiveness of our proposed model and clearly answers **RQ1**. MPCN consistently and significantly outperforms DeepCoNN, TransNet and D-ATT, which are all recent competitive review-based methods for recommendation. The relative improvement is also encouraging with gains of up to 71% (DeepCoNN), 19% (TransNet) and 5% (D-ATT). On majority of the datasets, performance gains are modest, seeing an improvement of $1\% - 3\%$ for most models. Notably, the average percentage improvement of MPCN over DeepCoNN is 16%. The average performance gain over TransNet and D-ATT is a modest 3.2% and 2.2% respectively.

Pertaining to the relative ranking of the review-based models, our empirical evaluation reaffirms the claim of [3], showing that TransNet always outperforms DeepCoNN. However, the relative ranking of D-ATT and TransNet switches positions frequently. Notably, TransNet uses the test review(s) as an additional data source (albeit as a training target) while D-ATT does not make use of this information. The additional training step of TransNet is actually quite effective and hypothetically could be used to enhance D-ATT or MPCN as well. However, we leave that for future work.

Next, the performance of interaction-only models (MF, FM, etc.) is consistently lower than review-based models (e.g., DeepCoNN). The relative performance of all interaction models is generally quite inconsistent over various datasets. However, one consistent fact is that MF performs worse than other models most of the time. The top scoring interaction model often switches between FM and MLP.

Finally, we give readers a sense of computational runtime. We provide an estimate that we found generally quite universal across multiple datasets. Let t be the runtime of DeepCoNN, the runtime of

	Interaction-based				Review-based				Improvement (%)		
Dataset	MF	FM	MLP	NEUMF	D-CON	TNET	D-ATT	MPCN	Δ_{DC}	Δ_{TN}	Δ_{DA}
Yelp17	1.735	1.726	1.727	1.691	1.385	1.363	1.428	**1.349**	2.7	1.0	5.9
Instant Video	2.769	1.331	1.532	1.451	1.285	1.007	1.004	**0.997**	28.9	1.0	0.7
Instruments	6.720	1.166	1.081	1.189	1.483	1.100	0.964	**0.923**	60.7	19.2	4.4
Digital Music	1.956	1.382	1.361	1.332	1.202	1.004	1.000	**0.970**	23.9	3.5	3.1
Baby	1.755	1.614	1.585	1.598	1.440	1.338	1.325	**1.304**	10.4	2.6	1.6
Patio / Lawn	2.813	1.311	1.279	1.251	1.534	1.123	1.037	**1.011**	51.7	11.1	2.6
Gourmet Food	1.537	1.348	1.442	1.464	1.199	1.129	1.143	**1.125**	6.6	0.4	1.6
Automotive	5.080	1.599	1.071	1.013	1.130	0.946	0.881	**0.861**	31.2	9.9	2.3
Pet Supplies	1.736	1.618	1.649	1.646	1.447	1.346	1.337	**1.328**	9.0	1.4	0.7
Office Products	1.143	0.998	1.122	1.069	0.909	0.840	0.805	**0.779**	16.7	7.8	3.3
Android Apps	1.922	1.871	1.805	1.842	1.517	1.515	1.509	**1.494**	1.5	1.4	1.0
Beauty	1.950	1.711	1.631	1.552	1.453	1.404	1.409	**1.387**	4.8	1.2	1.6
Tools / Home	1.569	1.310	1.356	1.314	1.208	1.122	1.101	**1.096**	10.2	2.4	0.5
Video Games	1.627	1.665	1.576	1.568	1.307	1.276	1.269	**1.257**	4.0	1.5	1.0
Toys / Games	1.802	1.195	1.286	1.222	1.057	0.974	0.982	**0.973**	8.6	0.1	0.9
Health	1.882	1.506	1.522	1.415	1.299	1.249	1.269	**1.238**	4.9	0.9	2.5
CellPhone	1.972	1.668	1.622	1.606	1.524	1.431	1.452	**1.413**	7.9	1.3	2.8
Sports / Outdoors	1.388	1.195	1.120	1.299	1.039	0.994	0.990	**0.980**	6.0	1.4	1.0
Kindle Store	1.533	1.217	1.231	1.398	0.823	0.797	0.813	**0.775**	6.2	2.8	4.9
Home / Care	1.667	1.547	1.584	1.654	1.259	1.235	1.237	**1.220**	3.2	1.2	1.4
Clothing	2.396	1.492	1.462	1.535	1.322	1.197	1.207	**1.187**	11.4	0.8	1.7
CDs / Vinyl	1.368	1.555	1.432	1.368	1.045	1.010	1.018	**1.005**	4.0	0.5	1.3
Movies / TV	1.690	1.852	1.518	1.775	1.960	1.176	1.187	**1.144**	71.3	2.8	3.8
Electronics	1.962	2.120	1.950	1.651	1.372	1.365	1.368	**1.350**	1.6	1.1	1.3

