Self-Attentive Neural Collaborative Filtering

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

Shuai Zhang
University of New South Wales
shuai.zhang@student.unsw.edu.au

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

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

ABSTRACT

The dominant, state-of-the-art collaborative filtering (CF) methods today mainly comprises neural models. In these models, deep neural networks, e.g.., multi-layered perceptrons (MLP), are often used to model nonlinear relationships between user and item representations. As opposed to shallow models (e.g., factorization-based models), deep models generally provide a greater extent of expressiveness, albeit at the expense of impaired/restricted information flow. Consequently, the performance of most neural CF models plateaus at 3-4 layers, with performance stagnating or even degrading when increasing the model depth. As such, the question of how to train really deep networks in the context of CF remains unclear. To this end, this paper proposes a new technique that enables training neural CF models all the way up to 20 layers and beyond. Our proposed approach utilizes a new hierarchical self-attention mechanism that learns introspective intra-feature similarity across all the hidden layers of a standard MLP model. All in all, our proposed architecture, SA-NCF (*Self-Attentive Neural Collaborative Filtering*) is a densely connected self-matching model that can be trained up to 24 layers without plateau-ing, achieving wide performance margins against its competitors. On several popular benchmark datasets, our proposed architecture achieves up to an absolute improvement of $23\% - 58\%$ and $1.3x$ to $2.8x$ fold improvement in terms of nDCG@10 and Hit Ratio (HR@10) scores over several strong neural CF baselines.

KEYWORDS

Deep Learning; Recommendation; Collaborative Filtering; Recommender Systems; Information Retrieval;

1 INTRODUCTION

Recommender systems play a significant role in personalizing the web. These systems not only ameliorate the prevalent issue of consumer overchoice, but also enhance the overall experience for many consumers. As such, these systems are a central component to many ecommerce and content streaming websites today. Given its ubiquity and importance in many real-world applications, the design of highly effective recommender systems remains an interesting and impactful research problem. At its core, collaborative filtering (CF) techniques [35] often drive the recommendation process, learning from historical user-item interactions in order to make future recommendations. Many machine learning techniques have been explored across the past decade, many revolving around the key notion of factorizing the interaction matrix [17, 24, 28].

Recently, the usage of neural networks (or deep learning) for CF has garnered considerable attention [7, 14, 16, 39, 51]. A common argument against shallow factorization models is that they are inherently linear models which may be weak against data with complex underlying structure. On the other hand, stacked nonlinear transforms, i.e., multi-layered perceptrons (MLPs) may have more expressive power. Moreover, MLPs are not only known to be able to universally approximate any function [19] but also model a hierarchical feature representation of the input data. Therefore, many recent works have argued in the favor of deep networks for the CF problem, with the most common strategy being to augment shallow models with a deep/nonlinear component [7, 16].

Unfortunately, going deep is not without weaknesses. For example, difficulty in feature propagation, i.e., restricted information/gradient flow are known problems of deep networks. This is made evident by many empirical works that report a lack of improvment (or worse, a degradation) of performance going beyond three hidden layers [4, 13, 16]. As such, existing *deep* neural CF models are technically really not that *deep* and the question of how to train 'really deep' networks in the context of recommender systems is still largely unclear.

This paper proposes a conceptually simple but highly effective model architecture for CF. Our approach augments a standard MLP network with a hierarchical self-attention mechanism which enables it to scale (performance-wise) up till *twenty* layers and beyond. While a standard MLP plateaus[1] at *three* to *four* layers, our model can reap significant benefits as we continue to increase the model depth. As a result, our model achieves a substantial performance gain of up to 23% to 58% absolute points over typical neural CF architectures. Conversely, many competitors hover at merely slightly

---

[1]To clarify, we mean that there is no substantial improvement relative to shallow MLPs

more than half the performance of our model, suggesting that this is no easy feat at all. Overall, it is not uncommon for our proposed model to *approximately* double (on average, $1.8x$) its performance relative to a standard MLP model, a testament to the effectiveness of this new paradigm.

Our proposed *Self-Attentive Neural Collaborative Filtering* (SA-NCF) model operates via three intuitive steps. Firstly, it models intra-relationships between all hidden layers of the core MLP, matching all intermediate hidden layers with all other hidden layers. Secondly, based on the matching scores of all hidden layers, we compute a self-attention matrix which is then used to learn an overall self-attentive representation, i.e., a weighted sum of all layers of the network. Lastly, this representation is then propagated and used for prediction.

There are several key intuitions behind the design of SA-NCF. Firstly, our model creates a pathway from one hidden layer to all other hidden layers. This aids in alleviating the problem of restricted information flow and is in similar spirit to residual strategies [10]. Additionally, by providing an alternative pathway for all hidden layers to reach the prediction layer, our network alters its hidden representations accordingly and encourages globally-aware feature representations. Secondly, the hidden state can be thought of as an intermediate joint user-item representation. As such, self-matching (i.e., introspective reasoning) can create richer feature representations as this enables a greater extent of interactions between user-item pairs. To the best of our knowledge, these ideas (e.g., self-matching) have never been explored within the context of collaborative filtering and recommender systems. Moreover, self-attention was primarily designed within the context of sequence encoding in NLP [37, 45]. As such, the notion of exploring the usage of self-attention across feature hierarchies is also new. All in all, the prime contributions of this paper are summarized as follows:

- We propose a new neural architecture for the collaborative filtering with implicit feedback (or personalized ranking) problem. Our proposed model, which we call the *Self-Attentive Neural Collaborative Filtering* (SA-NCF), incorporates novel self-attentive hidden layers for the enablement of training really deep ($\geq$ 20 layered) neural networks for the recommendation problem.
- We conduct extensive experiments on six benchmark datasets from Yelp, Yahoo Music and Amazon Reviews. Experimental results show that our proposed SA-NCF outperforms a myriad of strong neural CF baselines by a wide margin. When compared with a vanilla MLP, our proposed method is capable of approximately doubling ($1.8x$ fold on average) the performance on two ranking metrics. Performance gains are also reasonably wide, ranging from 23% to 58% in absolute percentage points and $1.3x$ to $2.8x$ in terms of relative performance as compared to a wide range of strong neural baselines.
- We conduct qualitative studies and provide some analysis pertaining to the inner workings of our proposed approach.

