6-2019

# Deep Learning for Recommender Systems

Travis Akira Ebesu

# SANTA CLARA UNIVERSITY

Department of Computer Engineering

Date: June 2019

I HEREBY RECOMMENDED THAT THE THESIS PREPARED UNDER

DR. YI FANG BY

**Travis Akira Ebesu**

ENTITLED

**Deep Learning for Recommender Systems**

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

**DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE & ENGINEERING**

[illegible]
Thesis Advisor
Dr. Yi Fang

B. Dezfouli
Thesis Reader
Dr. Behnam Dezfouli

[illegible]
Chairman of Department
Dr. Nam Ling

Yuhong Liu
Thesis Reader
Dr. Yuhong Liu

Haibing Lu
Thesis Reader
Dr. Haibing Lu

[illegible]
Thesis Reader
Dr. Weijia Shang

# Deep Learning for Recommender Systems

**by**

**Travis Akira Ebesu**

**Dissertation**

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Computer Science & Engineering
in the School of Engineering at Santa Clara University, 2019

Santa Clara, California

*Dedicated to my family...*

# *Acknowledgements*

First and foremost, I would like to express my sincere gratitude to my advisor Prof. Yi Fang for the continuous guidance, encouragement and enthusiasm throughout my doctoral studies. Everything composed in this thesis has been the fruit fostered by him from the late nights before paper submissions to long hours of mentoring. He provided an environment which cultivated my skills not only as a researcher but also in personal development. It has been a privilege to work with such an excellent mentor and role model.

I would also like to thank my doctoral committee consisting of Prof. Behnam Dezfouli, Prof. Yuhong Liu, Prof. Haibing Lu, and Prof. Weijia Shang for their time, effort and valuable suggestions to improve this thesis.

It was a pleasure to work with the other graduate students in our research group. I enjoyed our group meetings, presentations and thoughtful discussions which fostered a better understanding of complex topics.

Finally, I would like to thank my family and friends. My parents, whom provided unconditional love and support throughout my life. To Kellie, your love and patience helped to get me through this long journey.

# Deep Learning for Recommender Systems

Travis Akira Ebesu

Department of Computer Engineering
Santa Clara University
Santa Clara, California
2019

## ABSTRACT

The widespread adoption of the Internet has led to an explosion in the number of choices available to consumers. Users begin to expect personalized content in modern E-commerce, entertainment and social media platforms. Recommender Systems (RS) provide a critical solution to this problem by maintaining user engagement and satisfaction with personalized content.

Traditional RS techniques are often linear limiting the expressivity required to model complex user-item interactions and require extensive handcrafted features from domain experts. Deep learning demonstrated significant breakthroughs in solving problems that have alluded the artificial intelligence community for many years advancing state-of-the-art results in domains such as computer vision and natural language processing.

The recommender domain consists of heterogeneous and semantically rich data such as unstructured text (e.g. product descriptions), categorical attributes (e.g. genre of a movie), and user-item feedback (e.g. purchases). Deep learning can automatically capture the intricate structure of user preferences by encoding learned feature representations from high dimensional data.

In this thesis, we explore five novel applications of deep learning-based techniques to address top-$n$ recommendation. First, we propose Collaborative Memory Network, which unifies the strengths of the latent factor model and neighborhood-based methods inspired by Memory Networks to address collaborative filtering with implicit feedback. Second, we propose Neural Semantic Personalized Ranking, a novel probabilistic generative modeling approach to integrate deep neural network with pairwise ranking for the item cold-start problem. Third, we propose Attentive Contextual Denoising Autoencoder augmented with a context-driven attention mechanism to integrate arbitrary user and item attributes. Fourth, we propose a flexible encoder-decoder architecture called Neural Citation Network, embodying a powerful max time delay neural network encoder augmented with an attention mechanism and author networks to address context-aware citation recommendation. Finally, we propose a generic framework to perform conversational movie recommendations which leverages transfer learning to infer user preferences from natural language. Comprehensive experiments validate the effectiveness of all five proposed models against competitive baseline methods and demonstrate the successful adaptation of deep learning-based techniques to the recommendation domain.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

The widespread adoption of the Internet has led to an explosion in the number of choices available to consumers. Users begin to expect personalized content in modern E-commerce, entertainment, and social media platforms. Recommender systems (RS) have been established as a crucial solution to keep users engaged and satisfied with personalized content in addition to helping users navigate the vast variety of choices.

RS stem from the simple observation that users take recommendations from others. For example, users will read product reviews before deciding to purchase a product. Similarly, employers desire strong letters of recommendations from prospective employees. Users are often faced with the information overload problem as the amount of content available in a given platform expands at an ever increasing rate making it difficult to find an appropriate choice from the large number of items. The recommendation or suggestion is primarily concerned with a decision making process whether it is the next song to listen to, news story to read, movie to watch, or the next job to apply to. RS address this issue by filtering out a few highly relevant items the user may find interesting from the vast number of irrelevant items in the catalog. The system should predict user preferences from past interactions such as viewing a product page (implicit feedback) or writing a review for a movie (explicit feedback). Successful systems span a wide variety of platforms such as Amazon's book recommendations, Netflix's movie recommendations or Pandora's music recommendations [106, 31, 75].

Next, we briefly outline the business value associated with the successful implementation of a RS. We refer to the term 'item' as the product, service, movie etc. being recommended. First and foremost, RS provide value by increasing the number of items sold or user consumption rate leading to a growth in revenue. For instance, entertainment giant Netflix reports 80% of all hours watched stem from recommendations [31]. Second, RS can improve the diversity of items being sold. Having a large catalog of items produces no utility if the items are never bought or consumed. Conversely, the most popular items have limited stock and may lead to unhappy customers if the recommended item is unavailable. The system can carefully balance the current item's availability, popularity and diversity. Third, RS can improve the overall user's experience and satisfaction with personalized and relevant content. An increase in user satisfaction may result in expanding user loyalty and growth in revenue. Maintaining user loyalty leads to an increase in the number of user interactions resulting in better personalization and more accurate recommendations. Lastly, the inferred user preferences can provide insights into other tasks or decision making processes such as forecasting item stock, production or the level of targeted advertisement to specific regions or groups. An ideal system would provide all of the aforementioned qualities. In addition, the system must also be robust to the highly sparse nature of the data while simultaneously maintaining scalability to a large number of users and items, ideally in real-time.

We distinguish three different classes of recommendation algorithms: collaborative filtering (CF), content-based, and hybrid methods which combine the latter two. Below we provide a short overview of each approach. CF is a successful technique to implementing recommendation systems which attempts to identify preferences of users based on the user-item feedback matrix [100, 62]. This matrix is provided in two forms. The first is known as explicit feedback in which the user intentionally provides a rating or review for a given item. While the latter is implicit feedback which is collected automatically typically from clicks or views. Since implicit feedback can be collected automatically and is more abundant, hence is the focus of this thesis. CF has two main approaches, model-based methods and memory/neighborhood-based methods [61, 100]. Model-based methods such as matrix factorization attempts to decompose the user-item feedback matrix by projecting each user and item into a common low dimensional space [63]. Memory or neighborhood-based methods focus on identifying similar groups of users known as neighbors to perform recommendations [40].

Content-based recommenders focus on recommending items similar to the items the user has liked in the past [77], in contrast to CF methods which recommend items based on users with similar preferences. Users and items are represented by some form of auxiliary metadata, typically text or categorical attributes. For example, items could be represented by their textual description such as a movie plot. A user could be represented with their respective profile information such as a self-written biography, age, gender, or location. A simple approach is the vector space model which computes the similarity of item content and the content consisting of the user's past item interactions. Similarity is often computed by the cosine similarity of the term frequency inverse document frequency (TF-IDF) vectors. The items with the highest similarity score are presented to the user.

The hybrid approach combines the above mentioned techniques to integrate the user-item interaction matrix with content to remedy their respective drawbacks. The lack of interaction data from a new user or item into a system leads to the cold-start problem. CF suffers from the inability to handle the cold-start problem which is commonplace in real world systems. On the other hand, content-based systems can leverage the new item's content but may require specialized domain knowledge or feature engineering such as a knowledge base ontology to express special relations between entities such as movies and actors. Below, we motivate the usage of deep learning to remove the burden of a domain expert crafting specialized features while seamlessly integrating nonlinear user-item interactions from past feedback.

Recently, deep learning demonstrated advancements in many research areas obtaining state of the art performance in computer vision [37], question answering [118, 109, 66, 123], learning programs [35], machine translation [2], recommender systems [128, 3] and many other domains [68]. A fundamental reason is the ability to automatically encode learned representation from the data removing the need for feature engineering from domain experts. These learned representations from data typically perform better than the handcrafted versions. Furthermore, neural networks can approximate any function to an arbitrary precision given sufficient capacity [34]. A neural network consists of multiple layers, each performing a simple nonlinear transformation. Each layer learns multiple levels of abstractions starting with coarse structure at the lowest layers and further refinement in subsequent layers. Collectively, the entire network learns a full hierarchy of abstract concepts with increasing complexity. The terminology 'deep learning' refers to the depth or number of nonlinear layers stacked together that are learned in an end-to-end fashion.

RS consist of high dimensional data due to the increasing volume of users and items. Items are often accompanied with some form of rich metadata such as unstructured text from a summary or product description and categorical data such as the genre of a movie. Hence, deep learning can be leveraged to extract rich feature representations from item content in an automated fashion. In conjunction with nonlinear interactions between users and items, more intricate structure of user preferences can be extracted from the high dimensional data. Furthermore, many traditional recommendation algorithms can be expressed in the form of a shallow neural architecture consisting of a single linear layer [39, 121, 122].

## 1.2 Overview

In this thesis, we explore various techniques to integrate deep learning-based methods into the recommender domain. We propose five novel deep learning-based recommendation models each addressing a specific challenge of user personalization. The focus of this thesis is on top-$n$ item ranking with implicit feedback with the exception of Chapter 7 where user feedback is directly extracted from natural language conversations. The proposed models demonstrate the successful integration of various deep learning architectures such as multi-layer perceptrons, denoising autoencoders, memory networks, convolutional neural networks and recurrent neural networks into the RS domain.

We will first contextualize the relationship of each problem addressed at a high level. First, Chapter 3 based on our work [27], introduces nonlinear interactions to extract diverse user preferences in a traditional CF setting. In order to alleviate the problems associated with introducing new items into a system, Chapter 4 based on our work in [26], extends the CF setting to address the item cold-start problem by utilizing content information. Third, to exploit the high availability of contextual information in the CF setting, Chapter 5 based on our work [53], introduces the integration of contextual attributes into user personalization. Fourth, Chapter 6 based on our work [25], shifts towards addressing personalization in a purely content-based setting of context-aware citation recommendation. Finally, Chapter 7 further expands upon the adoption of content-based personalization to a conversational setting. Below we provide a more detailed breakdown of each methodology.

As previously discussed, a popular and successful technique, CF, assumes similar users will consume similar items by establishing the similarity between users and items from past interactions (e.g. clicks, ratings, purchases). The successful integration of deep learning methods in recommendation systems have demonstrated the noticeable advantages of complex nonlinear transformations over traditional linear models [128]. We propose Collaborative Memory Networks (CMN), a deep architecture to unify the two classes of CF models capitalizing on the strengths of the global structure of latent factor model and local neighborhood-based structure in a nonlinear fashion yielding a unified nonlinear hybrid model. Specifically, we fuse a memory component and neural attention mechanism as the neighborhood component. The associative addressing scheme with the user and item memories in the memory module encodes complex user-item relations coupled with the neural attention mechanism to learn a user-item specific neighborhood. Finally, the output module jointly exploits the neighborhood with the user and item memories to produce the ranking score. Stacking multiple memory modules together yield deeper architectures capturing increasingly complex user-item relations. Furthermore, we show strong connections between CMN components, memory networks and the three classes of CF models. Comprehensive experimental results demonstrate the effectiveness of CMN on three public datasets outperforming competitive baselines. Qualitative visualization of the attention weights provide insight into the model's recommendation process and suggest the presence of higher order interactions.

A core problem when employing CF techniques is the item cold-start problem, which is caused by the system's incapability of dealing with new items due to the lack of past relevant transaction history. At a high level the key idea to address this problem is to obtain item latent factors from rating matrix and content matrix respectively and couple them in the shared latent space. These methods extend the traditional matrix factorization models by integrating content information, but the latent representation learned is often not effective especially when the content information is very sparse which is the case for many recommendation tasks where the item descriptions are usually quite short. The ineffectiveness may lie in the fact that these techniques can be viewed as shallow models in capturing latent topics from item descriptions and feedback information by applying simple transformations (often linear) on the observed data, while the ideal latent factors may have more complex relations with the observations.

To address the above challenges, we propose a probabilistic modeling approach called Neural Semantic Personalized Ranking (NSPR) to unify the strengths of deep neural network and pairwise learning. Specifically, NSPR tightly couples a latent factor model with a deep neural network to learn a robust feature representation from both implicit feedback and item content, subsequently allowing our model to generalize to unseen items. We demonstrate NSPR's versatility to integrate various pairwise probability functions and propose two variants based on the Logistic and Probit functions. We conduct a comprehensive set of experiments on two real-world public datasets and demonstrate that NSPR significantly outperforms the state-of-the-art baselines.

Traditional collaborative filtering techniques have demonstrated effectiveness in a wide range of recommendation tasks, but they are unable to capture complex relationships between users and items as prior work does not incorporate contexual information, which is usually largely available in many recommendation tasks. Furthermore, integrating additional features such as contextual information to existing deep learning-based RS often require modifications to create a specialized neural network architecture. We propose a generic deep learning based model for contexual recommendation to seamlessly integrate arbitrary user and item contextual information. Specifically, the model consists of a denoising autoencoder neural network architecture augmented with a context-driven attention mechanism, referred to as Attentive Contextual Denoising Autoencoder (ACDA). The attention mechanism is utilized to encode the contextual attributes into the hidden representation of the user's preference, which associates personalized context with each user's preference to provide recommendation targeted to that specific user. Experiments conducted on multiple real-world datasets on event and movie recommendations demonstrate the effectiveness of the proposed model over the state-of-the-art baseline methods.

We now shift our focus to the task of context-aware citation-recommendation. The accelerating rate of scientific publications makes it difficult to find relevant citations or related work. Context-aware citation-recommendation aims to solve this problem by providing a curated list of high-quality candidates given a short passage of text. Existing literature adopts bag-of-word representations leading to the loss of valuable semantics and lacks the ability to integrate metadata or generalize to unseen manuscripts in the training set. We propose a flexible encoder-decoder architecture called Neural Citation Network (NCN), embodying a robust representation of the citation context with a max time delay neural network, further augmented with an attention mechanism and author networks. The recurrent neural network decoder consults this representation when determining the optimal paper to recommend based solely on its title. Quantitative results on the large-scale CiteSeer dataset reveal NCN cultivates a significant improvement over competitive baselines. Qualitative evidence highlights the effectiveness of the proposed end-to-end neural network revealing a promising research direction for citation-recommendation.

Lastly, we tackle the challenge of performing movie-recommendations in the conversational setting with a generic framework using transfer learning. Specifically, we transfer existing learned item preferences from the large scale MovieLens dataset and apply it to the conversational domain. Since no user interaction history exists we infer user preferences from the natural language conversation and learn a new user latent factor in an online fashion. Experimental results with two different interaction functions confirm the benefits of our framework. In addition, we examine the effect between the two interaction functions and multiple state of the art language model encoders.

## 1.3 Contributions

The contributions of this thesis can be summarized as follows:

- We propose a novel hybrid architecture, Collaborative Memory Network (CMN), which unifies the strengths of the latent factor model and neighborhood-based methods inspired by Memory Networks to address collaborative filtering (CF) with implicit feedback. Comprehensive experiments on three different datasets demonstrate significant performance gains for seven competitive baselines.

- We propose Neural Semantic Personalized Ranking (NSPR), a novel probabilistic generative modeling approach to integrate deep neural network (DNN) with pairwise ranking. With the modeling power of deep learning, we can extract semantic representation from items and couple it with the latent factors learned from implicit feedback. The experiments show that the proposed approach significantly outperforms the competitive baselines on two real-world public datasets.

- We propose Attentive Contextual Denoising Autoencoder (ACDA) based on denoising autoencoders augmented with a context-driven attention mechanism allowing the integration of arbitrary user and item attributes into the hidden representation. Extensive experiments on the tasks of movie-recommendation and Event recommendation validate the effectiveness of the proposed approach against competitive baseline methods.

- We propose a flexible encoder-decoder architecture called Neural Citation Network (NCN), embodying a powerful max time delay neural network encoder further augmented with an attention mechanism and author networks to address context-aware citation-recommendation. Quantitative results on the large-scale CiteSeer dataset reveal NCN cultivates a significant improvement over competitive baselines.

- We propose a generic framework called Recommendation through Conversational Transfer (RCT) to perform movie-recommendations in a conversational setting leveraging transfer learning. We infer user preferences from the natural language conversation and construct a mixture of transferred item latent factors to facilitate learning a new user latent factor in an online fashion. Experimental results validate the advantages of our framework with two different interaction functions and various language model encoders.

## 1.4 Outline

This thesis is organized as follows, Chapter 2 reviews related work of traditional recommendation systems, deep learning, and finally the application of deep learning to recommendation systems. Chapter 3 proposes Collaborative Memory Networks which extends the traditional linear CF methods to a deep nonlinear architecture. To mitigate the issues of the item cold-start problem introduced in the CF setting due to the lack of past interaction data, Chapter 4 introduces Neural Semantic Personalized Ranking to tightly integrate item content (side information) with a latent factor model to address the item cold-start problem. The incorporation of additional metadata often requires a specialized deep learning architectures. Hence, Chapter 5 proposes a generic framework for the seamless integration of arbitrary user and item side information with a denoising auto encoder and attention mechanism. Chapter 6 presents a purely content-based approach to context-aware citation-recommendation under a Seq2Seq framework. Chapter

7 proposes a transfer learning framework to address movie-recommendation in a conversational setting. Finally, we conclude the work in Chapter 8 and highlight promising future directions and challenges on the application of deep learning for recommender systems.