Table 1: Performance comparison (mean squared error) on 24 benchmark datasets. The best performance is in boldface. $\Delta_{DC}, \Delta_{TN}, \Delta_{DA}$ are the relative improvements (%) of MPCN over DeepCoNN (D-CON), TransNet (T-NET) and D-ATT respectively. MPCN achieves state-of-the-art performance, outperforming all existing methods on 24 benchmark datasets.

MPCN $p = 1$ is approximately $\approx 0.4t$. MPCN with 2 and 3 pointers are $0.8t$ and $1.2t$ respectively. TransNet and D-ATT run at $\approx 2t$. On medium size datasets (e.g., Amazon Beauty), $t \approx 40s$ on a GTX1060 GPU (batch size is 128). We found that if $p_{opt} \leq 2$, then MPCN is faster than DeepCoNN. While $p_{opt} \leq 5$ is the threshold for being equal with D-ATT and TransNet in terms of runtime.

5 HYPERPARAMETER & ABLATION ANALYSIS

In this section, we study the impact of key hyperparameter settings and various architectural choices on model performance.

5.1 Ablation Analysis

We study the impacts of various architectural decisions on model performance (**RQ2**). Table 2 reports an ablation analysis conducted on the development sets of four benchmark datasets (*Beauty, Office, Musical Instruments (M-Instr) and Amazon Instant Video (Inst-Vid)*). We report the results of several different model variations. We first begin describing the default setting in which ablation reports are deviated from. In the default setting, we use the standard model with all components (review gates, word-level co-attention and FM prediction layer). The Multi-Pointer aggregation (aggr) is set to use a neural network (single layer nonlinear transform). The number of layers in the co-attention layer is set to $l = 1$.

We report the validation results from 8 different variations, with the aims of clearly showcasing the importance of each component. In (1), we remove the review gating mechanism. In (2), we replace the FM with the simple inner product. In (3-4), we investigate the effects of different pointer aggregation (aggr) functions. They are the concatenate and additive operators respectively. In (5-6), we set $l = 0$ (remove layer) and $l = 2$. In (7), we remove the word level co-attention layer. In this case, the representation for user and item is simply the pointed review embedding. In (8), we remove the review-level co-attention (RLCA). This variation is no longer 'hierarchical', and simply applies word-level co-attention to user and item reviews.

Architecture	Beauty	Office	M-Instr	Inst-Vid
(0) Default	1.290	0.770	**0.827**	0.975
(1) Remove Gates	1.299	**0.760**	0.837	0.979
(2) Remove FM	1.286	0.808	0.924	0.997
(3) Agrr (Concat)	**1.279**	0.788	0.832	**0.971**
(4) Aggr (Additive)	1.290	0.767	0.829	0.971
(5) set $l = 0$	1.293	0.776	0.830	0.976
(6) set $l = 2$	1.295	0.775	0.829	0.974
(7) Remove WLCA	1.296	0.778	0.831	0.999
(8) Remove RLCA	1.304	0.789	0.839	0.1003

Table 2: Ablation analysis (validation MSE) on four datasets.