2 BACKGROUND

This section aims to provide a background to our work, touching on key works that are relevant to ours.

2.1 Implicit Collaborative Filtering

This paper tackles the problem of collaborative filtering with *implicit* feedback [3, 22]. The task is often formulated as a learning-to-rank problem and commonly known as *personalized ranking* or one-class CF [30]. In this formulation, user-item interactions are binary in nature (i.e., 1 if observed and 0 if otherwise) as opposed to *explicit* CF where item ratings (e.g., 1-5 stars) are typically the subject of interest. In this formulation, Bayesian Personalized Ranking (BPR) [33] is a well-known pairwise learning algorithm proposed for implicit CF and have seen widespread adoption in many related domains [8, 12, 29, 49]. Early work in recommender systems and collaborative filtering have typically focused on factorization of user-item interactions [17, 24, 25, 28, 32, 34].

2.2 Neural Networks for Recommendation

Today, the state-of-the-art models for recommendation mainly comprises neural models [7, 14, 16, 51].

2.2.1 Neural Models for CF. He et al. [16] cast the well-established matrix factorization algorithm into an overall neural framework, combining the shallow inner-product based learner with a series of stacked nonlinear transformations. More notably, the authors show that their proposed Neural Collaborative Filtering (NCF) framework can outperform many traditional approaches. All in all, the combination of shallow and deep components have been a central motiff in many recent works such as Neural Factorization Machines [13], DeepFM [9] and Wide & Deep [7]. JRL [52] passes the inner product into a MLP for representation learning. The recently proposed Convolutional NCF [14] exploits outer products and 2D convolution layers for learning joint representations of user-item pairs. NeuRec [51] applies a MLP layer on the user-item interaction vector for representation learning. Notably, many of the above mentioned neural models are really not that deep, e.g., NCF, Wide & Deep and JRL uses only three-layered MLPs while Convolutional NCF uses 6-layered 2D Convolution layers. On that note and to the best of our knowledge, there is no neural CF method that goes really deep, i.e., to the tune of at least ten or twenty layers.

2.2.2 Other Neural Recommender Systems. The list of successful neural approaches goes on, along with many specialized neural architectures that were designed to take advantage of various task settings such as sequential and/or review-driven recommeder systems. In the context of sequential recommendation, many popular neural architectures such as Recurrent Neural Networks [18, 47] and Convolutional Neural Networks [38] and Memory Networks [6] have been successfully adapted. Review-based recommender systems are generally based on convolutional text encoders [5, 36]. Today, it is easy to see that CF methods have defaulted to neural approaches, owing not only to ease of implementation but also far superior performance. A comprehensive review of neural recommender systems can be found at [50] .

2.3 Neural Attention Models

The key intuition behind neural attention is to learn to pay attention to only the most important parts of the target. This has been used across a plethora of applications mainly revolving around sentences

in natural language processsing [2] and images in computer vision [26].

2.3.1 Standard Vanilla Attention for Recommendation. The role of attention has been also prominent in the context of recommender systems. He et al. [15] proposed a Neural Attentive Item Similarity (NAIS) model, which uses attention mechanim for learning item-item similarity. Tay et al. [39] proposed a memory-bassed attention module that encourages a flexible translation based metric learning scheme. Wang et al. [46] proposed a tree-based embedding that uses an attention network for explainable recommendation. Note that the above mentioned models utilize a concept of standard vanilla attention, in which a parameterized context vector is used to learn to attend.

2.3.2 Co-Attentive and Self-Attentive Models. Our work is concerned with a new concept known as **self-attention**. Self-attention, most commonly associated with the seminal work of Vaswani et el. [45] was incepted for the purpose of compositional learning of textual sequences. Self-attention can be loosely interpreted as a form of self-targetted co-attention. Different from the standard vanilla attention, co-attention is mainly concerned with the co-learning attentions of two sequences whereby the attention weights of one sequence is conditioned on the other sequence (and vice versa). Co-attention was first incepted in the fields of multi-modal question answering [26] and machine comprehension [48] and is commonly used to learn joint representations of two sequences.

Owing to its not-so-straightforward applicability to recommender systems, it has only started to gain exposure. Namely, a recent work by Hu et al. [21] applies co-attention of meta-paths for top-N recommendation. However, its applicability to textual-based recommender systems remain comfortable, and have been proposed in both review-based [43] and tweet-based [40] recommender systems.

A defining characteristic of co-attentive/self-attentive models is that it enables a greater ease of modeling the global view of the sequence. This is achieved by self-matching, i.e., modeling all elements with all other elements. In the context of NLP, this has been commonly used to imbue the attention mechanism with knowledge of long-range dependencies and have shown great promise in potpourri of text-based applications [31, 37, 41, 42, 44, 45]. In this work, we utilize self-attention to model dependencies and intra-relationships between the hidden layers of a MLP model. To the best of our knowledge, self-attention has primarily been used mainly for sequence modeling. As such, the use of self-attentive MLP layers is novel and has never been attempted before. Moreover, the usage of self-attention in the context of recommender systems is far from straightforward, substantially contributing to the overall novelty of our approach.