# Chapter 2

# Related Work

In this section, we present three lines of related work. First, we discuss recommendation systems in general. Second, we review specific deep learning-based techniques used in this thesis and finally, we review the application of deep learning to recommendation systems.

## 2.1 Recommendation Systems

This thesis focuses on the item recommendation task under multiple settings: the traditional collaborative filtering (CF) setting; the item cold-start problem; the integration of contextual item information; and context-aware citation recommendation. In the following subsections, we briefly review these categories of the existing work relevant to ours.

### 2.1.1 Implicit Feedback

Matrix factorization has been adapted to learn from implicit feedback for recommendation. Regularized least-square optimization with case weights is proposed in [45] and [88]. One of the most effective techniques is based on Bayesian personalized ranking (BPR) [99] which has been shown to provide strong results in many item recommendation tasks. Several extensions of BPR include pairwise interaction tensor factorization [98], multi-relational matrix factorization [65], and non-uniformly sampled items [28]. Pan and Chen [90] proposed group Bayesian personalized ranking (GBPR) via introducing group preference. Rendle and Freudenthaler [97] incorporate an adaptive sampling method to speed up learning convergence rate by utilizing the fact that the implicit feedback matrix follows a tailed distribution of item popularity.

### 2.1.2 Cold-Start Problem

Cold-start problems are prevalent in recommender systems. They are often alleviated by utilizing content information or some auxilrary information. Standard collaborative filtering methods require past interaction history in order to make recommendations and hence cannot be used. Word-based similarity methods [91] recommend items based on textual content similarity in word vector space. Collaborative Topic Regression (CTR) couples a matrix factorization model with probabilistic topic modeling to generalize to unseen items [115]. Collective matrix factorization (CMF) [105] simultaneously factorizes both rating matrix and content matrix with shared item latent factors. SVDFeature

[12] combines content features with collaborative filtering. The latent factors are integrated with user, item, and global features. SVDFeature demonstrated state-of-the-art performance in benchmark evaluations.

### 2.1.3 Citation Recommendation

Citation recommendation broadly falls in two categories based on the textual context utilized. A local context comprises the immediate text surrounding a placeholder. In contrast, a global context may consist of a paper's title, abstract or full text [4].

Citation recommendation spans a variety of methodologies such as traditional information retrieval, topic modeling, Restricted Boltzmann Machines, collaborative filtering, statistical machine translation (SMT) and neural networks [4, 49]. Huang et al. [48] apply topic modeling to a global context combined with a local context CTM [47] for the RefSeer production system. In SMT, a translation model treats the citation context and cited document content as parallel sequences [78, 47, 36]. The objective is to learn an alignment or transition probability the given citation context requires a citation. Lu et al. [78] learn an alignment from the citation context and the corresponding document's text, demonstrating improved performance over information retrieval methods when aligning to the shorter abstract rather than the full body of text. Similarly, Citation Translation Model (CTM) [47] treats each cited document as a token aligning the citation contexts to this reference. In order to address the noisy alignment between citation contexts and documents, He et al. [36] argues the parallel pairs of citation contexts and documents lack a canonical alignment unlike the gold standard present in traditional machine translation. They address the noisy alignment problem by leveraging topical information in their SMT model. More recently, Huang et al. [49] learned a distributed word representation of the citation context and the associated document embedding via a feedforward neural network. In an analogous task, Tan et al. [110] recommend quotes using a listwise learning to rank approach, where the papers are books. A comprehensive survey on citation recommendation can be found in [4].

## 2.2 Deep Learning

In this section, we review literature relevant to the specific deep learning techniques used in this thesis. Recently, deep learning demonstrated advancements in many research areas obtaining state of the art performance in computer vision [37], question answering [118, 109, 66, 123], learning programs [35], machine translation [2], recommender systems [128, 3] and many other domains [68]. In particular, due to the ability to automatically encode learned feature representation and increasingly complex nonlinear transformations.

### 2.2.1 CNN for NLP

Convolutional neural networks (CNNs) demonstrate competitive performance to recurrent neural networks (RNNs) on natural language processing (NLP) tasks yet computationally cheaper by exploiting parallelism. In particular, the max time delay neural network (TDNN) [17] architecture performs a 1-dimensional convolution over a window of words constructing feature detectors followed by a max-pooling layer to extract relevant features from each sequence (time) simultaneously producing a fixed length representation from a variable length sequence. Generalizing the convolution to contain multiple channels, Kim [59] use two word embeddings per a word. One set of pretrained word vectors are fixed while the other set is allowed to be updated during training. Kalchbrenner et al. [56] introduce dynamic $k$-max pooling to adaptively pool features from variable length context window.

### 2.2.2 Neural Machine Translation

Neural Machine Translation (NMT) provides a general framework to address parallel pairs of arbitrary length sequences, where the source sequence is encoded to a fixed length representation followed by a decoder translating this representation to the target sequence conditioned on all previous states one token at a time. The encoder and decoder functions are application specific, in machine translation RNNs are typically used for both the encoder and decoder [16, 2] while in imaging captioning the encoder may be represented as a CNN [124]. Bahdanau et al. [2] propose adding an alignment mechanism or attention model to the encoder-decoder framework alleviating the bottleneck placed on the encoder function to represent the entire source sequence. Instead the attention mechanism assists the decoder by selecting the most useful encoded representation at each timestep. Secondly, the attention mechanism provides interpretability in visualizing the alignment of the source to target sequences.

### 2.2.3 Memory Augmented Neural Networks

We first provide a brief overview of the inner workings of memory-based architectures. Memory augmented neural networks, generally consist of two components: an external memory typically a matrix and a controller which perform operations on the memory (e.g. read, write, and erase). The memory component increases model capacity independent of the controller (typically a neural network) while providing an internal representation of knowledge to track long-term dependencies and perform reasoning. The controller manipulates these memories with either content-based or location-based addressing. Content-based or associative addressing finds a scoring function between the given question (query) and a passage of text, typically the inner product followed by the softmax operation leading to softly reading each memory location [118, 109, 66, 123, 2]. Performing a soft read over the memory locations allows the model to maintain differentiation hence can be trained via backpropagation. The latter type of addressing (usually combined with content-based) performs sequential reads or random access [35].

The initial framework proposed by Weston et al. [118] demonstrated promising results to track long-term dependencies and perform reasoning over synthetic question answering tasks. Sukhbaatar et al. [109] alleviated the strong levels of supervision required to train the original memory network becoming an End-to-End system. The notion of attention is biologically motivated how humans do not uniformly process all information in a given task but focus on specific subsets of information. Attention mechanisms also provide a level of insight into the deep learning black box by visualizing the attention weights [2]. Kumar et al. [66] improve upon the existing architecture by introducing an episodic memory component allowing for multiple passes or consultations of the memory before producing the final answer. The flexibility of the memory network architecture allows it to perform visual question answering [123] and joint task learning for identifying the sentiment and the relation to target entity [71]. Additional tasks were introduced by Dodge et al. [20] such as dialogue based movie recommendation, and question answering. Interested readers may refer to Goodfellow et al. [34] for a more comprehensive treatment on deep learning.

## 2.3 Deep Learning in Recommendation Systems

Recently, a surge of interest in applying deep learning to recommendation systems has emerged. In particular, deep learning allows for learning robust nonlinear representations from data.

Autoencoders have been a popular choice of deep learning architecture for recommender systems [120, 116, 103, 73, 127]. Autoencoders are a feedforward neural network which constructs a compressed representation by forming a bottleneck in the architecture before attempting to recover the model's initial inputs. Essentially, the autoencoder acts as a nonlinear decomposition of the rating matrix replacing the traditional linear inner product in matrix factorization. For example, AutoRec [103] decomposes the rating matrix with an autoencoder followed by reconstruction to directly predict ratings obtaining competitive results on numerous benchmark datasets. Two variants are proposed: user-based (U-AutoRec) and item-based (I-AutoRec). The study evaluates both models on the Netflix dataset and concludes that the item-based (I-AutoRec) version performs better than the user-based (U-AutoRec) model due to the high variance in the number of user ratings. Incorporating corrupt inputs or noise to auto-encoders further improved performance and as a result, many variants utilizing denoising auto-encoders have since emerged. Another example is Collaborative denoising autoencoders (CDAE) [120] which address top-$n$ recommendation by integrating a user-specific bias into an autoencoder demonstrating CDAE can be seen as a generalization of many existing collaborative filtering methods examining both pointwise and pairwise loss functions. Strub and Jeremie [107] establish a methodology capable of training deep architecture of stacked denoising auto-encoders by interpolating the corrupt input and reconstruction error as a loss function. Li et al. [73] adopt a marginalized denoising autoencoder to diminish the computational costs associated with deep learning. Employing two autoencoders, one for item content and the other for user content bridged with user and item latent factors. AutoSVD++ [127] extends the original SVD++ model with a contrastive autoencoder to capture implicit user feedback and auxiliary item content. Collaborative deep learning (CDL) [116] a hierarchical Bayesian model, is proposed to tightly couple deep representation learning for the content information and collaborative filtering for the rating matrix, allowing two-way interaction between the two. Collaborative deep ranking (CDR) [125] later extends CDL to include a pairwise loss.

We now shift our attention to models using multi-layer perceptrons (MLP). CoFactor [74] jointly factorizes the ratings matrix and the shifted positive pointwise mutual information (SSPMI) item embeddings matrix. Factoring the SSPMI matrix has been shown to be equivalent to Word2Vec [82] for item co-occurence embeddings [70]. Neural Network Matrix Factorization (NNMF) [24] take a different approach by replacing the traditional inner product of matrix factorization with a function learned from a feedforward neural network. Similarly, Neural Collaborative Filtering (NCF) [39] partners the output of a MLP concatenated with the latent factors from matrix factorization applying a nonlinear transformation to produce a local interaction before performing the final recommendation. The MLP and matrix factorization each retain separate embedding spaces for the user and item latent factors accommodating the required complexity for the task at hand. Volkovs et al. [114] tightly couple a deep neural network to learning a mapping from content and existing learned latent factors to both the user and item item latent factors to address either the user or item cold-start problem. Cheng et al. [15] tackle mobile app recommendation in the Google Play store by jointly training a generalized linear model and DNN on hand engineered user demographic and implicit app installations. The DNN produces diverse mobile app recommendations while the linear model mediates overgeneralization to irrelevant recommendations. The joint interaction between the two provides recommendation in middle ground between the two. Zhang et al. [130] confront click-through-rates by modifying the input layer of a feedforward neural network to perform different types of transformations over multi-field categorical data. Three transformations are proposed based on factorization machines, restricted boltzman machines (RBM) and denoising auto-encoders.

The sequential nature of RNNs provides desirable properties for time-aware [119] and session-based recommendation systems [42]. For example, Recurrent Recommender Networks (RRN) [119] represent user latent and item latent factors with two RNNs to capture the temporal aspect of movie recommendation. Collectively, the RNNs hidden states represent the user's preference and ratings at each time interval while additional stationary factors are maintained to handle a user's long-standing preferences. While other methods heuristically define relative temporal changes [7, 133] others propose complex and specialized recurrent cells based on the long-short term memory (LSTM) cell and gated recurrent unit (GRU) cell [21, 133, 92]. Jannach and Ludewig [51] interpolate K-Nearest Neighbor (KNN) with the session-based RNN proposed by Hidasi et al. [42] demonstrating further performance gains. Jing and Smola [54] endow an RNN with survival analysis to predict the future return of a given user. The RNN addresses the temporal aspect consulting previous hidden states with the survival rate to address the user's return time. Zheng et al. [131] portray user behavior and item properties with parallel CNNs on user reviews and item reviews respectively, before employing a final shared interaction layer. Chen et al. [13] tackle sequential recommendation with a memory network architecture. A fixed-size queue is used to store previous interactions forming the memory component to capture long-term dependencies.

Van den Oord et al. [111] tackled music recommendation by a two-step approach: matrix factorization is used to obtain the latent factors for items and then conventional CNN is applied to learn feature representation for content information by treating these latent factors as the output. In other words, the deep learning components of their models are deterministic and only loosely coupled with the matrix factorization of the rating matrix. They do not exploit the interactions between content information and ratings. CNN in recommendation systems have also been used to capture localized item feature representations text [104, 58] and images [128]. CNN overcomes the bag-of-words limitation by learning weight filters to identify the most prominent phrases within the text. Zhang et al. [126] leverage textual, structural and visual knowledge bases with convolutional and denoising auto-encoders to enhance the latent factor model.

The attention mechanism has been widely adopted in deep learning for tasks related to image recognition and natural language processing [124, 2] and is more recently being explored in recommender systems. Gong and Zhang [32] perform hashtag recommendation with a CNN augmented with an attention channel to concentrate on the most informative (trigger) words. However, a hyperparameter must be carefully set to control the threshold of triggering the word to be informative. Huang et al. [46] tackle the same task with an End-to-End Memory Network [109] integrating a hierarchical attention mechanism over the user's previous tweets on a word and sentence level. Chen et al. [11] incorporate multimedia content with an item level attention representing the user preferences and a component level attention to isolate item specific visual features. Similarly, Seo et al. [104] introduce a local and global attention mechanism over convolutions to model review text. Xiao et al. [122] extend Factorization Machines [96] with an attention mechanism to learn the importance of each pairwise interaction rather than treating them uniformly.

Additional work include, Salakhutdinov et al. [102] addressing CF with a two layer RBM modeling tabular movie ratings and Georgiev and Nakov [30] later extended this work in a unified non-IID framework. Neural Autoregressive Density Estimator (NADE) obtained state-of-the-art performance by modeling an ordinal cost [132]. Wang et al. [117] unify the generative and discriminative methodologies under the generative adversarial network [33] framework for web search, item recommendation, and question answering. Interested readers may find comprehensive surveys on deep learning for recommender systems in [128, 3].

# Chapter 3

# Collaborative Memory Networks

## 3.1 Introduction

In this chapter, we present a novel deep learning-based model named Collaborative Memory Networks to capture complex and nonlinear user-item interactions. Recall, Collaborative Filtering (CF) can generally be grouped in three categories: memory or neighborhood-based approaches, latent factor models and hybrid models [100, 61]. Memory or neighborhood-based methods form recommendations by identifying groups or neighborhoods of similar users or items based on the previous interaction history. The simplicity of these models such as item $K$-nearest neighbor (KNN) have shown success in production systems at Amazon [75, 106]. Latent factor models such as matrix factorization project each user and item into a common low dimensional space capturing latent relations. Neighborhood methods capture local structure but typically ignore the mass majority of ratings available due to selecting at most $K$ observations from the intersection of feedback between two users or items [61]. On the other hand, latent factor models capture the overall global structure of the user and item relationships but often ignore the presence of a few strong associations. The following weaknesses between the local neighborhood-based and global latent factor models lead to the development of hybrid models such as SVD++ [61] and generalizations such as Factorization Machines [96] which integrate both neighborhood-based approaches and latent factor models to enrich predictive capabilities. However, these models are limited in their model capacity due to their linear nature.

The successful integration of deep learning methods in recommendation systems have demonstrated the noticeable advantages of complex nonlinear transformations over traditional linear models [128]. However, existing composite architectures incorporate the latent factor model ignoring the integration of neighborhood-based approaches in a non-linear fashion. Hence, we propose to represent the neighborhood-based component with a Memory Network [118, 109] to capture higher order complex relations between users and items. An external memory permits encoding rich feature representations while the neural attention mechanism infers the user specific contribution from the community.

We propose a unified hybrid model which capitalizes on the recent advances in Memory Networks and neural attention mechanisms for CF with implicit feedback. The memory component allows read and write operations to encode complex user and item relations in the internal memory. An associative addressing scheme acts as a nearest neighborhood model finding semantically similar users based on an adaptive user-item state. The neural attention mechanism places higher weights on specific subsets of users who share similar preferences forming a collective neighborhood summary. Finally, a nonlinear interaction between the local neighborhood summary and the global latent factors[1] derives the ranking score. Stacking multiple memory components allows the model to reason and infer more precise neighborhoods further improving performance.

Our primary contributions can be summarized as follows:

- We propose Collaborative Memory Network (CMN) inspired by the success of memory networks to address implicit collaborative filtering. CMN is augmented with an external memory and neural attention mechanism. The associative addressing scheme of the memory module acts as a nearest neighborhood model identifying similar users. The attention mechanism learns an adaptive nonlinear weighting of the user's neighborhood based on the specific user and item. The output module exploits nonlinear interactions between the adaptive neighborhood state jointly with the user and item memories to derive the recommendation.
- We reveal the connection between CMN and the two important classes of collaborative filtering models: the latent factor model and neighborhood-based similarity model. Furthermore, we reveal the advantages of the nonlinear integration fusing the two types of models yielding a hybrid model.

[1]We use the terms user/item latent factors, memories and embeddings interchangeably.

- Comprehensive experiments on three public datasets demonstrate the effectiveness of CMN against seven competitive baselines. Multiple experimental configurations confirm the added benefits of the memory module [2].

- Qualitative visualizations of the attention weights provide insight into the memory component providing supporting evidence for deeper architectures to capture higher order complex interactions.