Firstly, we observe the default setting is not universally the best across four datasets. As mentioned, the review gating mechanism helps in 3 out of 4 presented datasets. In the *Office* dataset, removing the review gating layer improves performance. We found this to be true across most datasets, i.e., the review gating mechanism helps more often than not, but not always. The impacts of removing FM is quite easily noticed, leading to huge performance degradation on the *M-Instr* dataset. Deprovement on *Inst-Vid* and *Office* is also significant. On the other hand, removing FM marginally improved performance on *Beauty*. We also discovered that there is no straightforward choice of *aggr* functions. Notably, the relative ranking of all three variants (concat, additive and neural network) are always interchanging across different datasets. As such, they have to be tuned. We also noticed that the choice of $l = 1$ is safe across most datasets, as increasing or decreasing would often lead to performance degradation. Finally, removing the WLCA and RLCA consistently lowered performance on all datasets, which ascertains the effectiveness of the WLCA layer. Notably, removing RLCA seems to hurt performance more, which signifies that modeling at a review-level is essential.

5.2 Effect of Number of Pointers

Table 3 reports the effect of varying pointers on performance (**RQ3**). We use 4 datasets of varying sizes as an illustrative example (*Patio, Automotive, Sports* and *Video Games*). The datasets shown are sorted from smallest to largest in terms of number of interactions. Clearly, we observe that the optimal number of pointers varies across all datasets. We found this to be true for the remainder datasets that are not shown. This seems to be more of a domain-dependent setting since we were not able to find any correlation with dataset size. For most datasets, the optimal pointers falls in the range of $1 - 3$. In exceptional cases (*Video Games*), the optimal number of pointers was 5.

Ptr	Patio	Automotive	Sports	Video Games
1	0.992	**0.842**	**0.926**	1.885
2	0.980	0.855	0.933	1.178
3	**0.975**	0.843	0.938	1.194
4	0.991	0.844	0.938	1.189
5	0.991	0.844	0.935	**1.121**

Table 3: Validation MSE on various datasets when varying the number of pointers. The best result is in boldface. The optimal number of pointers is domain-dependent.

6 IN-DEPTH MODEL ANALYSIS

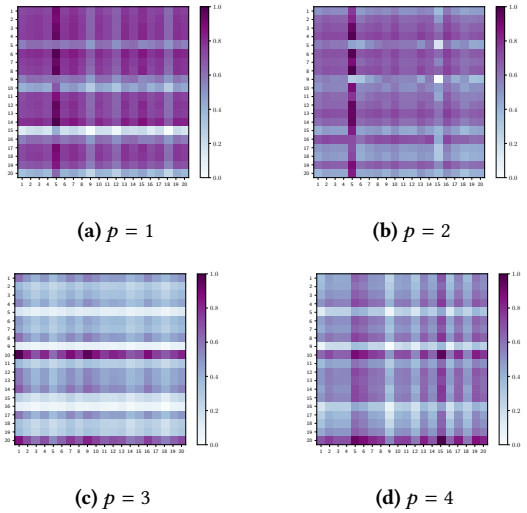
In this section, we present several insights pertaining to the inner workings of our proposed model. This aims to answer **RQ4**.

6.1 What are the pointers pointing to?

In this section, we list some observations by analyzing the behavior of the MPCN model. Firstly, we observed that pointers react to aspects and product sub-categories. In many cases, we observed that the pointer mechanism tries to find the most relevant review written by the user for a given product. If the target item is a phone case, then our model tries to find a review written by the user which is directed towards *another* phone case product. Intuitively, we believe that this is trying to solicit the user's preferences about a type of product. We provide some qualitative examples in Table 4. Consider the context of video games, it finds that the user has written a review about *rpg* (roleplaying games). At the same time, it finds that the item review consists of a (positive) review of the item being a good rpg game. As a result, it surfaces both reviews and concurrently points to both of them. This follows suit for the other examples, e.g., it finds a matching clue of *puzzle games* (turn-based) in the second example. The last example is taken from the *Gourmet Food* dataset. In this case, it discovers that the user likes cocoa, and concurrently finds out that the product in question has some relevant information about chocolate.