3 OUR PROPOSED MODEL

In this section, we introduce our self-attentive neural collaborative filtering framework. Figure 1 illustrates a brief high-level overview of the proposed architecture. Our model accepts three inputs - a user (denoted as p), an item (denoted as q) and another corrupt (negatively sampled) item (denoted as q'). On a whole, our model comprises a single encoder which encodes (p,q) and (p, q') individually. For the sake of brevity and since both positive and negative

sides of the network behaves identically, this section describes the representation learning process of a single user-item pair.

3.1 Embedding Layer

We begin by describing the encoding process of a single user-item pair. The inputs are represented as indices which map onto a user (or item) embedding matrix. The user embedding matrix is parameterized by $E_P \in \mathbb{R}^{d \times |N_P|}$ and the item embedding matrix by $E_Q \in \mathbb{R}^{d \times |N_Q|}$ where N_P, N_Q are the set of all user and items respectively. A look-up layer converts each user-item pair into their respective embeddings. Let $p \in \mathbb{R}^d$ be the user embedding and $q \in \mathbb{R}^d$ be the item embedding and d be the embedding size.

3.2 Hidden Layers

Next, we pass the concatenation[2] of p, q into a ℓ layered multi-layered perceptron (MLP). A MLP comprises multiple hidden layers, with each layer performing a nonlinear transformation over the output of the previous. Formally, this operation is defined as:

$$H_k(x) = ReLU(W_k x + b_k) \tag{1}$$

where $W_k \in \mathbb{R}^{n_{(k-1)} \times n_k}$ and $b_k \in \mathbb{R}^{d_k}$. d_k is the specifed dimension of layer n_k and $n_{(k-1)}$ is the output dimenssion of the $(k-1)$th layer. For simplicity we set the MLP layers to be $n = 2d$ which is also equal to the input representation size (concatenation of p and q). Moreover, while many works adopt a tower structure, i.e., $n_k = (n_{k-1}/2)$, we use a fix size $n_k = n_{k-1}$ for all layers up till layer ℓ where the output is projected to a scalar prediction score.

3.3 Self-Attentive Network

Given a ℓ layered MLP, we would have obtained ℓ intermediate hidden outputs. This layer applies self-attention across all ℓ intermediate states.

3.3.1 Self-Attention Matrix. Firstly, we compute an affinity score between all hidden layers. Let $h_i, \forall i = 1, 2 \cdots \ell$ denote the hidden states of the previously constructed MLP model, the affinity score between hidden states i, j is computed as the inner product between them:

$$S_{ij} = h_i^\top h_j \tag{2}$$

where $S \in \mathbb{R}^{\ell \times \ell}$. Intuitively, matrix S computes intra-feature relationships between all layers of the MLP and is also dubbed as the self-attention matrix. Note that this requires all layers of the MLP to be of equal dimensions which is precisely why we do not use the tower structure. Additionally, when $i = j$, we include a zero mask to pretend a high matching score between identical vectors.

3.3.2 Self-Attentive Representation. Next, we normalize the attention matrix as follows:

$$g_i = \sum_{j=1}^{\ell} \frac{exp(S_{ij})}{\sum_{k=1}^{\ell} exp(S_{ik})} h_j \tag{3}$$

where g_i is the local self-attentive representation of h_i. Intuitively, this operation selects and pools across all feature hierarchies, choosing the most informative layers in the MLP to represent each hidden

---

[2]We also tried concatenating the Hadamard product together with the input vector, i.e., $[p; q, p \odot q]$. However, results were mostly comparable.
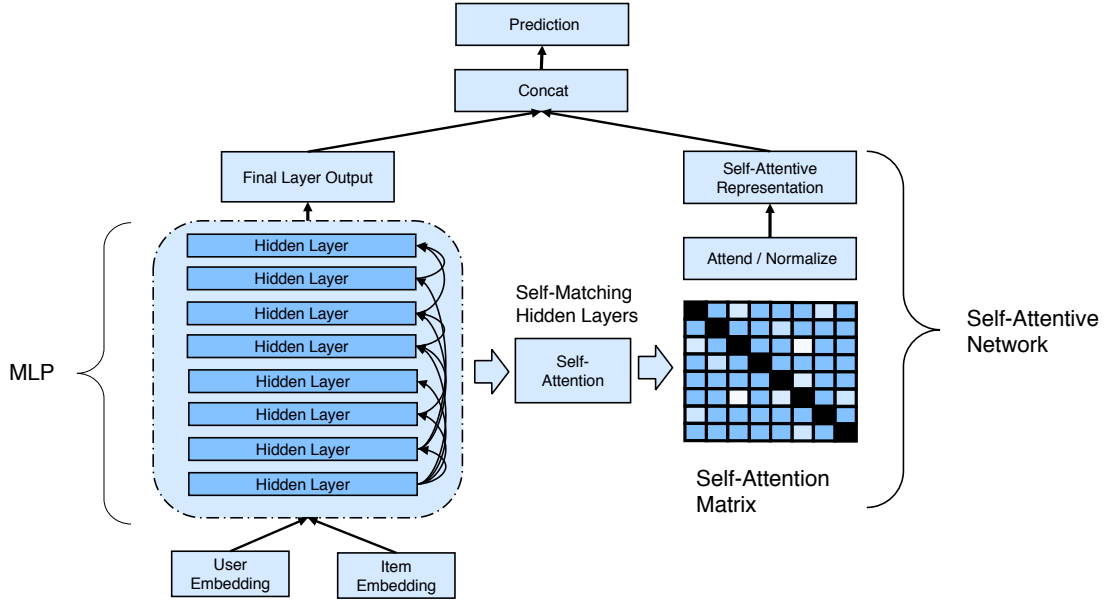
Figure 1: Illustration of our proposed Self-Attentive Neural Collaborative Filtering (SA-NCF) Framework. The shown example has 8 layers which can be increased or decreased. All hidden layers are matched against all other hidden layers to form a self-attention matrix. This self-attention matrix is then used to learn a self-attentive representation that is passed to the output layer for prediction. Note that not all self-matching (curved arrows) are shown due to the lack of space and all hidden layers are actually matched with all other hidden layers.

layer, e.g., g_i is a pooled representation across all MLP layers to represent h_i. In order to learn a single final self-attentive representation, we sum over all local self-attention representations.

$$h_s = \sum_{i=1}^{\ell} g_i \qquad (4)$$

Note that self-attention differs from simply applying regular attention over hidden states as it involves explicit self-matching (i.e., $S_{ij} = h_i^\top h_j$) in order to guide the attention.