## 3.2 Collaborative Filtering

Generally speaking, there are two main categories of recommendation tasks: rating prediction and item recommendation. The objective of rating prediction is to predict the rating that a user may give to an item that she has not interacted with before. Movie rating prediction in Netflix [6] is such an example. For item recommendation, recommender systems provide a user with a ranked list of items that she might prefer. Examples include product recommendation in Amazon [75, 106] and point-of-interest recommendation in location-based social networks [14].

Collaborative filtering (CF) is a popular and effective technique to perform user specific personalized item recommendations from previous user interactions. The Netflix Prize popularized and spurred many advancements in CF techniques. A core underlying assumption is that like minded users will consume similar items. Fundamentally, CF

[2]Source code available at: `http://github.com/tebesu/CollaborativeMemoryNetwork`

is concerned with two main entities: users and items. In this section, we introduce the core concepts associated with CF and the three approaches in CF.

If users have similar preferences in the past, then recommendations from these other users should be of interest as well. These similar preferences are inferred from a central concept to CF, user feedback. This feedback can be in the form of implicit or explicit. In the explicit case, the user provides a direct signal of preference such as rating a movie from 1 to 5 or providing a product review. In implicit feedback, the user provides an indirect signal of preferences. For example, if a user has viewed an item we record this in a binary fashion where positive feedback as a 1 otherwise a 0. Since implicit feedback can be collected at a larger scale automatically from user views/clicks it is more pervasive and abundant in practice. In this thesis, we focus on the setting of item ranking (top$-n$ recommendation) with implicit feedback.

### 3.2.1 Latent Factor Models

We first introduce some notation. The set of all users in the recommender system is denoted as $\mathcal{U}$ and similarly the set of all items as $\mathcal{I}$. A given user $u \in \mathcal{U}$, we use $i^+ \in \mathcal{I}_u^+$ to denote a positive item (i.e., interacted/observed item) where $\mathcal{I}_u^+$ is the set of all positive items for user $u$. Similarly, we use $i^- \in \mathcal{I} \setminus \mathcal{I}_u^+$ for a negative item (i.e., uninteracted/unobserved item). For a given user $u$ we denote the user's latent factor as a $d$ dimensional vector $\mathbf{m}_u \in \mathbf{M}$ and similarly, item $i$'s latent factor $\mathbf{e}_i \in \mathbf{E}$ where the entire matrix of user latent factors $\mathbf{M} \in \mathbb{R}^{|\mathcal{U}| \times d}$ and item latent factors $\mathbf{E} \in \mathbb{R}^{|\mathcal{I}| \times d}$.

The key intuition behind latent factor models are to disentangle latent features automatically from user feedback. Each user and item are projected into a shared low dimensional latent space. The item latent factor may represent concrete variations such as a genre of a movie or more subjective aspects such as a product's ease of use or to a completely hidden and uninterpretable latent structure. On the other hand, the user latent factors indicate the user's level of interest (or lack of) to each of the corresponding item latent dimensions. The user's affinity for a given item is measured as the similarity between the user-item latent feature interactions.

More specifically, user $u$'s ranking score for item $i$ is defined as the inner product between item $i$'s latent factor $\mathbf{e}_i$ and the user's latent factor $\mathbf{m}_u$

$$\hat{r}_{ui} = \mathbf{m}_u^\intercal \mathbf{e}_i \tag{3.1}$$

### 3.2.2 Neighborhood-based Similarity Models

Neighborhood-based or memory-based approaches provide recommendations based on all the feedback in the system or over a specific group of users' feedback known as neighbors. This is implemented in two main approaches: user-based or item-based. In the user-based variant, neighbors represent the level of interest of the target user to an item using ratings from other users' feedback for this item. The latter type performs recommendations based on the ratings of the current user. The primary difference between the two is the definition of the neighbors. We focus on the user-based variant but equivalently swapping the users and items yields the item-based approach.

Formally, neighborhood-based similarity methods estimate a user-user similarity matrix $\mathbf{S} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{U}|}$ in the user-based variant [3]. For each user who rated item $i$ (neighborhood) the item's ranking score is the sum of similarity score Typically, the neighborhood is restricted to be the weighted combination of the $K$ most similar users based on the similarity matrix $\mathbf{S}$. The neighborhood performs two roles, to select the trusted neighbors and identify their contribution to the final recommendation. The general ranking score of neighborhood similarity models for user $u$ and item $i$ are:

$$\hat{r}_{ui} = \alpha_i \sum_{v \in N(i)} \mathbf{S}_{uv} \tag{3.2}$$

where $N(i)$ the neighborhood of all users who provided implicit feedback for item $i$ and $\alpha_i$ is a normalization term to smooth the ranking score against the potentially large and variable size of the neighborhood. A common normalization scheme is $\alpha_i = |N(i)|^{-\rho}$ where $\rho$ is a hyperparameter controlling the level of similarity required to obtain a high score. The similarity matrix $\mathbf{S}$ can be computed in various ways, we first introduce a common heuristic method. A given user $u$'s observed preferences are a $k$-hot encoded vector denoted as $\mathbf{x}_u \in \mathbb{R}^Q$. Each dimension of the vector represents an item, i.e. if user $u$ has observed item $i$, $\mathbf{x}_{ui} = 1$ otherwise $\mathbf{x}_{ui} = 0$. A common similarity metric

[3]Equivalently estimating the item-item similarity matrix $\mathbf{S} \in \mathbb{R}^{|\mathcal{I}| \times |\mathcal{I}|}$ yields the item-based variant

frequently used in information retrieval is the cosine similarity [79] computed in vector space.

$$\mathbf{S}_{uv} = \cos(\mathbf{x}_u, \mathbf{x}_v) = \frac{\mathbf{x}_u^\intercal \mathbf{x}_v}{||\mathbf{x}_u||_2 ||\mathbf{x}_v||_2} \tag{3.3}$$

where $||\cdot||_2$ denotes the $l_2$ vector norm. Other heuristic similarity metrics include Pearson Correlation, adjusted cosine similarity and Spearman Rank Correlation [87, 63]. Instead of using a heuristic method to compute the similarity matrix another approach is to learn the similarity function for $\mathbf{S}$ [86, 55].

### 3.2.3 Hybrid Models

Previously, the two main approaches to CF was introduced however, each with their own drawbacks. Neighborhood methods capture local structure but typically ignore the mass majority of ratings available due to the inclusion of only a subset of interactions [61]. On the other hand, latent factor models capture the overall global structure of the user and item relationships but often ignore the presence of a few strong associations. The following weaknesses between the local neighborhood-based and global latent factor models lead to the development of hybrid models such as SVD++ [61] and generalizations such as Factorization Machines [96] which integrate both neighborhood-based approaches and latent factor models to enrich predictive capabilities.

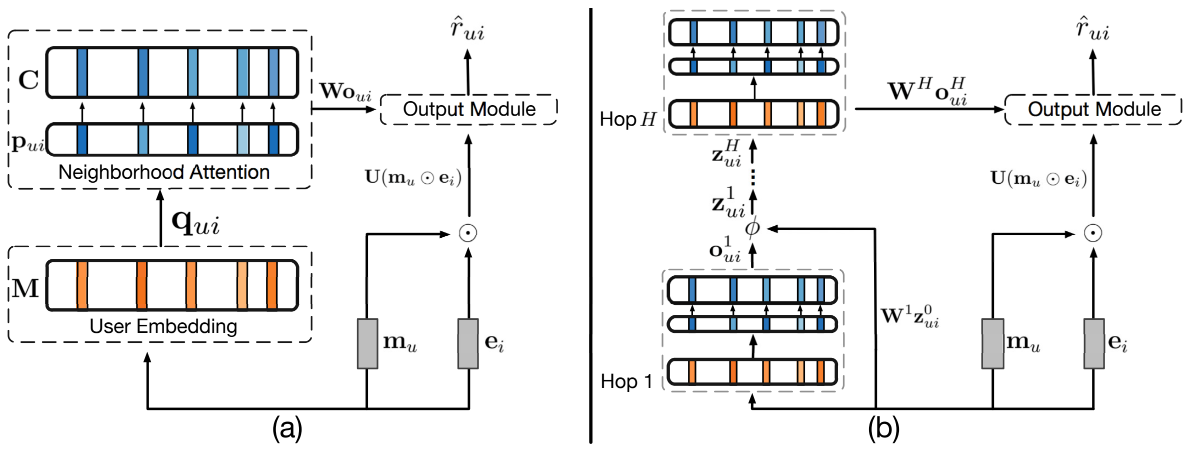


FIGURE 3.1: Proposed architecture of Collaborative Memory Network (CMN) with a single hop (a) and with multiple hops (b).

We can combine the latent factor model and the neighborhood-based model together yielding

$$\hat{r}_{ui} = \mathbf{m}_u^{\mathsf{T}}\mathbf{e}_i + \alpha_i \sum_{v \in N(i)} \mathbf{S}_{uv} \tag{3.4}$$

The first term captures the global interactions from the latent factor model while the second term captures the local neighborhood structure for a holistic view of all user feedback. Inherently, traditional CF models are linear and lack the ability to model more complex user-item interactions. We now explore methods to overcome this limitation with the integration of nonlinear interactions via deep learning.

## 3.3 Collaborative Memory Network

In this section, we introduce our proposed model Collaborative Memory Network (CMN), see Figure 3.1a for a visual depiction of the architecture. At a high level, CMN maintains three memory states: an internal user-specific memory, an item-specific memory, and a collective neighborhood state. The architecture allows for the joint nonlinear interaction of the specialized local structure of neighborhood-based methods and the global structure of latent factor models. The associative addressing scheme acts as a nearest neighbor similarity function that learns to select semantically similar users based on the current item. The neural attention mechanism permits learning an adaptive nonlinear weighting function for the neighbor model, where the most similar users contribute higher weights at the output module. We later extend the model to a deeper architecture by stacking multiple hops in Section 3.3.4 depicted in Figure 3.1b.

### 3.3.1 User Embedding

The memory component consists of a user memory matrix $\mathbf{M} \in \mathbb{R}^{|\mathcal{U}| \times d}$ and an item memory matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{I}| \times d}$, where $|\mathcal{U}|$ and $|\mathcal{I}|$ represents the number of users and items respectively and $d$ denotes the size (dimensionality) of each memory cell. Each user $u$ is embedded in a memory slot $\mathbf{m}_u \in \mathbf{M}$ storing her specific preferences. Similarly, each item $i$ corresponds to another memory slot $\mathbf{e}_i \in \mathbf{E}$ encoding the item's specific attributes. We form a user preference vector $\mathbf{q}_{ui}$ where each dimension $q_{uiv}$ is the similarity of the target user $u$'s level of agreement with user $v$ in the neighborhood given item $i$ as:

$$q_{uiv} = \mathbf{m}_u^\intercal \mathbf{m}_v + \mathbf{e}_i^\intercal \mathbf{m}_v \quad \forall\, v \in N(i) \tag{3.5}$$

where $N(i)$ represents the set of all users (neighborhood) who have provided implicit feedback for item $i$. We would like to point out $N(i)$ could be replaced or combined with $R(i)$ to handle the case of explicit feedback where $R(i)$ denotes the set of all users who provided explicit feedback for item $i$. The intuition is as follows, the first term computes compatibility between the target user and the users who have rated item $i$. The second term introduces the level of confidence user $v$ supports the recommendation of item $i$. Hence, the associative addressing scheme identifies the internal memories with the highest similarity of the target user $u$ with respect to neighborhood of users given the specific item.

### 3.3.2 Neighborhood Attention

The neural attention mechanism learns an adaptive weighting function to focus on a subset of influential users within the neighborhood to derive the ranking score. Traditional neighborhood methods predefine a heuristic weighting function such as Pearson correlation or cosine similarity and require specifying the number of users to consider [100]. While factorizing the neighborhood partially alleviates this problem, it is still linear in nature [55]. Instead by learning a weighting function over the entire neighborhood, we no longer need to empirically predefine the weighting function or number of neighbors to consider. Formally, we compute the attention weights for a given user to infer the importance of each user's unique contribution to the neighborhood:

$$p_{uiv} = \frac{\exp(q_{uiv})}{\sum_{k \in N(i)} \exp(q_{uik})} \quad \forall\, v \in N(i) \tag{3.6}$$

which produces a distribution over the neighborhood. The attention mechanism allows the model to focus on or place higher weights on specific users in the neighborhood while placing less importance on user's who may be less similar. Next we construct the final neighborhood representation by interpolating the external neighborhood memory with the attention weights:

$$\mathbf{o}_{ui} = \sum_{v \in N(i)} p_{uiv} \mathbf{c}_v \tag{3.7}$$

where $\mathbf{c}_v$ is another embedding vector for user $v$ which is called external memory in the original memory network framework [118]. Denoting the $v^{th}$ column of the embedding matrix $\mathbf{C}$ with the same dimensions as $\mathbf{M}$. The external memory allows the storage of long-term information pertaining specifically to each user's role in the neighborhood. In other words, the associative addressing scheme identifies similar users within the neighborhood acting as a key to weight the relevant values stored in the memory matrix $\mathbf{C}$ via the attention mechanism. The attention mechanism selectively weights the neighbors according to the specific user and item. The final output $\mathbf{o}_{ui}$ represents a weighted sum over the neighborhood composed of the relations between the specific user, item and the neighborhood.

CMN captures the similarity of users and dynamically assigns the degrees of contribution to the collective neighborhood based on the target item rather than a predefined number of neighbors which may restrict generalization capacity. Furthermore, the attention mechanism reduces the bottleneck of encoding all information into each individual memory slot and allows the joint exploitation of the user and item observations.

### 3.3.3 Output Module

As noted earlier neighborhood models capture the local structure from the rating matrix via the neighbors while latent factor models identify the global structure of the rating matrix [61]. Hence we consider the collective neighborhood state to capture localized user-item relations and the user and item memories to capture the global user-item interactions. The output module smoothly integrates a nonlinear interaction between the local collective neighborhood state and the global user and item memories. Existing models lack the nonlinear interaction between the two terms potentially limiting the extent of captured relations [127, 11]. For a given user $u$ and item $i$ the ranking score is given as:

$$\hat{r}_{ui} = \mathbf{v}^{\mathsf{T}} \phi\big(\mathbf{U}(\mathbf{m}_u \odot \mathbf{e}_i) + \mathbf{W}\mathbf{o}_{ui} + \mathbf{b}\big) \tag{3.8}$$

where $\odot$ is the elementwise product; $\mathbf{W}, \mathbf{U} \in \mathbb{R}^{d \times d}$; and $\mathbf{v}, \mathbf{b} \in \mathbb{R}^d$ are parameters to be learned. We first apply the elementwise product between the user and item memories followed by a linear projection with $\mathbf{U}$, subsequently introducing a skip-connection thus reducing the longest path from the output to input. Skip-connections have been shown to encourage the flow of information and ease the learning process [38]. In this way, the model can better correlate the specific target addresses (user and item memories) with the ranking score to propagate the appropriate error signals. We further motivate this choice by demonstrating its connection to the latent factor model (Section 3.4.1). Similarly, the final neighborhood representation $\mathbf{o}_{ui}$ is projected to a latent space with $\mathbf{W}$ then combined with the previous term followed by a nonlinear activation function $\phi(\cdot)$. Empirically we found the rectified linear unit (ReLU) $\phi(x) = \max(0, x)$ to work best due to its nonsaturating nature and suitability for sparse data [34, 38].

Our proposed model provides the following advantages. First, consider the case where the amount of feedback for a given user is sparse, we can leverage all users who have rated the item to gain additional insight about the existing user and item relations. Second, the neural attention mechanism adjusts the confidence of each user's contribution to the final ranking score dependent on the specific item. Finally, the nonlinear interaction between the local neighborhood and global latent factors provide a holistic view of the user-item interactions.

### 3.3.4 Multiple Hops

We now extend our model to handle an arbitrary number of memory layers or hops. Figure 3.1b (right) illustrates CMN's architecture with multiple hops. Each hop queries the internal user memory and item memory followed by the attention mechanism to derive the next collective neighborhood state vector. The first hop may introduce the need to acquire additional information. Starting from the second hop, the model begins to take into consideration the collective user neighborhood guiding the search for the representation of the community preferences. Each additional hop repeats this step considering the previous hop's newly acquired information before producing the final neighborhood state. In other words, the model has the chance to look back and reconsider the most similar users to infer more precise neighborhoods. More specifically, multiple memory modules are stacked together by taking the output from the $h^{th}$ hop as input to the $(h+1)^{th}$ hop. Similar to [109, 71, 123] we apply a nonlinear projection between hops:

$$\mathbf{z}_{ui}^{h} = \phi(\mathbf{W}^{h}\mathbf{z}_{ui}^{h-1} + \mathbf{o}_{ui}^{h} + \mathbf{b}^{h}) \tag{3.9}$$

where $\mathbf{W}^h$ is a square weight matrix mapping the user preference query $\mathbf{z}_{ui}^{h-1}$ to a latent space coupled with the existing information from the previous hop followed by a nonlinearity and the initial query $\mathbf{z}_{ui}^{0} = \mathbf{m}_u + \mathbf{e}_i$. Intuitively, the initial consultation of the memory may introduce the need for additional information to infer more precise neighborhoods. The nonlinear transformation updates the internal state then solicits the user neighborhood:

$$q_{uiv}^{h+1} = (\mathbf{z}_{ui}^{h})^{\top}\mathbf{m}_v \quad \forall\, v \in N(i) \tag{3.10}$$

The newly formed user preference vector then recomputes the compatibility between the target user and the neighborhood followed by the adaptive attention mechanism producing an updated collective neighborhood summary. This process is repeated for each hop yielding an iterative refinement. The output module receives the weighted neighborhood vector from the last ($H^{th}$) hop to produce the final recommendation.