6.2 Behavior of Multi-Pointer Learning

In this section, we study the behavior of our multi-pointer mechanism. First and foremost, this serves as another *sanity check* and to observe if multi-pointers are really necessary, i.e., if pointers are not pointing all to the same reviews. Hence, this section aims to provide better insight into the inner workings of our proposed model. We trained a MPCN model with four pointers. A quick observation is that all four pointers point to different reviews (given the same user item pair). This is automatic, and does not require any special optimization constraint (such as explicitly enforcing the model to choose different reviews through an extra optimization term). Moreover, we analyze the affinity matrix from the review-level co-attention. Figure 3 shows the affinity matrix for pointers one to four.



(a) $p = 1$ **(b)** $p = 2$ **(c)** $p = 3$ **(d)** $p = 4$

Figure 3: Visualisation of Review-level Co-Attention. Matching patterns differ significantly across multiple calls, hence generating different pointers.

Secondly, it is also intuitive that it is not absolutely necessary for MPCN to always point to unique reviews given the same user-item pair. We observed a relatively significant *one-to-many* pointer

User Review	Item Review
game is really beautiful! coolest rpg ever...	..a gift for a friend who really loves rpg games. he really loved it!
.. game is completely turned based, so you have time to ponder you actions..	..is a great and engaging puzzler game but wasn't too challenging
just love this little guy ... phone is reasonably easy to put in and take out	..case really fits the 5s like a glove..
is a nice charger..but after a few momths it wasn't charging...	it is clearly a used or refurbished battery..
cocoa is a wonderful, rich tasting, not overly sweet product ..	used to eat the dark and milk chocolate, and then i tried this and can't explain how good they are

Table 4: Excerpts from top matching User and Item reviews extracted by MPCN's pointer mechanism.

Condition	D-Music	Apps	V-Games	Food	Yelp17
(1) All unique	85.2%	87.5%	82.0%	99.2%	99.2%
(2) 1 Repeated	13.2%	12.5%	17.2%	0.7%	0.7%
(3) All Repeated	1.6%	0.0%	1.6%	0.0%	0.0%
(4) One-to-Many	64.8%	57.8%	57.8%	23.4%	43.7%

Table 5: Analysis of Multi-Pointer Behavior of MPCN on five datasets using $n_p = 3$.

pattern on top of the usual *one-to-one* pattern. In this case, the same review for user (item) is being matched with n (many) different reviews from the item (user). This is also observed to be dataset / domain dependent. In a small minority of cases, all pointers pointed to the same reviews constantly (all repeated condition). However, this is understandable as there is just insufficient information in the user and item review bank. Additionally, we analyzed a small sample from the test set, determining if any of the following conditions hold for each test case.

Table 5 reports the percentages of test samples which falls into each category. We report results on five datasets *Digital Music (D-Music)*, *Android Apps (Apps)*, *Video Games (V-Games)*, *Gourmet Food (Food)* and *Yelp17*. Here, we observe that pointer behavior is largely influenced by domain. The first three are concerned with electronic domains while the last two are more focused on food (and restaurants). We clearly observe that Food and Yelp have very similar pointer behavior. In general, the electronic domains usually make an inference using a fewer subsets of reviews. This is made evident by the high one-to-many ratio which signifies that there is often one important review written by the user (or item) that contributes more (and needs to be matched with multiple opposing reviews). Conversely, the food domains require more evidences across multiple reviews. We believe this is one of the biggest insights that our work offers, i.e., shedding light on how evidence aggregation works (and differs across domains) in review-based recommendation.