3.4 Prediction Layer and Optimization

In order to compute the prediction score between p and q, we concatenate the self-attentive representation with the final layer of the MLP. Next, we pass this concatenated vector into a single linear layer.

$$F(p, q) = W_f([h_s; h_\ell]) + b_f \qquad (5)$$

where $y \in \mathbb{R}$ and $W_f \in \mathbb{R}^{2d \times 1}, b_f \in \mathbb{R}$. In order to train our model, we utilize the BPR (Bayesian Personalized Ranking) [33] loss function.

$$L(\theta) = \sum_{(p, q) \in \Delta} \sum_{(p, q') \notin \Delta} - \log \sigma(F(p, q) - F(p, q')) + \lambda ||\theta||_{\ell 2} \qquad (6)$$

where $F()$ is the scoring function of our self-attentive NCF model, θ are the model parameters, σ is the sigmoid function and λ is the L2 regularization weight.

3.5 Discussion

We delve into several possible explanations to why our model works. There are four key intuitions.

- Each hidden layer is being matched (via the inner product) with every other hidden layer. As such, this effectively connects the entire MLP in a dense fashion, not only shortening the pathway from shallow layers to the prediction layers but also providing a skip-connector from each other layer to every other layer. We hypothesize that this enables our model to scale to a large number of hidden layers, while being able to leverage on this depth to improve performance.
- By attending over feature hierarchies, we also enable a dynamic selection of the most important features for prediction.
- Each hidden layer can be interpreted as a partial joint user-item representation. As such, self-matching enables richer and larger extents of matching interfaces between user-item pairs.
- Since the self-attention mechanism is parameterless (with memory efficiency as a bonus), the network provides an inductive bias such that the hidden representations learn to alter themselves (collectively) in order to generate the most effective/efficient self-attentive representation. This can be interpreted as self/introspective reasoning across layers, where all feature layers come to a consensus pertaining to the hidden representations they create. Our qualitative experiments also depict that the quality of representations are indeed very unlike a standard MLP model.

4 EXPERIMENTS

In order to ascertain the effectiveness of our proposed architecture, we perform a series of quantitative and qualitative experiments. The key research questions (**RQs**) that our experiments are designed to answer are as follows:

(1) **RQ1** - Does our self-attentive NCF framework achieve state-of-the-art performance? Does it outperform other strong competitor baselines?

(2) **RQ2** - What is the effect of the key hyperparameters such as model depth (number of layers) on performance? How deep can self-attentive NCF go?

(3) **RQ3** - What can we understand from the attention maps produced by our self-attentive model? How are representations in SA-NCF different from a regular MLP?

4.1 Datasets

In total, we evaluate our models on six public benchmark datasets. This section describes the data sources and methods to obtain these public datasets. Table 1 reports the overall statistics of datasets used.

- **Yelp Dataset Challenge** - Yelp is a crowdsourced platform for local businesses such as restaurants, bars, salons etc. We use the dataset from the 2018 edition of the Yelp dataset challenge[3]. For this large dataset, we use a 20-core setting, i.e, retaining only users whom have at least more than 20 reviews.

- **Yahoo Music** - This is a dataset obtained from Yahoo Music Services which models a user's preferences for music. This dataset can be obtained via Yahoo! Webscope. We do not perform any filtering on this dataset.

- **Amazon Reviews** - [11, 27] Amazon is an ecommerce platform and is widely used for product recommendation tasks. We use four subsets[4] of the Amazon review corpus, namely *Movies & TV, CDs & Vinyl, Gourmet Food* and *Video Games*.

Dataset	# User	# Item	# Interaction	Density (%)
Yelp 2018	22040	18447	999745	2.45 x 10-3
Yahoo Music	1540	1000	334903	2.17 x 10-1
Movies & TV	123960	50023	1449613	2.34 x 10-4
CDs & Vinyl	75258	64393	947076	1.94 x 10-4
Gourmet Food	14681	8687	121892	9.56 x 10-4
Video Games	24304	10640	183174	7.08 x 10-4

Table 1: Statistics of datasets used in our experiments.

4.2 Compared Methods

This section describes the baselines used for comparisons in our experiments.

- **Matrix Factorization (MF)** is a well-known shallow baseline for CF which scores each user item pair using the dot product $\sigma(p^\top q)$.

- **Factorization Machines (FM)** [32] is another well known baseline for CF. In this baseline, we pass the concatenation of user and item embedding into the FM model.

- **Collaborative Metric Learning (CML)** [20] is a strong competitive shallow baseline for implicit CF. It models the relationship between p and q using the L2 distance, i.e., $|p-q|_{\ell_2}$.

- **Multi-layered Perceptron (MLP)** is the standard deep neural network model with ReLU activations. p and q are concatenated and passed into the neural model. The number of MLP layers are tuned and reported at fixed intervals as described in subsequent sections. Each layer of the MLP is represented as $f_\ell(x) = ReLU(W_\ell(x_{\ell-1}) + b_\ell)$. We compare with two settings of MLP. The first is denoted as MLP (Tower) and has a similar structure to [16]. The second considers hidden layers of equal dimensions up till the final prediction layer, e.g., $128 \rightarrow 128 \rightarrow \cdots 128 \rightarrow 1$.