### 3.3.5 Parameter Estimation

Since our objective is to study implicit feedback which is more pervasive in practice and can be collected automatically (e.g. clicks, likes). In the case of implicit feedback, the rating matrix contains a 1 if the item is observed and 0 otherwise. We opt for the pairwise assumption, where a given user $u$ prefers the observed item $i^+$ over unobserved or negative item $i^-$. The traditional pointwise approach assumes the user is not interested in the item $i^-$ but in reality may not be aware of the item. We can form triplet preferences $(u, i^+, i^-)$ since the number of preference triplets is quadratic in nature we uniformly sample a ratio of positive items to negative items which we further investigate in Section 3.5.7. We leverage the Bayesian Personalized Ranking (BPR) optimization criterion [99] as our loss function which approximates AUC (area under the ROC curve):

$$\mathcal{L} = -\sum_{(u,i^+,i^-)} \log \sigma(\hat{r}_{ui^+} - \hat{r}_{ui^-}) \tag{3.11}$$

where $\sigma(x) = 1/\big(1 + \exp(-x)\big)$ is the logistic sigmoid function. It is worth noting we are not restricted to setting $\sigma(x)$ as the logistic sigmoid function. Other pairwise probability functions such as the Probit function can be used. Since the entire architecture is differentiable, CMN can be efficiently trained with the backpropagation algorithm.

To reduce the number of parameters we perform layerwise weight tying sharing all embedding matrices across hops [109, 71, 123].

### 3.3.6 Computational Complexity

The computational complexity for a forward pass through CMN for a user is $O\big(d|N(i)| + d^2 + d\big)$ where $|N(i)|$ denotes the size of the neighborhood for item $i$ and $d$ is the embedding size. The first term $O\big(d|N(i)|\big)$ is the cost for computing the user preference vector and the latter terms correspond to the final interactions in the output module. Each additional hop introduces $O\big(d|N(i)| + d^2\big)$ complexity. During training two forward passes are computed one for the observed positive item and the second for the negative unobserved item. Parameters can be updated via backpropagation with the same complexity. In real-world datasets, $|N(i)|$ is usually slightly larger than or comparable to $d$, and thus the primary computational complexity is computing $O(d|N(i)|)$. The cost is reasonable since other deep learning methods such as CDAE [120] compute a forward pass in $O\big(|\mathcal{I}|d\big)$ where $|\mathcal{I}|$ is the total number of items. The proposed memory module only computes the similarity with the target user's neighbors (not over all users) and $|N(i)|$ is often less than or comparable to $|\mathcal{I}|$. Thus, the training in CMN is quite efficient. In practice, prefiltering techniques can be used to limit the number of neighbors to the top-$K$ because not all neighbors may be indicative in contributing to the final prediction [100]. Since the purpose of our study is to understand the characteristics of CMN, we leave prefiltering techniques to future work.

Recommendation can be performed by computing the predicted ranking score (Equation 3.8) for a given user and item with a single pass through the network. The item with the highest value is recommended to the user. The computational complexity for runtime recommendation is the same with that of the single forward pass during training.

## 3.4 Relation to Existing Models

CMN consists of components which can be interpreted in terms of the three classes of collaborative filtering methods. We show the connection with the latent factor model and neighborhood-based similarity models, and finally the relation to hybrid models such as SVD++. We conclude the section by drawing parallels between memory networks and CMN.

### 3.4.1 Latent Factor Model

The latent factor model discovers hidden relations by decomposing the ratings matrix into two lower rank matrices. By omitting the neighborhood term and bias, and further setting $\mathbf{U}$ to the identity matrix Equation 3.8 becomes the following:

$$\hat{r}_{ui} = \mathbf{v}^{\mathsf{T}} \phi(\mathbf{m}_u \odot \mathbf{e}_i) \tag{3.12}$$

which leads to a generalized matrix factorization (GMF) model [39]. Removing the nonlinearlity by setting $\phi(\cdot)$ to the identity function and constraining $\mathbf{v}$ to $\mathbb{1}$ vector of all ones, we recover matrix factorization. Under our pairwise loss function (Equation 3.11) we recover BPR [99].

### 3.4.2 Neighborhood-based Similarity Model

The objective of neighborhood-based similarity models are to estimate a user-user[4] similarity matrix $\mathbf{S} \in \mathbb{R}^{P \times P}$. For each user who rated item $i$ an aggregated similarity score produces the confidence of recommending the item. The general form of neighborhood similarity models are:

$$\hat{r}_{ui} = \alpha_i \sum_{v \in N(i)} \mathbf{S}_{uv} \tag{3.13}$$

where $\alpha_i$ is a normalization term to weight the ranking score. In the simplest case, the normalization term is set to $|N(i)|^{-\rho}$ where $\rho$ is a hyperparameter controlling the level of similarity required to obtain a high score. In KNN, the neighborhood $N(i)$ is restricted to be the weighted combination of the $K$ most similar users and the similarity matrix $\mathbf{S}$ is approximated with a heuristically predefined function such as Pearson correlation or cosine similarity. Another approach is to learn the similarity function by approximating $\mathbf{S}$ [86, 55]. In our case, the attached memory module from Equation 3.7 acts as the neighborhood similarity matrix. If we designate the attention mechanism as a predefined normalization term the user-user similarity matrix is then factorized as $\mathbf{S} = \mathbf{C}\mathbf{C}^{\mathsf{T}}$. Using the prediction rule from Equation 3.13 the memory module yields a user-based variant of FISM and under the BPR loss function we recover FISMauc [55].

[4] Equivalently switching users with items yields item-based methods.

### 3.4.3 Hybrid Model

We have shown the connection between the components of CMN and the two classes of collaborative filtering models. Hybrid models such as SVD++ [61] contains two general terms, a user-item latent factor interaction and a neighborhood component. The output module (Equation 3.8) smoothly integrates the latent factors and the similarity or neighborhood terms together leading to a hybrid model.

$$\hat{r}_{ui} = \mathbf{v}^\mathsf{T}\phi\big(\overbrace{\mathbf{m}_u \odot \mathbf{e}_i}^{\text{Latent Factors}} + \underbrace{\sum_{v \in N(i)} p_{uiv}\mathbf{c}_v}_{\text{Neighborhood}}\big) \tag{3.14}$$

We remove the projection matrices and bias terms for clarity. We can see the global interaction from the latent factors consist of the user and item memories. The memory module represents the localized neighborhood component and the neighborhood normalization term is replaced with the adaptive attention mechanism pushed inside the summation becoming a user-item specific weighting scheme. Unlike SVD++, our hybrid model allows for complex nonlinear interactions between the two terms to model the diverse tastes of users.

### 3.4.4 Memory Networks

Traditional memory networks address the task of question answering. A short story or passage of text is provided along with a question for which the answer can be derived by leveraging some form of reasoning. If we pose recommendation as a question answering

| **Dataset** | **Ratings** | **Users** | **Items** | **Sparsity** |
|---|---|---|---|---|
| *Epinions* | 664,823 | 40,163 | 139,738 | 99.98% |
| *citeulike-a* | 204,987 | 5,551 | 16,980 | 99.78% |
| *Pinterest* | 1,500,809 | 55,187 | 9,916 | 99.73% |

Table 3.1: Dataset statistics.

problem we are asking how likely will this user enjoy the item where the user neighborhood is the story and the output ranking score is the answer. Continuing our analogy, each word in the story acts as a user in the neighborhood providing supporting evidence for the recommendation.

## 3.5 Experimental Results

### 3.5.1 Datasets

We study the effectiveness of our proposed approach on three publicly available datasets. The first dataset is collected from *Epinions*[5] [80] which provides an online service for users to share product feedback in the form of explicit ratings (1-5) and reviews. We convert the explicit ratings to implicit feedback as a 1 if the user has rated the item and 0 otherwise. The second dataset is *citeulike-a*[6] [115] collected from CiteULike an online service which provides users with a digital catalog to save and share academic papers. User preferences are encoded as 1 if the user has saved the paper (item) in their library. The third dataset from *Pinterest*[7] [29] allows users to save or pin an image (item) to their

[5]`http://www.trustlet.org/downloaded_epinions.html`
[6]`http://www.cs.cmu.edu/~chongw/data/citeulike/`
[7]`http://sites.google.com/site/xueatalphabeta/`

board indicating a 1 or positive interaction otherwise a 0 and preprocessed according to [39]. Table 3.1 summarizes the statistics of the datasets.

### 3.5.2 Evaluation

We validate the performance of our proposed approach using the leave-one-out evaluation method following the prior work [39, 11, 99]. Closely following the setup from He et al. [39], for each user we randomly hold out one item the user has interacted with and sample 100 unobserved or negative items to form the test set. The remaining positive examples form the training set. If the user has only rated a single item we keep it in the training set to prevent the cold-start setting. We rank the positive item along with the 100 negative items and adopt two common ranking evaluation metrics *Hit Ratio (HR)* and *Normalized Discounted Cumulative Gain (NDCG)* [100]. Intuitively, HR measures the presence of the positive item within the top $N$ and NDCG measures the items position in the ranked list and penalizes the score for ranking the item lower in the list.

### 3.5.3 Baselines and Settings

We compare our proposed approach against seven competitive baselines representing neighborhood-based; traditional latent factor models; hybrid model and deep learning-based models.

- KNN [100] is a neighborhood-based approach computing the cosine item-item similarity to provide recommendations.

- Factored Item Similarity Model (FISM) [55] is a neighborhood-based approach factorizing the item-item similarity matrix into two low rank matrices optimizing the BPR loss function.

- Bayesian Personalized Ranking (BPR) [99] is a competitive pairwise matrix factorization for implicit feedback.

- SVD++ [61] is a hybrid model combining the neighborhood-based similarity and the latent factor model.

- Generalized Matrix Factorization (GMF) [39] is a nonlinear generalization of the latent factor model. We use the ReLU activation function and optimize the BPR loss function.

- Collaborative Denoising Auto Encoder (CDAE) [120] is an item-based deep learning model for item ranking with a user specific bias.

- Neural Matrix Factorization (NeuMF) [39] is a composite matrix factorization jointly coupled with a multilayer perceptron model for item ranking.

We would like to note that FISM [55] improves upon Sparse Linear Methods (SLIM) [86] by factorizing the item-item similarity matrix to handle missing entries hence we do not compare to SLIM. We exclude baselines utilizing additional information for fair comparison since our objective is to study implicit collaborative filtering without content or contextual information.

All hyperparameters are tuned according to the validation set. The validation set is formed by holding out one interaction per user from the training set [39]. We perform a grid search over each model's latent factors from $\{10, 20, 30, 40, 50\}$ and regularization terms $\{0.1, 0.01, 0.001\}$. In addition, we varied CDAE's corruption ratio from $\{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ and NeuMF's layers from $\{1, 2, 3\}$. The number of negative samples is set to 4. Careful initialization of deep neural networks is crucial to avoid saddle points and poor local minima [34]. Thus, CMN initializes the user ($\mathbf{M}$) and item ($\mathbf{E}$) memory embeddings from a pretrained model according to Equation 3.12. Remaining parameters are initialized according to [37] which adapts the variance preserving Xavier initialization [34] for the ReLU activation function. The gradient is clipped if the norm exceeds 5; $l_2$ weight decay of 0.1 with the exception of the user and item memories; and the mini-batch size is set to 128 for *Epinions*, *citeulike-a* and 256 for *Pinterest*. We adopt the RMSProp [34] optimizer with a learning rate of 0.001 and 0.9 for both decay and momentum. The default number of hops is set to 2 and memory or embedding size $d$ to 40, 50 and 50 for the *Epinions*, *citeulike-a* and *Pinterest* datasets respectively. The effects of these hyperparameters are further explored in Sections 3.5.5 and 3.5.7.

| | *Epinions* | | | | *citeulike-a* | | | | *Pinterest* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HR@5 | HR@10 | NDCG@5 | NDCG@10 | HR@5 | HR@10 | NDCG@5 | NDCG@10 | HR@5 | HR@10 | NDCG@5 | NDCG@10 |
| KNN | 0.1549 | 0.1555 | 0.1433 | 0.1435 | 0.6990 | 0.7348 | 0.5789 | 0.5909 | 0.5738 | 0.8376 | 0.3450 | 0.4310 |
| FISM | 0.5542 | 0.6717 | 0.4192 | 0.4573 | 0.6727 | 0.8072 | 0.5106 | 0.5545 | 0.6783 | 0.8654 | 0.4658 | 0.5268 |
| BPR | 0.5584 | 0.6659 | 0.4334 | 0.4683 | 0.6547 | 0.8083 | 0.4858 | 0.5357 | 0.6936 | 0.8674 | 0.4912 | 0.5479 |
| SVD++ | 0.5628 | 0.6754 | 0.4112 | 0.4477 | 0.6952 | 0.8199 | 0.5244 | 0.5649 | 0.6951 | 0.8684 | 0.4796 | 0.5362 |
| GMF | 0.5365 | 0.6562 | 0.4015 | 0.4404 | 0.7271 | 0.8326 | 0.5689 | 0.6034 | 0.6726 | 0.8505 | 0.4737 | 0.5316 |
| CDAE | 0.5666 | 0.6844 | 0.4333 | 0.4715 | 0.6799 | 0.8103 | 0.5106 | 0.5532 | 0.7008 | 0.8722 | 0.4966 | 0.5525 |
| NeuMF | 0.5500 | 0.6660 | 0.4214 | 0.4590 | 0.7629 | 0.8647 | 0.5985 | 0.6316 | 0.7041 | 0.8732 | 0.4978 | 0.5530 |
| CMN-1 | 0.5962 | 0.6943 | 0.4684 | 0.5003 | 0.6692 | 0.7809 | 0.5213 | 0.5575 | 0.6984 | 0.8662 | 0.4960 | 0.5507 |
| CMN-2 | 0.6017† | **0.7007**† | 0.4724† | 0.5045† | **0.7959**† | **0.8921**† | 0.6185† | 0.6500† | 0.7267† | 0.8904† | **0.5180**† | 0.5714† |
| CMN-3 | **0.6020**† | 0.6985† | **0.4748**† | **0.5062**† | 0.7932† | 0.8901† | **0.6234**† | **0.6551**† | **0.7277**† | **0.8931**† | 0.5175† | **0.5715**† |

TABLE 3.2: Experimental results for different methods on the *Epinions*, *citeulike-a* and *Pinterest* datasets. Best results highlighted in bold. † indicates the improvement over baselines is statistically significant on a paired $t$-test ($p < 0.01$).

| | *Epinions* | | | | *citeulike-a* | | | | *Pinterest* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HR@5 | HR@10 | NDCG@5 | NDCG@10 | HR@5 | HR@10 | NDCG@5 | NDCG@10 | HR@5 | HR@10 | NDCG@5 | NDCG@10 |
| CMN | 0.6017 | 0.7007 | 0.4724 | 0.5045 | 0.7959 | 0.8921 | 0.6185 | 0.6500 | 0.7267 | 0.8904 | 0.5180 | 0.5714 |
| CMN-Attn | 0.5807 | 0.6948 | 0.4438 | 0.4809 | 0.7411 | 0.8589 | 0.5503 | 0.5887 | 0.6995 | 0.8773 | 0.4949 | 0.5530 |
| CMN-Linear | 0.5954 | 0.6977 | 0.4660 | 0.4992 | 0.7721 | 0.8665 | 0.5974 | 0.6282 | 0.6992 | 0.8777 | 0.4951 | 0.5534 |
| CMN-Linear-Attn | 0.5830 | 0.6937 | 0.4457 | 0.4816 | 0.7649 | 0.8676 | 0.5922 | 0.6256 | 0.6996 | 0.8775 | 0.4947 | 0.5527 |

TABLE 3.3: CMN variants without attention (CMN-Attn); linear activation with attention (CMN-Linear); and linear without attention (CMN-Linear-Attn).

### 3.5.4 Baseline Comparison

Table 3.2 lists results of CMN with one, two and three hops along with the baselines for HR and NDCG with cut offs at 5 and 10 on the *Epinions*, *citeulike-a* and *Pinterest* datasets. We denote CMN with one hop as 'CMN-1', two hops with 'CMN-2' and so on. At a high-level CMN variants obtain the best performance across both HR and NDCG at all cut offs for all datasets. We now provide a detailed breakdown of our results on the *Epinions* dataset. All baselines with the exception of KNN show competitive performance with each other across all metrics and cut offs. Since CMN shares the same loss function with BPR, FISM and GMF, we can attribute the performance increase to the memory component. The application of a nonlinear transformation does not necessarily help as evident from BPR outperforming its nonlinear counterpart GMF. KNN demonstrated the poorest performance particularly due to the restrictive ability to handle sparse data when only a few neighbors are present and a large number (139k) of items. However, FISM's learned similarity function performs better since it can address missing entries in the item-item similarity matrix. On the other hand, CMN can leverage the global structure of the latent factors encoded in the memory vectors and the additional memory component to infer complex user preferences. Furthermore, CMN's performance gains over CDAE, GMF and NeuMF portray the successful integration of the memory component and attention mechanism over existing nonlinear and deep learning-based methods.

The denser *citeulike-a* dataset contains fewer items than the *Epinions* dataset leading to competitive performance from the item-based KNN method obtaining the strongest baseline NDCG@5. SVD++ outperforms the neighborhood-based FISM and latent factor BPR revealing the effectiveness of combining two approaches into a single hybrid model. The linear decomposition of the item-item similarity matrix in FISM may lack the expressiveness to capture complex preferences as suggested by the nonlinear GMF outperforming the linear BPR. CMN demonstrates improved performance over the fixed neighborhood-based weighting of KNN and the learned linear similarity scheme of FISM indicating the additional nonlinear transformation and adaptive attention mechanism captures more complex semantic relations between users. CMN can be viewed as NeuMF by replacing the memory network component with a multilayer perceptron. CMN outperforming NeuMF further establishes the advantage of the memory network component to identify complex interactions and iteratively update the internal neighborhood state.