7 CONCLUSION

We proposed a new state-of-the-art neural model for recommendation with reviews. Our proposed Multi-Pointer Co-Attention Networks outperforms many strong competitors on 24 benchmark datasets from Amazon and Yelp. We conduct extensive analysis on the inner workings of our proposed multi-pointer learning mechanism. By analyzing the pointer behavior across multiple domains, we conclude that different domains (such as food-related and electronics-related) have different *'evidence aggregation'* patterns. While our model dynamically handles this aspect, we believe this warrants further investigation.

8 ACKNOWLEDGEMENTS

The authors thank anonymous reviewers of KDD 2018 for their time and effort to review this paper.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
[2] Da Cao, Xiangnan He, Liqiang Nie, Xiaochi Wei, Xia Hu, Shunxiang Wu, and Tat-Seng Chua. 2017. Cross-platform app recommendation by jointly modeling ratings and texts. *ACM Transactions on Information Systems (TOIS)* 35, 4 (2017), 37.
[3] Rose Catherine and William Cohen. 2017. TransNets: Learning to Transform for Recommendation. *arXiv preprint arXiv:1704.02298* (2017).
[4] Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2017. Unsupervised learning of task-specific tree structures with tree-lstms. *arXiv preprint arXiv:1707.02786* (2017).
[5] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J. Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14*.
[6] Cícero Nogueira dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive Pooling Networks. *CoRR* abs/1602.03609 (2016).
[7] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 507–517.
[8] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 1661–1670.
[9] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. (2017).
[10] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. 173–182.

[11] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. [n. d.]. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016.*

[12] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

[13] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional Matrix Factorization for Document Context-Aware Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, New York, NY, USA.

[14] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).

[15] Piji Li, Zihao Wang, Zhaochun Ren, Lidong Bing, and Wai Lam. 2017. Neural Rating Regression with Abstractive Tips Generation for Recommendation. (2017).

[16] Xiaopeng Li and James She. 2017. Collaborative Variational Autoencoder for Recommender Systems. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*.

[17] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*. 289–297.

[18] Chris J Maddison, Daniel Tarlow, and Tom Minka. 2014. A* sampling. In *Advances in Neural Information Processing Systems*. 3086–3094.

[19] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 165–172.

[20] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 43–52.

[21] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.

[22] Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. 2249–2255.

[23] Steffen Rendle. 2010. Factorization Machines. In *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*. 995–1000.

[24] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664* (2015).

[25] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 111–112.

[26] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017. Interpretable Convolutional Neural Networks with Dual Local and Global Attention for Review Rating Prediction. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17)*.

[27] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Latent Relational Metric Learning via Memory-based Attention for Collaborative Ranking. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 729–739. https://doi.org/10.1145/3178876.3186154

[28] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017. A Compare-Propagate Architecture with Alignment Factorization for Natural Language Inference. (2017). arXiv:arXiv:1801.00102

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv preprint arXiv:1706.03762* (2017).

[30] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*. 2692–2700.

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

[32] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1235–1244.

[33] Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905* (2016).

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

[35] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent Recommender Networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*. 495–503.

[36] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. 3119–3125. https://doi.org/10.24963/ijcai.2017/435

[37] Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic Coattention Networks For Question Answering. *CoRR* abs/1611.01604 (2016).

[38] Shuai Zhang, Lina Yao, Aixin Sun, Sen Wang, Guodong Long, and Manqing Dong. 2018. NeuRec: On Nonlinear Transformation for Personalized Ranking. *arXiv preprint arXiv:1805.03002* (2018).

[39] Wei Zhang, Quan Yuan, Jiawei Han, and Jianyong Wang. 2016. Collaborative Multi-Level Embedding Learning from Reviews for Rating Prediction.. In *IJCAI*. 2986–2992.

[40] Xiaodong Zhang, Sujian Li, Lei Sha, and Houfeng Wang. 2017. Attentive Interactive Neural Networks for Answer Selection in Community Question Answering. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. 3525–3531.

[41] Lei Zheng, Vahid Noroozi, and Philip S. Yu. 2017. Joint Deep Modeling of Users and Items Using Reviews for Recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017*. 425–434.