- **Joint Representation Learning (JRL)** [52] passes the inner product between p and q into a tower-structured multi-layered perceptron. This deep model remains identical to the MLP model except for two major differences. Firstly, $p \odot q$ is passed into the first layer of the MLP instead of $[p; q]$. Secondly, eLU activations are used as opposed to ReLU units, following the original paper. We use a 3-layered tower JRL model.

- **Neural Matrix Factorization (NeuMF)** [16] is a strong deep neural baseline that combines the inner product of p and q together with a side MLP network. The scoring function of NeuMF is denoted as $\sigma(h^\top [p \odot q; MLP([p; q])])$. Following the original work, we use the proposed tower structure and reference the original source code[5] in our implementation.

- **Convolutional NCF** (ConvNCF) [14] is a recent work that exploits the outer product between user and item embeddings for representation learning. It uses multiple stacked 2D convolutional layers for learning features from the outer product matrix. For ConvNCF model, we use the released sourcecode[6] by the authors and adapted it to our experimental framework and overall settings. The original ConvNCF uses six CNN layers. We tune this as a hyperparameter from [3, 6].

For fair comparison, all models optimize the BPR (Bayesian personalized ranking) loss.

4.3 Experimental Setup

This section reports our evaluation protocol and implementation details.

4.3.1 Evaluation Protocol. We adopt the leave-one-out evaluation setting used in many works [16, 39]. For each user, the last item is withheld as the testing sample while the penultimate is used as the development set. For each test sample, we evaluate the ranking metrics nDCG@10 and hit ratio (HR@10) by ranking the ground truth against 100 random negative samples that the user has not interacted with before. We train all models for a maximum of 500 epochs with early stopping, i.e., premature stopping if nDCG@10

---

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

[4]Datasets are obtained from http://jmcauley.ucsd.edu/data/amazon/ using the 5-core setting. Note that the choice of subsets were based largely on several factors including size and domain diversity. Due to the lack of space, we do not report results on all subsets.

[5]https://github.com/hexiangnan/neural_collaborative_filtering

[6]https://github.com/duxy-me/ConvNCF

	Yelp 2018		YahooMusic		Movies & TV		CDs and Vinyl		Gourmet Food		Video Games	
	nDCG	H@10	nDCG	H@10	nDCG	H@10	nDCG	H@10	nDCG	H@10	nDCG	H@10
MF-BPR	0.548	0.763	0.405	0.552	0.418	0.597	0.536	0.693	0.284	0.409	0.436	0.594
FM	0.562	0.768	0.411	0.539	0.499	0.558	0.513	0.686	0.194	0.286	0.233	0.336
CML	0.395	0.596	0.368	0.518	0.348	0.534	0.315	0.463	0.241	0.395	0.385	0.556
MLP (Tower)	0.547	0.768	0.355	0.516	0.449	0.612	0.453	0.618	0.267	0.400	0.347	0.500
JRL (Tower)	0.616	0.639	0.368	0.505	0.510	0.600	0.457	0.617	0.211	0.326	0.297	0.422
NeuMF	0.547	0.760	0.411	0.549	0.467	0.634	0.452	0.616	0.285	0.420	0.391	0.542
ConvNCF	0.577	0.702	0.415	0.448	0.496	0.579	0.467	0.530	0.194	0.292	0.282	0.402
MLP (ℓ=3)	0.538	0.764	0.359	0.503	0.465	0.631	0.467	0.637	0.252	0.383	0.358	0.518
MLP (ℓ=12)	0.543	0.759	0.377	0.533	0.466	0.632	0.467	0.629	0.254	0.385	0.336	0.492
MLP (ℓ=20)	0.546	0.765	0.368	0.519	0.529	0.594	0.496	0.535	0.294	0.384	0.414	0.464
MLP (ℓ=24)	0.538	0.759	0.358	0.510	0.510	0.600	0.457	0.617	0.255	0.396	0.394	0.432
SA-NCF (ℓ=3)	0.701	0.884	0.484	0.602	0.615	0.738	0.574	0.707	0.401	0.522	0.529	0.664
SA-NCF (ℓ=12)	<u>0.810</u>	<u>0.977</u>	0.627	0.956	0.762	0.991	0.630	0.946	0.626	0.961	0.642	<u>0.884</u>
SA-NCF (ℓ=20)	0.769	**1.000**	<u>0.757</u>	<u>0.985</u>	<u>0.854</u>	**1.000**	**0.818**	**0.999**	<u>0.774</u>	**1.000**	<u>0.805</u>	**1.000**
SA-NCF (ℓ=24)	**0.876**	**1.000**	**0.797**	**0.994**	**0.880**	**1.000**	<u>0.811</u>	<u>0.970</u>	**0.825**	<u>0.990</u>	**0.860**	**1.000**
Improvement (+)	+26%	+23%	+38%	+44%	+35%	+37%	+28%	+31%	+53%	+58%	+42%	+41%
Improvement (x)	1.4x	1.3x	1.9x	1.8x	1.9x	1.6x	1.5x	1.4x	2.8x	2.4x	2.1x	1.7x

Table 2: Experimental results (nDCG@10 and HR@10) on six benchmark datasets. Best result is in boldface and second best is underlined. SA-NCF achieves state-of-the-art performance, outperforming many strong neural CF baselines by a wide margin. The last section shows the absolute ($+$%) and relative improvement (x) of SA-NCF compared to the best performing baseline.

score on the development set does not improve after 25 epochs. On most datasets, most models converge before 100 epochs. However, this is not a concern since we save the model only when the development score is topped. Models are evaluated on the development set at fixed intervals of 5 epochs. The final reported score is choosen from the best performing model on the development set.