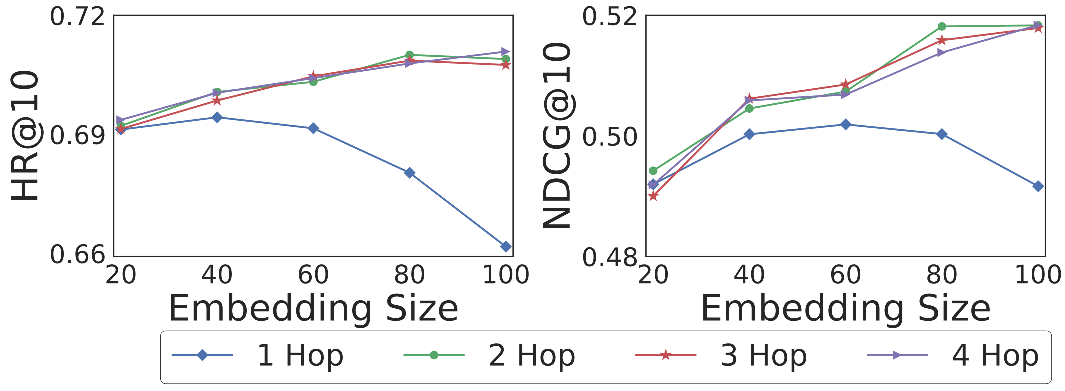
In the *Pinterest* dataset, CMN with a single hop demonstrates competitive performance to baseline methods but with additional hops performance is further enhanced. SVD++ demonstrates competitive performance but may lack the full expressiveness of nonlinearity found in deep learning-based models to capture latent user-item relations. The larger dataset helps the two deep learning baselines to outperform the non-deep learning-based methods but the hybrid nature of CMN allows the joint nonlinear exploitation of the local neighborhood and global latent factors yielding additional performance gains. Overall, CMN with two hops outperforms CMN with a single hop but supplementing additional hops greater than two did not provide significant advantages.

### 3.5.5 Embedding Size

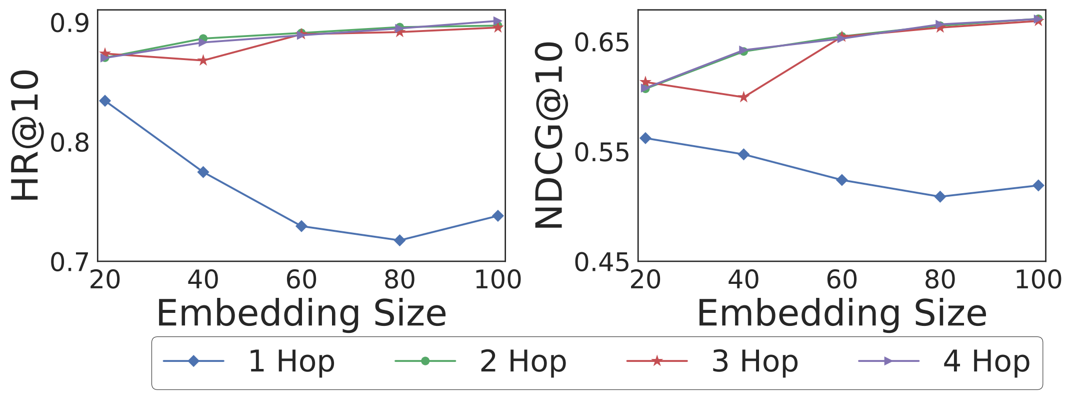
We illustrate the effect of varying the size of memory slots or embeddings and the number of hops for HR@10 and NDCG@10 on the *Epinions* dataset in Figure 3.2a. Since HR and NDCG show similar patterns, we focus our analysis on NDCG. The general trend shows a steady improvement as the embedding size increases with the exception of a single hop where an embedding size of 40 shows peak HR@10 performance followed by a degradation potentially due to overfitting. A single hop confines the model's expressiveness to infer and discriminate between the relevant and irrelevant information encoded in each memory cell. With a small embedding size of 20 increasing the number of hops provides negligible benefits but as the embedding size increases multiple hops show significant improvement over a single hop.

For the *citeulike-a* dataset, Figure 3.2b portrays the best performance of a single hop at an embedding size of 20 followed by a degradation as model capacity increases which is somewhat similar to the *Epinions* dataset. At three hops and an embedding size of 40 shows an unusual drop in performance potentially from finding a poor local minima due to the nonconvex nature of neural networks. Two and four hops show almost identical performance with at most a deviation of 0.3% from each other on HR and NDCG. In general, two hops demonstrates competitive performance against the three and four hop models across all embedding sizes.

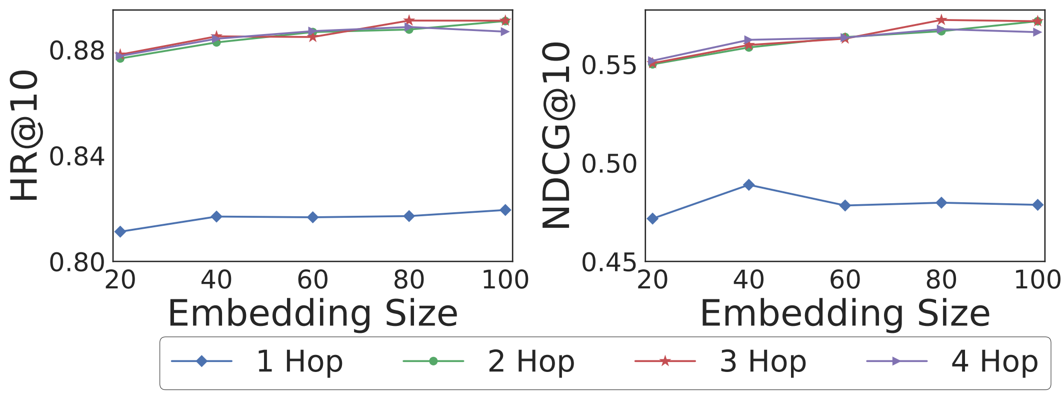
The *Pinterest* dataset in Figure 3.2c shows a similar trend of gradual performance gains as the embedding size increases but a single hop shows insufficient capacity to model



(A) *Epinions* dataset



(B) *citeulike-a* dataset



(C) *Pinterest* dataset

FIGURE 3.2: Experimental results for CMN varying the embedding size from 20-100 and hops from 1-4.

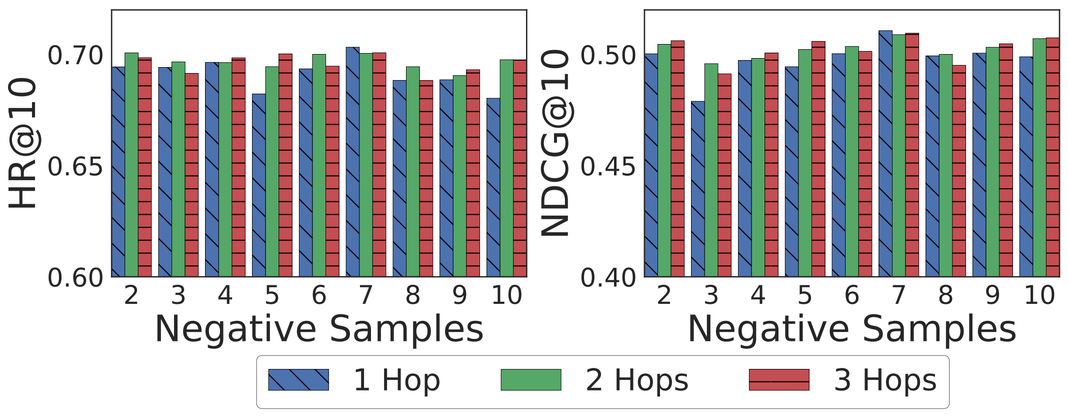
complex user-item interactions. Unlike the results from the previous datasets the performance of a single hop does not degrade as the embedding size increases. The larger dataset may provide some implicit form of regularization to prevent overfitting. Two, three and four hops show similar performance and incremental with larger embedding sizes. Identifying a sufficient number of hops initially takes precedence over the size of the embeddings. With a sufficient number of hops the embedding size can be increased yielding a trade off between computational cost and performance. By introducing additional hops, CMN can better manipulate the memories and internal state to represent more complex nonlinear relations. Consistent with previous results, the addition of more than two hops do not show significant benefit.

### 3.5.6 Effects of Attention and Nonlinearity

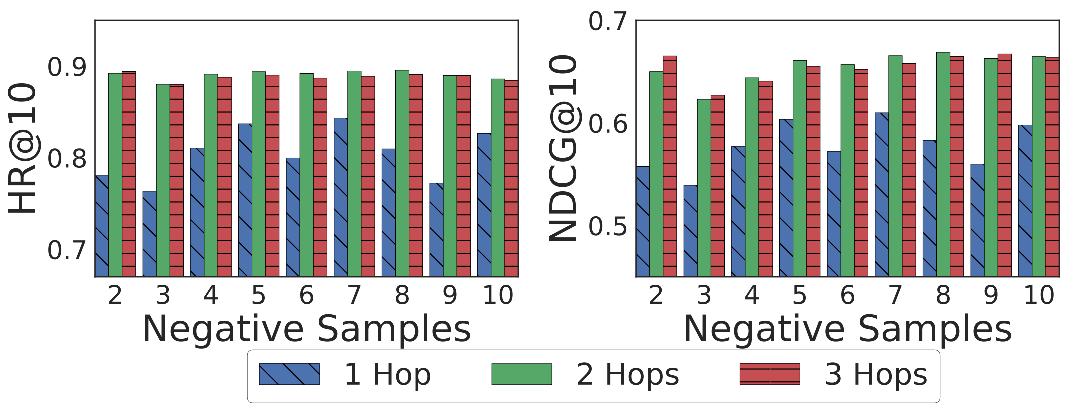
In this section, we seek to further understand the effect of individual components on performance. In Table 3.3, the results for CMN without attention denoted 'CMN-Attn' uniformly performs worse than with attention hinting at the effectiveness of the attention mechanism. We also experimented with a linear version of CMN where all ReLU activation functions are set to the identity function denoted as 'CMN-Linear'. The linear version with the attention mechanism generally outperforms the nonlinear version without attention. Further illustrating the effectiveness of the attention mechanism. The final variation removes the attention mechanism from the linear version denoted as 'CMN-Linear-Attn' which generally performs worse than the linear version with the attention mechanism. In general, removing the nonlinear transformation and attention mechanism in some variation yield similar performance on the *Epinions* and *Pinterest* datasets. In the *citeulike-a* dataset the linear version with and without the attention mechanism show improvements over the nonlinear variant without the attention mechanism. This seems counter intuitive and may indicate a potential difficulty in finding a good local minima or a vanishing gradient problem which is consistent with the unusual drop in performance reported in Section 3.5.5. CMN requires a combination from both the attention mechanism and nonlinear transformations to yield the best performance.

### 3.5.7 Negative Sampling

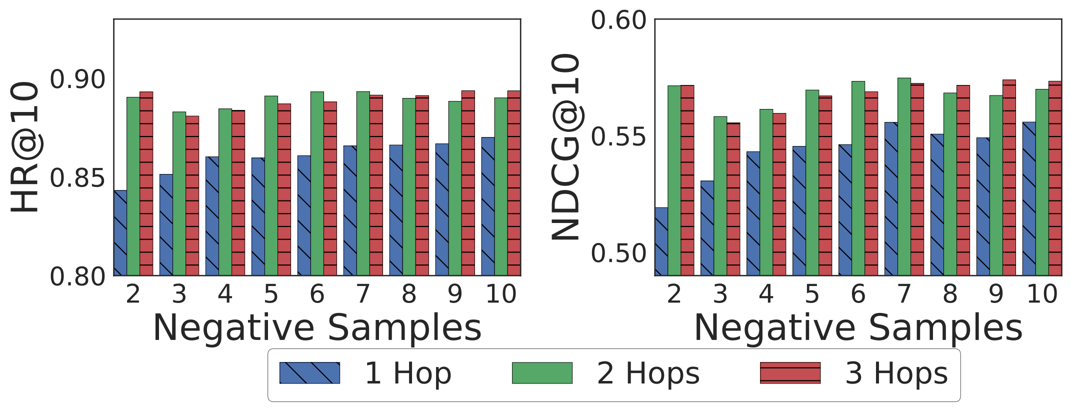
In this section, we study the characteristics of varying the negative samples for CMN reporting HR@10 and NDCG@10. We exclude the results of four hops since the results were consistent with that of three hops. We also omit 1 negative sample since CMN was unable to distinguish between positive and negative samples leading to random performance. Figure 3.3a illustrates the performance of CMN varying the negative samples from 2-10 on the *Epinions* dataset. A single hop shows fluctuations reporting low HR at 5 and 10 negative samples but outperforms the two and three hop versions with 7 negative samples. A single hop uniformly performs worse than the two and three hop counterparts in the *citeulike-a* dataset presented in Figure 3.3b. In Figure 3.3c, the results for a single hop on the *Pinterest* dataset describes a general upward trend where performance improves as the number of negative samples increase. In both the *citeulike-a* and *Pinterest* datasets we observe two and three hops show comparable results and more stability to the number of negative samples while outperforming a



(A) *Epinions* dataset



(B) *citeulike-a* dataset



(C) *Pinterest* dataset

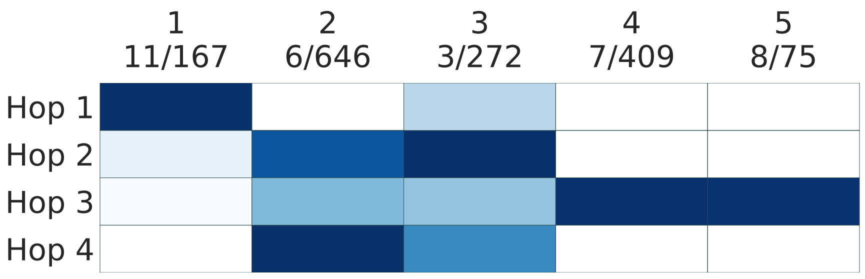
FIGURE 3.3: Experimental results for CMN varying the number of negative samples from 2-10 and hops from 1-3.

single hop. Overall, the performance of CMN is fairly stable with respect to the number of negative samples when at least two hops are present. Similar to the previous section on embedding size, we notice having at least two hops reduces the sensitivity to the hyperparameter.

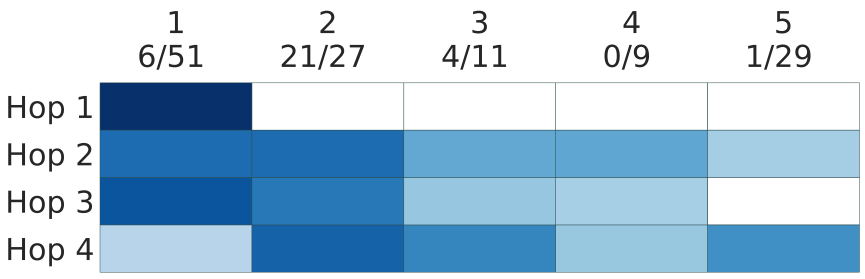
### 3.5.8 Attention Visualization

Attention mechanisms allow us to visualize the weights placed on each user in the neighborhood with the hope of providing interpretable recommendations. We plot a heatmap of the weights from Equation 3.6 in Figure 3.4. The color scale represents the intensities of the attention weights, where a darker color indicates a higher value and lighter colors indicate a lower value. For ease of reference, we label each column representing a user in the neighborhood starting from 1 which may not necessarily reflect the true user id from the dataset. Furthermore, for each user we provide additional context in the form of user statistics. We denote 11/167 to indicate the user has rated a total of 167 items with 11 items observed in common with the target user in the training set. Since the size of the neighborhood can be large we limit the visualization to top 5 neighbors sorted by the highest aggregated attention values. We would like to point out that in some cases the attention weights can be small and hence not visually distinguishable.

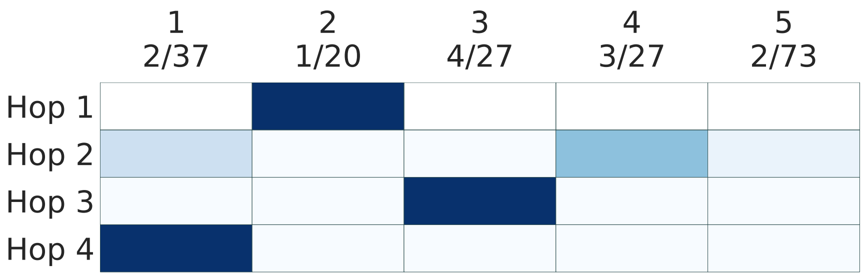
The attention weights for a random user from the *Epinions* dataset is portrayed in Figure 3.4a. The user has a total of 49 neighbors thus we show only the top 5 neighbors



(A) *Epinions* dataset



(B) *citeulike-a* dataset



(C) *Pinterest* dataset

FIGURE 3.4: Heatmap of the attention weights over four hops. The color scale indicates the intensities of the weights, darker representing a higher weight and lighter a lower weight. Each column represents a user in the neighborhood labeled with the number of items in common with the target user / number of ratings in training set. For example, 11/167 indicates user 1 has 11 items corated with the current user $u$ and has rated a total of 167 items in the training set.

due to space constraints. We can see all the top users attended to have at least a single item in common. The first hop places heavy levels of attention on user 1 and lightly on user 3. At two hops the attention on user 3 increases. Progressing to three hops the attention spread out across four users. Finally, at four hops user 2 and 3 have the highest weight with a balance of the number items in common with the target user and overall number of ratings observed suggesting these users may be the most influential in the recommendation process. As shown in previous sections performance generally increases with additional hops suggesting considering a combination of multiple users may be beneficial.