4.3.2 Implementation Details. We implement our models in Tensorflow [1]. We optimize all models with the Adam Optimizer [23] with a learning rate of 0.001. The batch size is fixed at 512 and the L2 regularization is tuned amongst $\{10^{-4}, 10^{-6}, 10^{-8}\}$ across the board. We tuned the dropout keep rate in the range of $p_d = \{0.8, 0.9, 1.0\}$ but found that in most cases, no dropout ($p_d = 1.0$) performed the best. All model parameters are initialized with the default xavier initializer. We fix the embedding dimension of all models to 64 for fair comparison. Similarly, the number of negative samples is set to 2 across the board. All models learn their parameters from scratch and no pretraining is done. This is observe the effect and relative effectivenes of the raw model architectures. We scale the loss function by dividing the batch size since we found it to be slightly more stable. For MLP and SA-NCF, we report results at varying depths, i.e., $\{3, 12, 20, 24\}$ in order to study the effects of model depth on performance.

4.4 Experimental Results

Table 2 reports our experimental results on six benchmark datasets. Firstly, we observe that SA-NCF outperforms all other baselines. Notably, the performance gains over the strongest baseline is reasonably large with performance gains of +23% (Yelp 2018) all the way up to +58% (Gourmet Food). On average, SA-NCF improves the score of the strongest baseline by approximately 1.8 folds, with

the largest being a 2.8 fold (almost three times) increase in performance score. Secondly, SA-NCF achieves a perfect[7] HR@10 score (1.000) on four out of six datasets. On the remaining two, the score remains almost close to perfect (e.g., 0.994). Thirdly, we find that the superior performance of SA-NCF generally applies universally across all datasets, irregardless of size or density.

We also observe that increasing the depth of the SA-NCF model brings notable differences in performance. This ties down to the primary claim of this work, i.e., a conceptually simple and efficient way to enable training deep neural CF models that do not plateau easily. Furthermore, we can observe that, in most cases, a larger number of layers correlates to a much better performance. Finally, on most datasets, we also observe that ℓ=24 is the most optimal setting, achieving a huge margin over many competitor baselines.

When varying model depth, SA-NCF is also consistently superior to the MLP model. We observe that, across all six datasets, the MLP model does not benefit from increasing the number of layers. A surprising and notable observation is the fact that SA-NCF (ℓ=3) actually outperforms MLP (ℓ=3) by quite a bit (easily $\geq$ +10%) even though the self-attention mechanism has only two hidden layers to work with. Nevertheless, this shows the overall power of this new paradigm, empirically proving that SA-NCF does not necessarily need to go deep to benefit from the self-attention mechanism.

Additionally, a plethora of baselines are provided for reference. We make several observations. Firstly, we find that MF-BPR and FMs are an extremely tough baselines to beat. CML, on the other

---

[7]Note that HR@10 measures how often the ground truth label is found within the top-10 items ranked by the model, ranked against 100 negative samples. Hence, unlike other metrics such as accuracy/F1 score, it is actually very possible to achieve perfect HR@10. While this may seem like the task is too easy, we would argue that this is not the case since it is evident that many baselines still struggle with achieving a score comparable to our model.

hand, seems to be underperforming. Secondly, NeuMF and ConvNCF gets better performance only on some datasets. For these complex models, we note that more finetuning of hyperparameters such as L2 regularization and initialization schemes could possibly lead to a better performance. However, we choose to keep the hyperparameter searches consistent across all models in the spirit of fairness. Thirdly, we observe that there is no clear winner between JRL (tower) and MLP (tower), with their relative rankings ocassionally exchanged.

As a final sectional closing note and recapitulation, our SA-NCF model outperforms all baselines by a wide margin. We find that increasing the model depth strongly correlates to a better model performance, in contrast to the standard MLP model that plateaus at lower depths. Overall performance gains and empirical results of SA-NCF are extremely promising.

4.5 Model Efficiency and Parameter Size

Table 3 reports the runtime and parameter size of all models on the Yelp dataset challenge. Firstly, we note that SA-NCF does not incur any significant cost over the MLP model with a larger number of hidden layers. At $\ell = 3$, SA-NCF takes approximately 1.4 times of a MLP of equal number of layers. This is understandable given that the self-attention can be seen as a kind of overhead cost. However, at $\ell = 12$ and $\ell = 20$, the runtimes are almost identical, with only a $1 - 2$ seconds difference. As such, the cost of self-attention becomes negligible[8] with deeper networks, which is overall complementary to it's initial purpose/design.

In terms of parameter size, the parameter cost of MLP and SA-NCF are almost identical because the self-attention does not incur any parameter cost. Moreover, in neural CF models, majority of the parameter cost often comes from the user and item embeddings. As such, ramping up the layers of the neural CF model often incur negligible costs in the overall scheme of things. Finally, we note that NeuMF has the highest parameter costs, largely due to utilizing dual embedding spaces. On the other hand, CML and MF-BPR only contain parameters from the user-item embedding. Expectedly, CML and MF-BPR are the most efficient models, since they only use relatively simple matching functions such as the dot product and L2 distance.

Model	Time / Epoch (s)	# Param
MF-BPR	16	2.59M
FM	39	2.59M
CML	13	2.59M
MLP (tower)	33	2.62M
JRL (tower)	33	2.59M
NeuMF	70	5.21M
ConvNCF	55	2.61M
MLP (ℓ=3/12/20)	41/66/84	2.62M/2.77M/2.91M
SA-NCF (ℓ=3/12/20)	59/67/86	2.62M/2.77M/2.91M

Table 3: Runtime (time/epoch) and model parameter size on Yelp 2018 dataset with batch size 512, embedding size 64 and 2 negative samples. Benchmarked on a TitanXP GPU.
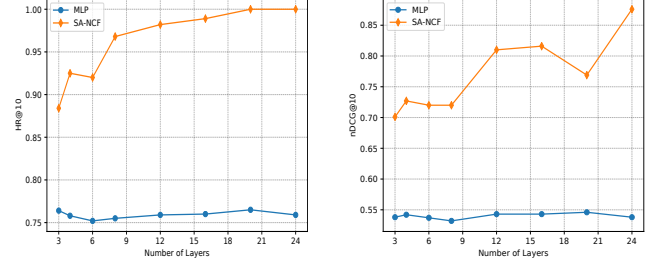
4.6 Effect of Model Depth

Figure 2: Effect of model depth (ℓ) on performance on Yelp 2018 dataset. Unlike standard MLPs, SA-NCF enables training deep networks without hitting an early plateau.

In this section, we study the effect of model depth on performance. Figure 2 charts the effect of ℓ against HR@10 (left) and nDCG@10 (right) on Yelp 2018 dataset. On both metrics, we observe the while MLP plateaus and achieves rather identical scores from layers 3 all the way up till 24, SA-NCF is capable of overcoming this limit. More concretly, it scales it performance with the number of layers, hitting the best scores at $\ell = 24$.

5 MODEL ANALYSIS AND DISCUSSSION

In this section, we delve into an in-depth model analysis, aiming to further understand behaviour of our model. Moreover, our goals is to extract network behaviour that can possibly help us to understand how SA-NCF learns and generalizes by observing how hidden representations change within the network. Note that this is fundamentally very **different** from trying to make recommendations explainable to end users.

5.1 Self-Attention Visualisation

Figure 3 shows the visualisation of the self-attention matrix on the Yelp dataset. We trained a SA-NCF model with 20 layers[9]. Several observations can be made. Firstly, we observe that different user-item pairs have different self-attention activations. This is intuitive given that a different joint representation would invoke different self-matching patterns. Secondly, we observe that the strongest activations are often from adjacent layers. However, it is good to note that there are also reasonably strong activation from far away layers, i.e., $(1, 11), (1, 12)$ in the *right* example. Finally, we note that all in all, that hidden layers in general (far or near) have relatively high similarity with other layers, made evident by the relatively high intensity scores across the self-attention matrix. This is relative to the extracted inner products between a standard MLP model which we will discuss in the following section.

---

[8]Note that the dot product of all hidden layers in our self-attention is efficiently handled by a simple matrix multiplication operation of two tensors. GPUs make this process extremely fast.

[9]Note that this corresponds to only 19 self-matching layers, since the last layer is the prediction layer.
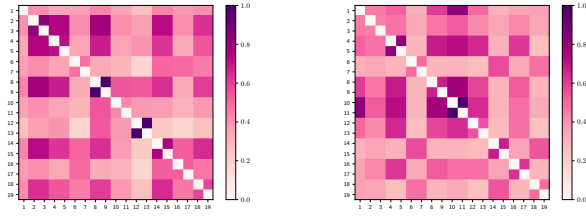
Figure 3: Visualisation of the normalized Self-Attention Matrix of SA-NCF ($\ell = 20$) with two unique user-item pairs on Yelp Dataset. Higher intensity denotes a higher attention weight. Each value represents the attention weight between two hidden layers (x and y axis) of the network.

5.2 Effect of Self-Attention on Hidden Representations

In order to study the effect of our self-attention on the hidden representations, we extracted all hidden layers from a 20 layered **standard** MLP model. Following which, we computed the inner product between all hidden layers of the MLP model. This is to enable comparison and vividly observe the impacts of the self-attention. Figure 4 shows the result of this processs.
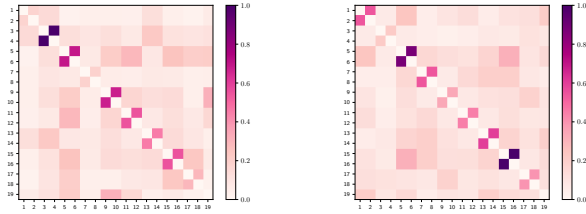

Figure 4: Visualisation of the normalized inner product similarity between hidden layers of a standard MLP model with $\ell = 20$.

Firstly, we observe a stark contrast with the inner product similarity between layers in our SA-NCF model. This can be easily noticed by comparing Figure 3 with Figure 4. In the MLP model, it is intuitive that the strongest activations are almost soley focused on consecutive layers. While there is still some similarity between far-away layers, they are remarkedly less prominent as compared to that of SA-NCF. As such, we would be able to deduce that the self-attention mechanism of NCF contributes to altering the representation learning process of the MLP model, forcing all layers to come to a global 'agreement' while maintaining a feed-forward architecture.

5.3 Layer Importance Analysis

Our self-attention mechanism allows us to inspect layer importance. We trained SA-NCF with $\ell = 6$. After which, we extracted the self-attention matrices of all training samples. Intuitively, the self-attention matrix allows us to inspect how much a layer contributes

to the overall self-attentive representation. This provides a brief idea about how important that layer actually is. More specifically, by taking the *argmax* of the row-summed self-attention weights, we are able to determine, for a given user-item sample which layer has the highest importance. To the best of our knowledge, this is the first attempted study to understand how MLPs learn in the context of neural CF and recommendation.

Figure 5 illustrates the layer importance with respect to training epochs. Note that layer importance is defined by the number of training samples (user-item pairs) that regard a particular layer as the most important. At this point, we make a very interesting observation - the behaviour of layer importance is approximately **identical** even across different datasets and domains[10], hinting at the possibility that SA-NCF (and in general MLPs) learn to generalize in the same way irregardless of domain/dataset.