Figure 3.4b illustrates the attention weights over four hops for a random user from the *citeulike-a* dataset with a total of 9 neighbors. We observe the first hop places a large amount of weight on a single user which may explain the poorer performance of CMN with a single hop. User 1 has the highest number of observations out of the neighborhood which may be a reasonable choice but it ignores other information that may be present from other users. Examining the weights of the second hop we see the attention is spread out across five users with a higher emphasis on users 1 and 2 who have the most items in common with the target user. Four out of the five users have a common item with the target user providing a strong indicator of the successful integration of the attention mechanism. Next, we focus on three hops which removes the attention over user 5 and reduces the intensities on user 4, 2 and 3. In the final hop, attention is returned to user 5 with stronger weights than in hop two. The overall attention levels shift around slightly but focus most heavily on user 2 which makes sense since it has the highest number of commonly rated items. Since user 4 has no items in common with the target user but large attention weights this warranted further investigation. We found user 4 to have at least one item in common with all other users in the neighborhood which may explain the attention placed on user 4 despite no corated items with the target user in the training data. This demonstrates the memory component captures higher level interactions within the neighborhood suggesting some form of transitive reasoning.

Figure 3.4c illustrates the attention weights over four hops for a random user from the *Pinterest* dataset. Similar to the previous visualization the first hop places heavy weights on a single user followed by a more dispersed weighting in the following hops. In hops two through four, a small amount of attention is placed upon each user which may not be visually distinguishable. Each hop allows CMN to examine the external memory and perhaps through some form of trial and error arrives at identifying the most useful neighbor as user 1 in hop four. We would like to note the attention mechanism may not necessarily place weights on all users who have rated items in common.

## 3.6 Summary

We introduced a novel hybrid architecture unifying the strengths of the latent factor model and neighborhood-based methods inspired by Memory Networks to address collaborative filtering (CF) with implicit feedback. We reveal the connection between components of Collaborative Memory Network (CMN), the three important classes of CF models, and draw parallels with the original memory network framework. Comprehensive experiments under multiple configurations demonstrate significant improvements over competitive baselines. Qualitative visualization of the attention weights provide insight into the model's recommendation process and suggest higher order transitive relations may be present.

# Chapter 4

# Neural Semantic Personalized Ranking

## 4.1 Introduction

The problem of item cold-start is of great practical importance because modern online platforms publish thousands of new items everyday and effectively recommending them is essential for keeping the users continuously engaged. Content-based approaches, on the other hand, may still produce recommendations by using the descriptions of the items, but they tend to achieve lower accuracy. Combining CF and content becomes a common approach to item cold-start problems. Several hybrid latent factor models were proposed in the literature including collective matrix factorization (CMF) [105] and collaborative topic regression (CTR) [115]. The key idea is to obtain item latent factors from rating matrix and content matrix respectively and couple them in the shared latent space. These methods extend the traditional matrix factorization models by integrating content information, but the latent representation learned is often not effective especially when the content information is very sparse which is the case for many recommendation tasks where the item descriptions are usually quite short. The ineffectiveness may lie in the fact that these techniques can be viewed as shallow models in capturing latent topics from item descriptions and feedback information by applying simple transformations (often linear) on the observed data, while the ideal latent factors may have more complex relations with the observations.

Another challenge in many recommendation tasks is the presence of implicit feedback where users' explicit preferences (e.g., ratings) on items are unavailable. In the real world, often only implicit feedback is available to learn a recommendation model. Examples of implicit feedback are clicks, watched movies, played songs, purchases or assigned tags. Implicit feedback is tracked automatically and thus it is much easier to collect than explicit feedback. A characteristic of implicit feedback is that it is one-class, i.e. only positive observations are available. Moreover, the observed implicit feedback is generally very sparse, which makes the preference modeling even more challenging. As the result, existing solutions often deliver unsatisfactory recommendation accuracies.

On the other hand, deep learning models recently demonstrated great success for learning effective representations in various applications including computer vision, speech recognition, and natural language processing [18, 68]. However, the existing literature contains very few work on developing deep learning models for recommender systems, especially for addressing the cold-start problem with implicit feedback. In this paper, we propose a Neural Semantic Personalized Ranking (NSPR) probabilistic model by learning item representations using a deep neural network (DNN). To handle implicit feedback, we adopt pairwise probability functions that aim to discriminate between a small set of positive items and a very large set of all remaining items. In this way, items both with and without feedback will contribute to learning the ranking function and thus the data sparsity problem can be alleviated. DNN is used to map high-dimensional sparse text features into low-dimensional dense features in a latent semantic space. These low-dimensional features are tightly coupled with the latent factors learned from the pairwise probability, which allows two-way interactions between the content information and implicit feedback. The pairwise probability derived from the implicit feedback can guide the learning of feature representations. The learned features can further improve the predictive power of the pairwise model. The latent factors of new items can be inferred by applying the trained DNN to their content and then be used for item ranking. The contributions of the paper can be summarized as follows:

A popular and effective approach to recommendations is collaborative filtering (CF), which focuses on finding users with similar interests and recommending items favored by the like-minded [62]. One of the fundamental problems arising when employing CF techniques is the item cold-start problem, which is caused by the system's incapability of dealing with new items due to the lack of relevant transaction history.

Recommender systems help users deal with information overload and enjoy a personalized experience on the Web. One of the main challenges in these systems is the item cold-start problem which is very common in practice since modern online platforms have thousands of new items published every day. Furthermore, in many real-world scenarios, the item recommendation tasks are based on users' implicit preference feedback such as whether a user has interacted with an item. To address the above challenges, we propose a probabilistic modeling approach called Neural Semantic Personalized Ranking (NSPR) to unify the strengths of deep neural network and pairwise learning. Specifically, NSPR tightly couples a latent factor model with a deep neural network to learn a robust feature representation from both implicit feedback and item content, subsequently allowing our model to generalize to unseen items. We demonstrate NSPR's versatility to integrate various pairwise probability functions and propose two variants based on the Logistic and Probit functions. We conduct a comprehensive set of experiments on two real-world public datasets and demonstrate that NSPR significantly outperforms the state-of-the-art baselines.

The architecture consists of an item and user auto-encoder for content information coupled with a latent factor model. CDL and DCF models both share some similarities with NSPR. However, they directly predict user ratings and lack the ability to address implicit feedback which is pervasive in modern recommender systems. While CDR does use a pairwise loss for implicit feedback it does not exploit a latent factor model.

NSPR has notable differences from the existing work. First of all, no prior work has studied the cold-start problems by coupling deep semantic representation with user feedback. Secondly, many previous deep models in recommender systems use denoising auto-encoders to learn a feature representation from content, while NSPR utilizes a deep neural network which allows to model the latent semantic space directly without modeling the recovery of input as the auto-encoders do. We demonstrate the effectiveness of using a DNN to learn robust feature representations without complex preprocessing data transformations. Last but not the least, NSPR utilizes stochastic gradient descent for parameter estimation, which is often more scalable for large datasets than the batch estimation methods [8] which were used in the existing deep learning based recommendation models such as CDL and DCF.

## 4.2 Neural Semantic Personalized Ranking

We take a pairwise approach to item recommendation by assuming that a user prefers the items that she has interacted with rather than those items that she has not interacted with. This assumption is more reasonable than the pointwise assumption which treats all observed entries in the user-item interaction/feedback matrix as positive examples and all missing entries as negative examples.

Formally, given user $u \in \mathcal{U}$, we use $i^+ \in \mathcal{I}_u^+$ to denote a positive item (i.e., interacted/observed item) where $\mathcal{I}_u^+$ is the set of all positive items for user $u$. Similarly, we use $i^- \in \mathcal{V} \setminus \mathcal{I}_u^+$ for a negative item (i.e., uninteracted/unobserved item) where $\mathcal{V}$ is the set of all items. Since item $i^+$ is preferred over item $i^-$, we can form a preference instance $(u, i^+, i^-) \in \mathcal{D}_S$ where $\mathcal{D}_S = \{(u, i^+, i^-)|u \in \mathcal{U}, i^+ \in \mathcal{I}_u^+, i^- \in \mathcal{V} \setminus \mathcal{I}_u^+\}$ is the whole set of preference instances. The total number of preference triplets is quadratic in the number of items. Thus, we sample from $\mathcal{D}_S$ for training instead of going over the complete set of item pairs (Section 4.3.3 gives the details about our sampling strategy). Table 4.1 lists the main notations used in the paper.

Table 4.1: Notations

| | |
|---|---|
| $u, i$ | Index for user and item respectively |
| $\mathcal{U}, \mathcal{V}$ | User set and item set respectively |
| $\mathbf{m}_u$ | Latent factor for user $u$ |
| $\mathbf{e}_i$ | Latent factor for item $i$ |
| $\hat{r}_{ui}$ | Ranking score of item $i$ for user $u$ |
| $\mathcal{I}_u^+$ | Set of all positive items for user $u$ |
| $(u, i^+, i^-)$ | A preference triplet indicating user $u$ prefers item $i^+$ over item $i^-$ |
| $\mathcal{D}_s$ | Set of preference triplet instances |
| $\mathcal{D}_u$ | Set of preference instances for user $u$ |
| $\mathbf{x}_i$ | Input content vector for item $i$ |
| $\mathbf{y}_i$ | Output latent feature vector for item $i$ |
| $\mathbf{a}_l$, $\mathbf{W}_l$, $\mathbf{b}_l$ | Activation output, weight matrix and bias vector at the $l^{th}$ layer in DNN respectively |
| $\sigma_m^2$ | Variance in user prior distribution |
| $\sigma_e^2$ | Variance of noise in latent item factor |
| $\sigma_r^2$ | Variance of noise in ranking score |
| $K$ | Number of latent factors |
| $L$ | Number of layers in DNN |
| $N$ | Size of vocabulary |

### 4.2.1 Probabilistic Generative Modeling

We propose Neural Semantic Personalized Ranking (NSPR) by tightly incorporating a deep neural network (DNN) [43] to learn effective feature representation from item content. The DNN architecture maps the raw text features into the features in a semantic space. The input (raw text features) to the DNN is a high dimensional term vector, e.g., TF-IDF of terms in the item content, and the output of the DNN is a concept vector in a low-dimensional semantic feature space. Formally, we denote $\mathbf{x}_i$ as the input term vector, $\mathbf{y}_i$ as the output vector, $l$ as the $l^{th}$ hidden layer ($l \in [1, L-1]$). $\mathbf{a_l}$, $\mathbf{W_l}$ and $\mathbf{b_l}$ are the activation output, weight matrix and bias vector respectively. We have

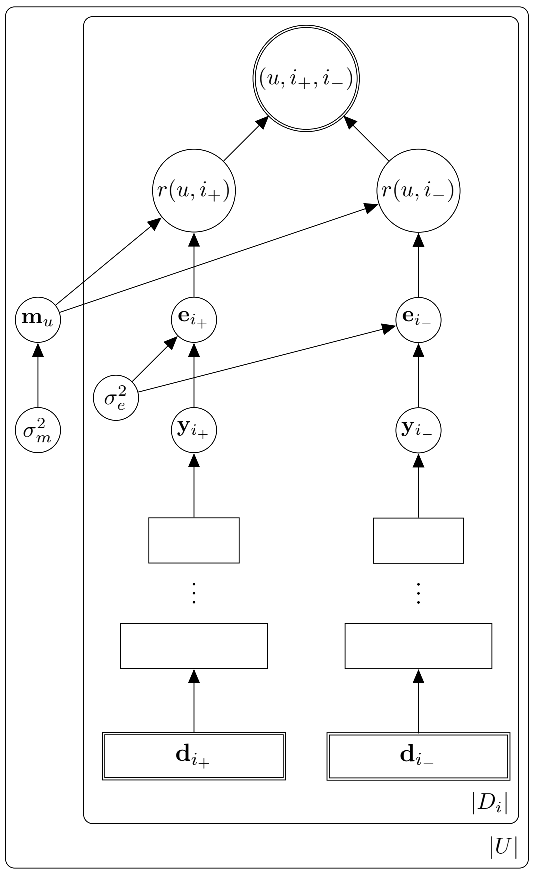


FIGURE 4.1: Graphical model representation of NSPR. The double circled nodes represent observed variables and other nodes are latent variables.

$$\mathbf{a}_{1,i} = \mathbf{W}_1 \mathbf{d}_i$$

$$\mathbf{a}_{l,i} = \psi(\mathbf{W}_l \mathbf{a}_{l-1,i} + \mathbf{b}_l)$$

$$\mathbf{y}_i = \mathbf{W}_L \mathbf{a}_{L-1,i} + \mathbf{b}_L$$

where we use the tanh as the activation function $\psi$ at the hidden layers and the identity function for the output layer.

$$\psi(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{4.1}$$

The output concept vector $\mathbf{y}_i$ is used to calibrate the latent item factor vector $\mathbf{e}_i$ learned from the feedback matrix. On the other hand, the weights and bias in DNN are learned with the guidance of $\mathbf{e}_i$. In other words, $\mathbf{y}_i$ and $\mathbf{e}_i$ are tightly coupled, which allows two-way interactions between the content information and implicit feedback. Specifically, NSPR can be viewed as a probabilistic modeling approach with the generative process described as follows (the graphical model representation of NSPR is shown in Figure 4.1).

1. For each item $i$,

   (a) Map high-dimensional sparse text feature vector $\mathbf{d}_i$ into low-dimensional dense features $\mathbf{y}_i$ via DNN

   (b) Draw a latent item offset vector from normal distribution:

   $$\epsilon_j \sim \mathcal{N}(\mathbf{0}, \sigma_e^2 \mathbf{I}_K) \tag{4.2}$$

   (c) Set the latent item vector to be

   $$\mathbf{e}_i = \mathbf{y}_i + \epsilon_i$$

2. For each user $u$, draw a latent user factor

$$\mathbf{m}_u \sim \mathcal{N}(\mathbf{0}, \sigma_m^2 \mathbf{I}_K)$$

3. For item $i$ given user $u$, calculate the ranking score $r(u, i) = f(\mathbf{m}_u, \mathbf{e}_i)$. For user $u$, item $i^+$ and $i^-$, form the preference triplet $(u, i^+, i^-)$ with the probability $\mathcal{S}\Big(r(u, i^+) - r(u, i^-)\Big)$ where $\mathcal{S}$ is a sigmoid 'S' shape class of functions.

Here $\mathcal{S}\Big(r(u, i^+) - r(u, i^-)\Big)$ defines a pairwise probability that is a monotonically non-decreasing function with respect to the argument $r(u, i^+) - r(u, i^-)$. The intuitive explanation is that if item $i^+$ is preferred over $i^-$ for user $u$, the difference between their ranking scores $r(u, i^+)$ and $r(u, i^-)$ is maximized given the monotonically non-decreasing function $\mathcal{S}(x)$. As a result, item $i^+$ is more preferable than item $i^-$. In Section 4.2.3, we define two variants over the NSPR framework drawing on the Logistic and Probit probability functions.

In this paper, we set the ranking score as $r(u, i) = f(\mathbf{m}_u, \mathbf{e}_i) = \mathbf{m}_u^T \mathbf{e}_i$, which leads to

$$r(u, i^+) - r(u, i^-) = \mathbf{m}_u^T (\mathbf{e}_{i+} - \mathbf{e}_{i-})$$

It is worth noting that the output of the DNN serves as a bridge between the feedback and content information, which is the key that enables NSPR to simultaneously learn an effective feature representation and capture the implicit preference relations between items. The low-dimensional output obtained by DNN is tightly coupled with the latent factors learned from the pairwise probability. The pairwise probability derived from the implicit feedback can guide the learning of feature representations. The learned features can further improve the predictive power of the pairwise ranking model. Thus, the low-dimensional feature representation obtained by DNN captures the latent semantic of item content while being predictive for item ranking, which is very desirable for addressing the item cold-start problem.

### 4.2.2 Parameter Estimation

Based on the NSPR framework above, the posterior likelihood of observing all the preference triplets is:

$$L = \prod_{u} \prod_{i^+,i^-} \mathcal{S}\Big(r(u,i^+) - r(u,i^-)\Big) \prod_{i^+,i^-} \mathcal{N}(\mathbf{e}_i|\mathbf{y}_i, \sigma_e^2 \mathbf{I}) \prod_{u} \mathcal{N}(\mathbf{m}_u|\mathbf{0}, \sigma_m^2 \mathbf{I})$$

By taking the log of the likelihood and simplifying we obtain

$$\begin{aligned} \mathcal{L} &= \sum_{u} \sum_{i^+,i^-} \log \mathcal{S}\Big(r(u,i^+) - r(u,i^-)\Big) \\ &- \frac{1}{2\sigma_e^2} \sum_{i^+,i^-} ||\mathbf{e}_i - \mathbf{y}_i||_2^2 - \frac{1}{2\sigma_m^2} \sum_{u} ||\mathbf{m}_u||_2^2 \end{aligned} \tag{4.3}$$

The parameters to be learned include latent factors $\mathbf{m}_u$ and $\mathbf{e}_i$, and the weights $\mathbf{W}_l$ and bias $\mathbf{b}_l$ in the DNN. The second term in the objective function above is to encode a deep neural network using the latent item vectors $\mathbf{e}_i$ as the target.

We use Stochastic Gradient Descent (SGD) to obtain the Maximum A Posteriori (MAP) estimate. For a given triplet of latent factors $(\mathbf{m}_u, \mathbf{e}_{i^+}, \mathbf{e}_{i^-})$, we compute the stochastic gradients given the current outputs of the DNN (i.e. $\mathbf{y}_i$).

$$\frac{\partial \mathcal{L}}{\partial \mathbf{m}_u} = \frac{1}{\mathcal{S}} \frac{\partial \mathcal{S}}{\partial x} \Big( \mathbf{e}_{i^+} - \mathbf{e}_{i^-} \Big) - \frac{1}{\sigma_m^2} \mathbf{m}_u \tag{4.4}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{e}_{i+}} = \frac{1}{\mathcal{S}} \frac{\partial \mathcal{S}}{\partial x} \mathbf{m}_u - \frac{1}{\sigma_e^2} \Big( \mathbf{e}_{i+} - \mathbf{y}_{i+} \Big) \tag{4.5}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{e}_{i-}} = -\frac{1}{\mathcal{S}} \frac{\partial \mathcal{S}}{\partial x} \mathbf{m}_u - \frac{1}{\sigma_e^2} \Big( \mathbf{e}_{i-} - \mathbf{y}_{i-} \Big) \tag{4.6}$$

where $\frac{\partial \mathcal{S}}{\partial x}$ is the stochastic gradient of the pairwise probability $\mathcal{S}(\cdot)$ with respect to its input ranking score preference. Section 4.2.3 will derive $\frac{\partial \mathcal{S}}{\partial x}$ for various forms of $\mathcal{S}(\cdot)$.

Given the current $\mathbf{e}_{i^+}$ and $\mathbf{e}_{i^-}$, we can then update the weights $\mathbf{W}_l$ and biases $\mathbf{b}_l$ for each layer of the DNN using the Backpropagation algorithm [101]. The stochastic gradients of the likelihood with respect to $\mathbf{W}_l$ and biases $\mathbf{b}_l$ are as follows:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_l} = \delta_{l,i} \mathbf{a}_{l,i}^T \quad \text{and} \quad \frac{\partial \mathcal{L}}{\partial \mathbf{b}_l} = \delta_{l,i}$$

$$\text{where} \quad \delta_{l,i} = \mathbf{W}_{l+1}^T \delta_{l+1,i} \odot (\mathbf{1} - \mathbf{a}_{l,i} \odot \mathbf{a}_{l,i})$$

$$\text{and} \quad \delta_{L,i} = \mathbf{e}_i - \mathbf{y}_i$$

where $\odot$ is the element-wise product. The algorithm iterates over the gradient updates for each preference triplet $(u, i^+, i^-)$ until convergence. Section 4.3.3 discusses the details about the setting of the algorithm.

### 4.2.3 Pairwise Probability

NSPR seamlessly integrates with a multitude of pairwise probability functions for $\mathcal{S}(\cdot)$. In our case, the two pairwise functions we chose can also be interpreted as cumulative distribution functions. We define two variants over the NSPR framework to demonstrate its capabilities.

#### 4.2.3.1 Logistic Probability

One of the most widely used sigmoid functions is the Logistic function, defined as

$$\mathcal{S}(x) = \frac{1}{1 + \exp(-x)} \tag{4.7}$$

It is worth noting in this setting, if $\sigma_e^2$ goes to infinity, the maximization of the objective function Eq.(4.3) is degenerated to the BPR-MF model [99]. The use of non-zero $\sigma_e^2$ in NSPR enables the coupling between the semantic item representation learned by the deep neural network and the latent item factors learned from the pairwise implicit feedback. This tight coupling is missing in the BPR based models.

Computing the stochastic gradient, we obtain the following

$$\frac{\partial \mathcal{S}}{\partial x} = \Big(1 - \mathcal{S}\big(r(u, i^+) - r(u, i^-)\big)\Big)\mathcal{S}\big(r(u, i^+) - r(u, i^-)\big) \tag{4.8}$$

Plugging Eq. (4.8) into Eq.(4.4), (4.5) and (4.6), we obtain the parameter estimation update for the Logistic variant of NSPR, called as NSPR-L.

#### 4.2.3.2 Probit Probability

In statistics, closely related to the Logistic function are the Probit function and Probit model [81]. The Logistic and Probit are both sigmoid functions with a domain between 0 and 1, which makes them both quantile functions - i.e., inverses of the cumulative distribution function (CDF) of a probability distribution. In fact, the Logistic is the quantile function of the Logistic distribution, while the Probit is the quantile function of the Gaussian distribution. We derive the Probit variant of NSPR, denoted as NSPR-P, by setting $\mathcal{S}(x) = \Phi(x)$ as the cumulative distribution function of the Gaussian distribution as follows:

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) dx$$

We can then obtain the stochastic gradient of the objective function as follows:

$$\frac{\partial \mathcal{S}}{\partial x} = \mathcal{N}\big(r(u,i^+) - r(u,i^-)\big)$$

where

$$\mathcal{N} = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

For simplicity we set $\mu = 0$ and $\sigma^2 = 1$ yielding the standard normal Gaussian distribution. Figure 4.2 plots the Logistic, Probit, and Heaviside step functions. As we can see, these functions have a similar ‘S’ shape. The Logistic has a slightly flatter tail while the Probit curve approaches the axes more quickly. In the Probit function, as we increase

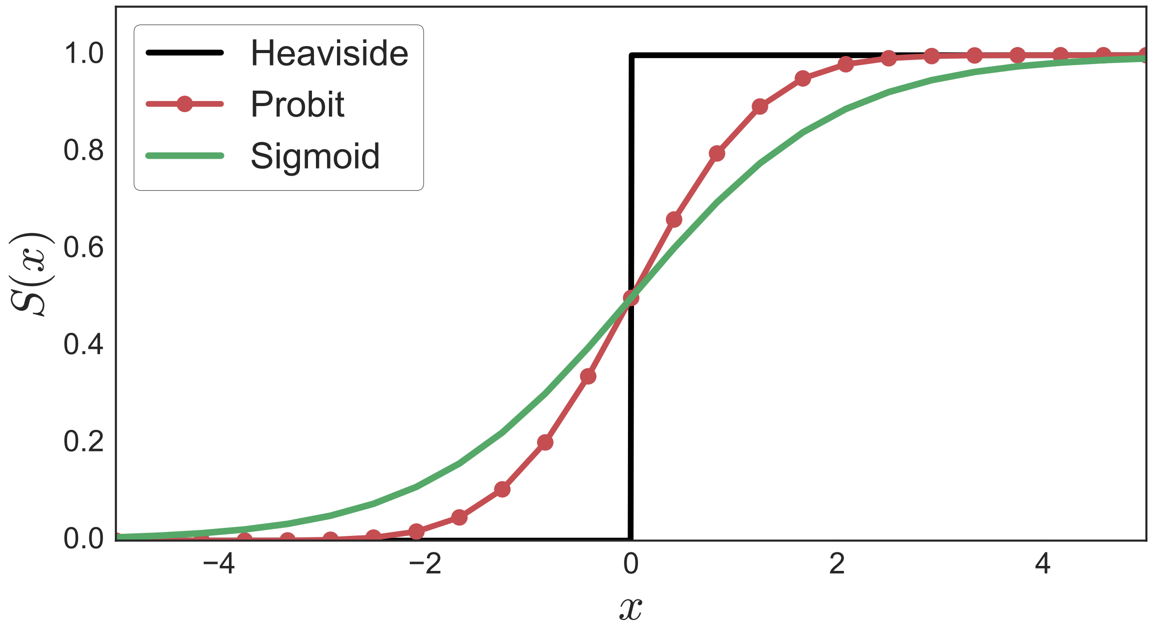


FIGURE 4.2: Logistic and Probit pairwise probability functions in NSPR

the variance the curve will become flatter and elongated. The experiments in Section 5.3 compare the performance of the two variants of NSPR.

### 4.2.4 Prediction for Cold-Start Items

Once the NSPR model is trained, the parameters are used to calculate the ranking score $r(u,i)$ for item $i$ given user $u$. The items are ranked in descending order of $r(u,i)$ for providing personalized recommendation. Similar to Wang and Blei [115], we use the MAP point estimates of parameters to calculate the predicted ranking score

$$r(u,i) \approx (\mathbf{m}_u^*)^T(\mathbf{y}_i + \epsilon_i^*) = (\mathbf{m}_u^*)^T(\mathbf{e}_i^*) \tag{4.9}$$

| | *citeulike-a* | *Yahoo! Movies* |
|---|---|---|
| Users | 5,551 | 7,642 |
| Items | 16,980 | 11,915 |
| Ratings | 204,987 | 211,231 |
| Sparsity | 99.78% | 99.76% |
| Vocabulary Size | 68,911 | 39,664 |
| Avg. Words/Document | 187.97 | 68.26 |
| Avg. Ratings/User | 37.92 | 118.50 |

TABLE 4.2: Dataset statististics

where $\mathbf{m}_u^*$ and $\mathbf{e}_i^*$ are the point estimates by SGD in Section 4.2.2 for the random variables $\mathbf{m}$ and $\mathbf{e}$. $\mathbf{y}$ is deterministic mapped from the content feature vector $\mathbf{d}$.

For the cold-start problem when the item $i$ is unseen in the training data, we set the noise offset $\epsilon_i^*$ in Eq.(4.9) to be zero and obtain the predicted ranking score as follows

$$r(u,i) \approx (\mathbf{m}_u^*)^T(\mathbf{W}_L\mathbf{a}_{L-1,i} + \mathbf{b}_L) \tag{4.10}$$

where $\mathbf{W}_L\mathbf{a}_{L-1,i} + \mathbf{b}_L$ is the output of DNN based on item content input $\mathbf{d}_i$.

## 4.3 Experimental Results

### 4.3.1 Datasets

We evaluate our model on two public datasets from CiteULike[1] and Yahoo! Movies[2]. CiteULike is a web service that allows users to save and share citations to academic

[1]http://www.citeulike.org

[2]R4 - Yahoo! Movies User Ratings and Descriptive Content Information, v.1.0 http://webscope.sandbox.yahoo.com/

papers. The first dataset *citeulike-a*[3] [115] contains 5,551 users, 16,980 items with 204,987 positive entries. Implicit feedback is encoded as positive if the user has the item in their personal library and encoded as negative otherwise. The second dataset, *Yahoo! Movies* consists of users rating movies on a scale of 1-5 with a short synopsis. To be consistent with the implicit feedback setting, we extract only positive ratings (rating 5) for training and testing. After removing movies without a synopsis, this yields 7,642 users, 11,915 items, and 221,367 positive ratings. The characteristics of the dataset are summarized in Table 4.2. It is worth noting that *citeulike-a* is sparser in ratings and has over twice the number of average words per a document while the contrary is true for *Yahoo! Movies.* Similar to [115, 116], we preprocess the data by removing the users with fewer than 3 positive entries, concatenating the title and abstract (movie synopsis), removing stopwords, stemming and construct our vocabulary from the top $N$ terms based on TF-IDF then use raw term counts. We randomly hold out 20% of the items for testing and the remaining 80% of the items are used for training. The split of data yields the cold-start setting since the items in each set are disjoint from the other sets, and are new items for the users. We set the vocabulary size ($N$) to 8,000 and 20,000 for the *citeulike-a* and *Yahoo! Movies* datasets, respectively.

## 4.3.2 Evaluation Metrics

The accuracy of a recommendation model is measured by using three commonly used metrics, namely Mean average precision (MAP), Normalized discounted cumulative gain

[3]http://www.cs.cmu.edu/~chongw/data/citeulike

(NDCG), and Recall (R) [79]. MAP is widely adopted for evaluation of item recommendation. Because users are usually interested in a few top-ranked items, NDCG@$N$ is used to compare the top-$n$ recommendation performance. We also use Recall because the feedback information is implicit. Precision oriented metrics such as MAP and NDCG may not be sufficient since a negative entry could be caused by the fact that the user is not interested in the item, or that the user is not aware of its existence.

### 4.3.3 Baselines and Settings

We use the following baselines for comparison in the experiments. They are the state-of-the-art recommendation algorithms for recommendation tasks and consider content information a requirement for an algorithm to address the item cold-start problem.

- SVDFeature [12], which performs feature-based matrix factorization allowing for additional content and relationships.
- Collective matrix factorization (CMF) [105], which simultaneously factors multiple matrices to learn integrating relations between them.
- Collaborative topic regression (CTR) [115], which combines probabilistic topic modeling with a latent factor model.
- Collaborative deep learning (CDL) [116], which creates a deep feature representation using stacked denoising auto-encoders with CTR.

- Neural Semantic Personalized Ranking (NSPR) with two variants: Logistic (NSPR-L) and Probit (NSPR-P) which we proposed in Section 4.2.

We select all hyperparameters by cross-validation grid search, holding out 10% of the training data to create a separate validation set. We then tune hyperparameters according to Recall@300 achieved on the validation set. In our experimental results, we utilize both the training and validation sets as training data. For SVDFeature, we use the ranking setting and found good results when $\lambda_u$ and $\lambda_v$ are set to 0.04. In CMF, we set both matrices (rating and item content) to the sparse setting and 0.1 and 0.05 for the ratings and item content matrices respectively. CTR performed best when we set $a = 1$, $b = 0.01$, $\lambda_u = 0.1$, and $\lambda_v$=10. CDL performed best with the architecture "200-200-$K$-200-200" with $\lambda_v = 10$, $\lambda_u = 1$ and $\lambda_n = 100$.

For the SGD algorithm of our NSPR models, we use the adaptive subgradient method (AdaGrad) [23] to schedule the learning rate with the initial value of 0.1. The regularization parameter $\sigma_m^2$ of latent user factors are set to be 9. We randomize the preference triplets $(u, i^+, i^-)$ for SGD training by uniformly randomly sampling a user $u$ from $\mathcal{U}$, a positive item from $I_u^+$, and a negative item from $\mathcal{V} \setminus \mathcal{I}_u^+$, respectively. This sampling strategy reduces the chance of updating the same user-item combination in consecutive iterations, which otherwise may lead to poor convergence [99]. The initial values of the parameters in the SGD algorithm are uniformly randomly sampled from $[0, 1]$ and the stopping criteria is when the relative change of the likelihood function is less than 0.01%. The default number of nodes for each hidden layer is 256. We set the default

| | SVDFeature | CMF | CTR | CDL | NSPR-P | NSPR-L |
|---|---|---|---|---|---|---|
| R@10 | 0.0039 | 0.0023 | 0.0692 | 0.0919 | **0.1294** | 0.1290 |
| R@25 | 0.0095 | 0.0055 | 0.1516 | 0.1693 | **0.2324** | 0.2308 |
| R@50 | 0.0188 | 0.0110 | 0.2518 | 0.2580 | **0.3402** | 0.3378 |
| R@100 | 0.0335 | 0.0562 | 0.3802 | 0.3634 | **0.4716** | 0.4646 |
| R@150 | 0.0493 | 0.0919 | 0.4616 | 0.4304 | **0.5526** | 0.5443 |
| R@200 | 0.0666 | 0.1066 | 0.5197 | 0.4807 | **0.6100** | 0.6042 |
| R@250 | 0.0825 | 0.1198 | 0.5647 | 0.5203 | **0.6547** | 0.6515 |
| R@300 | 0.0985 | 0.1459 | 0.6044 | 0.5514 | **0.6862** | 0.6859 |
| MAP@500 | 0.0025 | 0.0026 | 0.0522 | 0.0672 | **0.0923** | 0.0906 |
| NDCG@5 | 0.0027 | 0.0024 | 0.0457 | 0.0773 | **0.1296** | 0.1259 |
| NDCG@10 | 0.0039 | 0.0026 | 0.0578 | 0.0809 | **0.1432** | 0.1418 |

Table 4.3: Experimental results for different methods on the *citeulike-a* dataset. The best results in each metric are highlighted.

parameters for both variants on the *citeulike-a* dataset with 128 latent factors, $\sigma_e^2$ to 500 and dropout to 0.1 with two hidden layers. In the *Yahoo! Movies* dataset, both variants use two hidden layers with $K = 16$. We set $\sigma_e^2$ to 9 and 200 for NSPR-L and NSPR-P respectively. We use the default parameter values in the experiments unless otherwise specified.

### 4.3.4 Baseline Comparison

Table 4.3 contains the results of NSPR compared to the baseline models measuring Recall@$M$, MAP@500, NDCG@5, and NDCG@10 on the *citeulike-a* dataset. We can see that both NSPR models perform equally well and outperform all baselines across all metrics. The nearest competitor is CTR for Recall@300. Concerning MAP@500 and NDCG, the three models using deep learning (NSPR-P, NSPR-L and CDL) obtain superior performance over models that do not. We can speculate deep learning methods utilize learned latent semantics from item content to prioritize more relevant items. CMF

| | SVDFeature | CMF | CTR | CDL | NSPR-P | NSPR-L |
|---|---|---|---|---|---|---|
| R@10 | 0.0042 | 0.0013 | 0.0051 | 0.0234 | 0.0200 | **0.0453** |
| R@25 | 0.0109 | 0.0040 | 0.0112 | 0.0414 | 0.1193 | **0.1361** |
| R@50 | 0.0209 | 0.0090 | 0.0200 | 0.0653 | 0.0619 | **0.0840** |
| R@100 | 0.0427 | 0.0324 | 0.0336 | 0.1071 | 0.2054 | **0.2127** |
| R@150 | 0.0625 | 0.0769 | 0.0495 | 0.1439 | 0.2764 | **0.3010** |
| R@200 | 0.0837 | 0.1161 | 0.0639 | 0.1816 | 0.3377 | **0.3559** |
| R@250 | 0.1046 | 0.1395 | 0.0778 | 0.2181 | 0.4436 | **0.4437** |
| R@300 | 0.1259 | 0.1551 | 0.0903 | 0.2518 | 0.5179 | **0.5266** |
| MAP@500 | 0.0034 | 0.0022 | 0.0042 | 0.0168 | 0.0173 | **0.0221** |
| NDCG@5 | 0.0028 | 0.0015 | 0.0044 | 0.0172 | 0.0094 | **0.0217** |
| NDCG@10 | 0.0035 | 0.0018 | 0.0046 | 0.0175 | 0.0186 | **0.0380** |

Table 4.4: Experimental results for different methods on the *Yahoo! Movies* dataset. The best results in each metric are highlighted.

outperforms SVDFeature when the metric is at a higher level, i.e. when Recall is at 100 or greater but SVDFeature reports better NDCG while both have similar performance on MAP@500. SVDFeature may place a higher priority on relevant recommendations by drawing upon stronger user-based features. Both models use relatively simple linear transformations on the item content deteriorating performance to generalize to new items. These results indicate the benefits of using deep learning to construct robust feature representations of item content for the cold-start problem.