In general, the importance of the final layer always declines. This is understandable given that the final layer is also passed to the prediction layer, defeating the purpose of attentively selecting it. The first layer always starts off at a significantly higher position, but gives way for the *center-most* layer (in this case, layer 3). Towards the end of training, layer 3 converges to roughly equal importance as layer 1. We also observed that, for other values of ℓ (mostly higher), a common trend is that layer $\ell/2$ is always the second most important layer. Notably, for $\ell = 10$, we observe that the layers are often ranked in this order $[1, 5, 7, 3, 6, 2, 8, 4, 9]$, which is surprisingly similar to the pattern $[1, 3, 2, 4, 5]$ in Figure 5. This suggests that there is a universal pattern that SA-NCF/MLP might follow within the context of CF **regardless** of dataset/domain. All in all, understanding how MLPs encode and generalize from interaction data is indeed an interesting line of work which is beyond the scope of this paper.
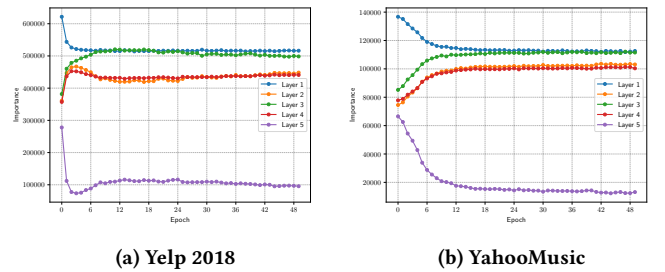

(a) Yelp 2018   (b) YahooMusic

Figure 5: Layer importance analysis on of SA-NCF ($\ell = 6$) two datasets.

6 CONCLUSION

We proposed Self-Attentive Neural Collaborative Filtering, a new way to train MLPs up to twenty layers and beyond without plateauing. SA-NCF achieves highly competitive results, outperforming many neural baselines by a wide margin. We train a deep SA-NCF of 24 layers, achieving perfect HR@10 scores on 4 out of 6 datasets. SA-NCF outperforms all existing methods by almost two fold on all datasets, along with a $23\% - 58\%$ improvement.

---

[10]Note that all six datasets exhibit similar patterns but we only show two due to the lack of space.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). Software available from tensorflow.org.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A Generic Coordinate Descent Framework for Learning from Implicit Feedback. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*. 1341–1350.

[4] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 46–54.

[5] Rose Catherine and William Cohen. 2017. TransNets: Learning to Transform for Recommendation. *arXiv preprint arXiv:1704.02298* (2017).

[6] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential Recommendation with User Memory Networks. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 108–116.

[7] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.

[8] Vachik S Dave, Baichuan Zhang, Pin-Yu Chen, and Mohammad Al Hasan. 2018. Neural-Brane: Neural Bayesian Personalized Ranking for Attributed Network Embedding. *arXiv preprint arXiv:1804.08774* (2018).

[9] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. *arXiv preprint arXiv:1703.04247* (2017).

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[11] 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.

[12] Ruining He and Julian McAuley. 2016. VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback.

[13] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. (2017).

[14] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. 2018. Outer Product-based Neural Collaborative Filtering. (2018).

[15] Xiangnan He, Zhenkui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. NAIS: Neural Attentive Item Similarity Model for Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2018).

[16] 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.

[17] 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*.

[18] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[19] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2, 5 (1989), 359–366.

[20] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge J. Belongie, and Deborah Estrin. 2017. Collaborative Metric Learning. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*. 193–201. https://doi.org/10.1145/3038912.3052639

[21] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and S Yu Philip. 2018. Leveraging Meta-path based Context for Top-N Recommendation with A Neural Co-Attention Model. *Proceedings of KDD 2018* (2018).

[22] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*. 263–272. https://doi.org/10.1109/ICDM.2008.22

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

[24] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 426–434.

[25] Yong Liu, Peilin Zhao, Xin Liu, Min Wu, Lixin Duan, and Xiaoli Li. 2017. Learning User Dependencies for Recommendation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. 2379–2385. https://doi.org/10.24963/ijcai.2017/331

[26] 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.

[27] 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.

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

[29] Wei Niu, James Caverlee, and Haokai Lu. 2018. Neural Personalized Ranking for Image Recommendation. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18)*. ACM, New York, NY, USA, 423–431. https://doi.org/10.1145/3159652.3159728

[30] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 502–511.

[31] 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.

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

[33] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*. 452–461.

[34] Jasson DM Rennie and Nathan Srebro. 2005. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*. ACM, 713–719.

[35] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*. 285–295. https://doi.org/10.1145/371920.372071

[36] 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)*.

[37] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2017. Disan: Directional self-attention network for rnn/cnn-free language understanding. *arXiv preprint arXiv:1709.04696* (2017).

[38] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. (2018).

[39] 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

[40] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. CoupleNet: Paying Attention to Couples with Coupled Attention for Relationship Recommendation. *arXiv preprint arXiv:1805.11535* (2018).

[41] 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

[42] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-Cast Attention Networks for Retrieval-based Question Answering and Response Prediction. *arXiv preprint arXiv:1806.00778* (2018).

[43] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-Pointer Co-Attention Networks for Recommendation. *arXiv preprint arXiv:1801.09251* (2018).

[44] Yi Tay, Luu Anh Tuan, Siu Cheung Hui, and Jian Su. 2018. Reasoning with Sarcasm by Reading In-between. *arXiv preprint arXiv:1805.02856* (2018).

[45] 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).

[46] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2018. Tem: Tree-enhanced embedding model for explainable recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1543–1552.

[47] 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.

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

[49] Baichuan Zhang, Sutanay Choudhury, Mohammad Al Hasan, Xia Ning, Khushbu Agarwal, Sumit Purohit, and Paola Pesntez Cabrera. 2016. Trust from the past: Bayesian personalized ranking based link prediction in knowledge graphs. *arXiv preprint arXiv:1601.03778* (2016).

[50] Shuai Zhang, Lina Yao, and Aixin Sun. 2017. Deep learning based recommender system: A survey and new perspectives. *arXiv preprint arXiv:1707.07435* (2017).

[51] 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).

[52] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W Bruce Croft. 2017. Joint representation learning for top-n recommendation with heterogeneous information sources. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management.* ACM, 1449–1458.