In the *Yahoo! Movies* dataset, NSPR models outperform or demonstrate competitive performance against each baseline for all metrics shown in Table 4.4. Again, NSPR-L performs best with NSPR-P performing very closely. As noted earlier, the dataset is characterized by denser ratings and sparser item content may lead to a more complex relation. Subsequently, topic models may lack the ability to capture this intricate relationship with sparser documents leading to CTR's poor performance. SVDFeature and CMF both obtain better Recall@300 at 0.1259 and 0.1551, respectively. The fact that

CDL outperforms other baselines additionally with NSPR's performance demonstrate the advantages of deep learning models which aim to capture complex and subtle relations between item content and latent features. The NSPR framework's flexibility to integrate different types of pairwise probability functions demonstrates its adaptability. Furthermore, the difference in NSPR variations performance is the pairwise function. The Probit function's hyperparameters $\mu$ and $\sigma^2$ could be further optimized to suit different dataset characteristics which we leave to future work. These results prove the effectiveness of NSPR using a pairwise probability for implicit feedback and utilizing DNN for learning latent semantics from item content, compared to the pointwise loss and auto-encoder in CDL. As we can see, the NSPR models demonstrate competitive or superior performance over the state-of-the-art baselines across all metrics.

### 4.3.5 Number of Latent Factors

Selecting the optimal number of latent factors and hidden layers can have a devastating effect on performance as we demonstrate in this section. Varying these hyperparameters may introduce noise causing difficulty in isolating the actual effect. To account for the variance, we perform 10-Fold cross-validation by splitting the items into ten equal parts. We use nine folds as training data and the final fold as testing such that we yield a cold-start setting as described earlier in Section 4.3.3. We repeat this process ten times each with a different test fold and report the average Recall@300 and NDCG@10.

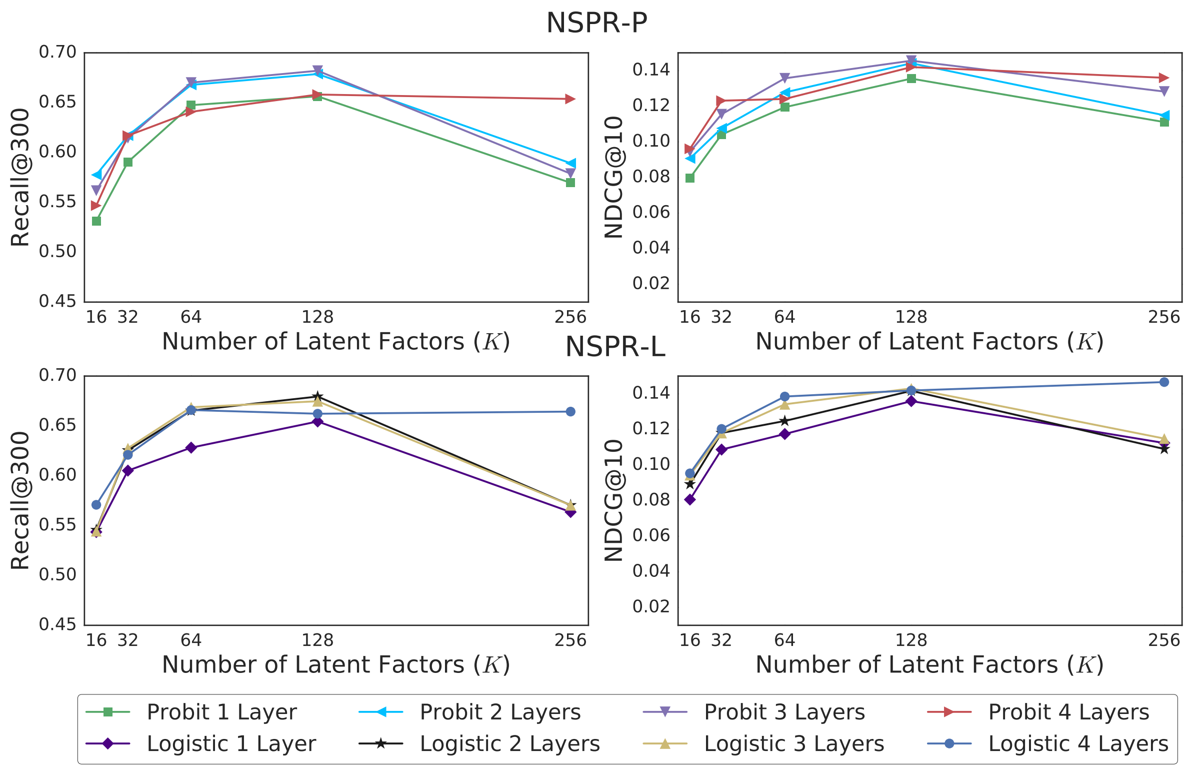


FIGURE 4.3: Recall@300 (left) and NDCG@10 (right) for varying number of latent factors ($K$) and hidden layers ($L$) averaged over 10-folds on the *citeulike-a* dataset.

Figure 4.3 illustrates the effect of varying the number of latent factors ($K$=16, 32, 64, 128, and 256) and hidden layers ($L$=1,2,3,4) for both NSPR variants reporting the mean Recall@300 and NDCG@10 for the *citeulike-a* dataset. In both variants, as the number of latent factors increases a corresponding climb in performance is seen on both metrics despite the number of hidden layers. Each particular configuration obtains peak performance on both metrics at 128 latent factors with the exception of NSPR-L where the curve continues to increase with 256 latent factors and four hidden layers. With respect to the number of hidden layers in the DNN, a single hidden layer struggles to capture the intricate non-linear semantics. The optimal Recall and NDCG occurs at two and three hidden layers where sufficient modeling capacity exists. NSPR-L

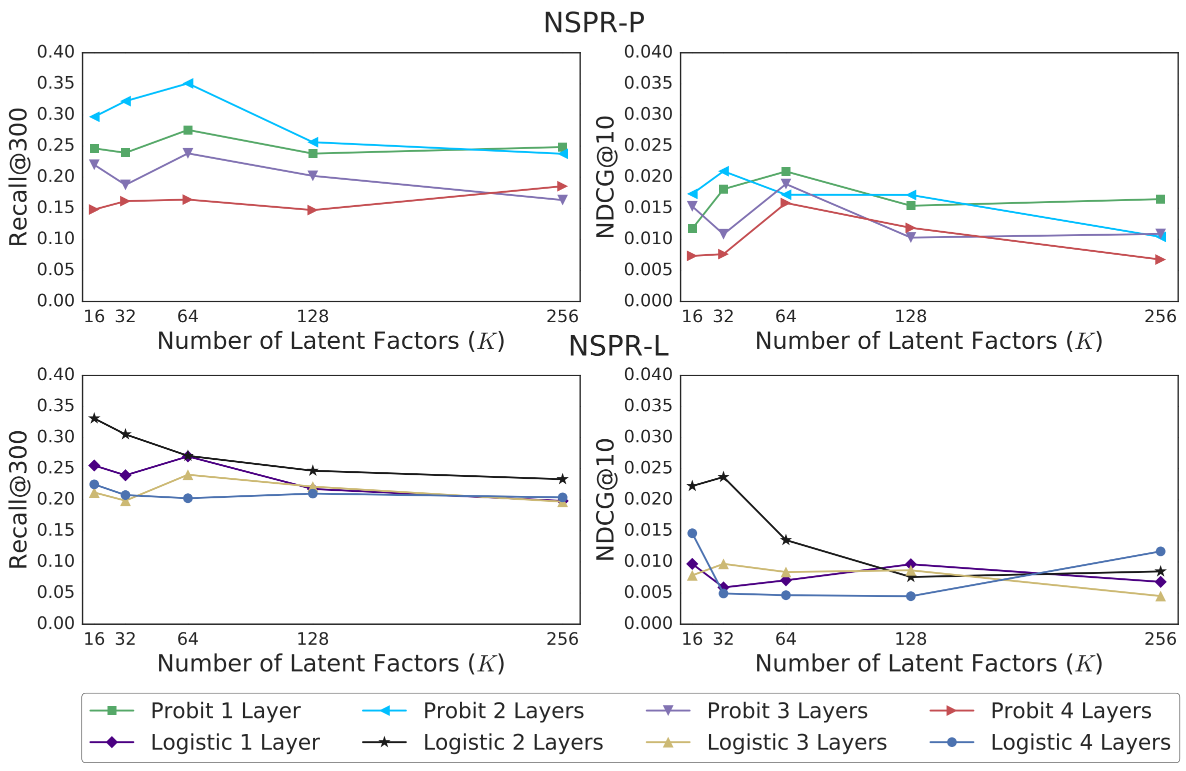


FIGURE 4.4: Recall@300 (left) and NDCG@10 (right) for varying number of latent factors ($K$) and hidden layers ($L$) averaged over 10-folds on the *Yahoo! Movies* dataset.

(bottom) with two hidden layers obtains the best performance for Recall@300 with 128 latent factors. NSPR-P (top) simultaneously performs the best on Recall and NDCG with three hidden layers and 128 latent factors. Multiple parameter configurations demonstrate competitive performance across both metrics. We report the variance over the ten folds in Table 4.5 where we find the variance is relatively small. Overall, lower fluctuations were reported with latent factors in the range of [32, 64] and two hidden layers in both variants. Conversely, the highest volatility is achieved with three hidden layers and 256 latent factors

In Figure 4.4, both NSPR variants show the best performance with two hidden layers and latent factors in the range of [32, 64] on the *Yahoo! Movies* dataset. In general,

| | Recall@300 | | | | | NDCG@10 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 16 | 32 | 64 | 128 | 256 | 16 | 32 | 64 | 128 | 256 |
| P/1 | 0.0020 | 0.0022 | 0.0003 | 0.0012 | 0.0152 | 0.0001 | 0.0002 | 0.0001 | 0.0001 | 0.0011 |
| P/2 | 0.0009 | 0.0010 | 0.0002 | 0.0003 | 0.0072 | 0.0001 | 0.0002 | 0.0002 | 0.0000 | 0.0008 |
| P/3 | 0.0026 | 0.0017 | 0.0003 | 0.0003 | 0.0242 | 0.0001 | 0.0002 | 0.0003 | 0.0001 | 0.0012 |
| P/4 | 0.0046 | 0.0020 | 0.0068 | 0.0025 | 0.0020 | 0.0002 | 0.0002 | 0.0008 | 0.0005 | 0.0004 |
| L/1 | 0.0021 | 0.0007 | 0.0032 | 0.0011 | 0.0040 | 0.0001 | 0.0000 | 0.0002 | 0.0000 | 0.0001 |
| L/2 | 0.0040 | 0.0003 | 0.0003 | 0.0001 | 0.0039 | 0.0002 | 0.0001 | 0.0002 | 0.0001 | 0.0006 |
| L/3 | 0.0096 | 0.0007 | 0.0003 | 0.0003 | 0.0104 | 0.0003 | 0.0002 | 0.0002 | 0.0002 | 0.0010 |
| L/4 | 0.0030 | 0.0011 | 0.0003 | 0.0008 | 0.0002 | 0.0004 | 0.0002 | 0.0002 | 0.0003 | 0.0001 |

Table 4.5: Variance over 10-Folds of NSPR variants for varying the number of latent factors ($K$) and hidden layers ($L$) on the *citeulike-a* dataset. We denote P/1 to indicate NSPR-Probit with one hidden layer and similarly, L/2 to indicate NSPR-Logistic with two hidden layers.

| | Recall@300 | | | | | NDCG@10 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 16 | 32 | 64 | 128 | 256 | 16 | 32 | 64 | 128 | 256 |
| P/1 | 0.0205 | 0.0184 | 0.0116 | 0.0140 | 0.0134 | 0.0001 | 0.0002 | 0.0001 | 0.0001 | 0.0002 |
| P/2 | 0.0146 | 0.0060 | 0.0053 | 0.0137 | 0.0128 | 0.0001 | 0.0000 | 0.0000 | 0.0005 | 0.0001 |
| P/3 | 0.0091 | 0.0136 | 0.0171 | 0.0096 | 0.0107 | 0.0003 | 0.0001 | 0.0005 | 0.0001 | 0.0003 |
| P/4 | 0.0166 | 0.0089 | 0.0058 | 0.0067 | 0.0119 | 0.0000 | 0.0000 | 0.0004 | 0.0004 | 0.0001 |
| L/1 | 0.0124 | 0.0190 | 0.0185 | 0.0135 | 0.0246 | 0.0000 | 0.0000 | 0.0000 | 0.0002 | 0.0000 |
| L/2 | 0.0196 | 0.0071 | 0.0138 | 0.0137 | 0.0166 | 0.0004 | 0.0016 | 0.0003 | 0.0001 | 0.0001 |
| L/3 | 0.0121 | 0.0122 | 0.0167 | 0.0137 | 0.0140 | 0.0001 | 0.0002 | 0.0001 | 0.0002 | 0.0000 |
| L/4 | 0.0123 | 0.0138 | 0.0118 | 0.0134 | 0.0143 | 0.0007 | 0.0000 | 0.0000 | 0.0000 | 0.0005 |

Table 4.6: Variance over 10-Folds of NSPR variants for varying the number of latent factors ($K$) and hidden layers ($L$) on the *Yahoo! Movies* dataset. We denote P/1 to indicate NSPR-Probit with one hidden layer and similarly, L/2 to indicate NSPR-Logistic with two hidden layers.

NSPR with a single hidden layer lacks the capability to model the complex relations present. As three or more hidden layers are used a diminish in performance may suggest overfitting. NSPR-P with one and two hidden layers show similar performance and a balance between Recall and NDCG. Illustrating an equilibrium between the pairwise, latent factors and DNN architecture is achievable. In some cases, high Recall does not directly translate to the NDCG metric. Particularly we observe this behavior in NSPR-L with four hidden layers and 256 latent factors, where Recall is among the lowest obtained yet NDCG is among the highest. Demonstrating NSPR is flexible, and its architecture can be fine tuned for specific metrics.

Table 4.6 summarizes the variance of NSPR over each of the ten folds for Recall and

NDCG@10. Similar to the *citeulike-a* dataset, the variance is generally low across different configurations. Since the diversity on NDCG is too small, we limit our discussion to the Recall metric. NSPR demonstrates the highest variance with a single hidden consisting of 16 and 256 latent factors for NSPR-P and NSPR-L respectively. Cumulatively, NSPR-L has more variance. However, the variance reduces as the number of layers increases. A similar yet more subtle trend is present in the Probit version. We could speculate the optimization of the non-convex nature causes the DNN to become stuck at a saddle point or bad local minium producing the variance. Nevertheless, the small size of the dataset could easily lead to overfitting a large number of parameters. In this case, initializing the DNN weights with pretrained word embeddings may improve performance. In short, the *Yahoo! Movies* dataset contains denser ratings with sparse item content leading to a more complex relation where deeper architectures can capture these nonlinearlities.

### 4.3.6 Architecture of NSPR

In this section, we more closely examine the architecture of NSPR by varying the number of hidden layers from $L = 1, 2, 3, 4$ using the default parameters. Table 4.7 reports the values of the *citeulike-a* dataset. We can see the performance is relatively stable across different number of hidden layers for NSPR-P. For NSPR-L, we can see a single hidden layer does not provide enough modeling capacity as performance increases with additional layers. Both models obtain the best results with two hidden layers and increasing the number of layers may result in overfitting. In the *Yahoo! Movies* dataset,

| | *citeulike-a* | | | |
|---|---|---|---|---|
| Hidden Layers ($L$) | 1 | 2 | 3 | 4 |
| NSPR-P | 0.6730 | 0.6862 | 0.6663 | 0.6674 |
| NSPR-L | 0.5696 | 0.6859 | 0.6638 | 0.6381 |

Table 4.7: Recall@300 for NSPR with different number of hidden layers ($L = 1, 2, 3, 4$) for *citeulike-a*.

| | *Yahoo! Movies* | | | |
|---|---|---|---|---|
| Hidden Layers ($L$) | 1 | 2 | 3 | 4 |
| NSPR-P | 0.4583 | 0.5179 | 0.3921 | 0.1414 |
| NSPR-L | 0.4548 | 0.5266 | 0.4393 | 0.0941 |

Table 4.8: Recall@300 for NSPR with different number of hidden layers ($L = 1, 2, 3, 4$) for *Yahoo! Movies*.

we see performance also peaking around two hidden layers and then sharply decreasing at four hidden layers in Table 4.8. We believe this to be an overfitting issue from the DNN, also observed by Salakhutdinov et al. [102]. Dropout lead to a decrease in performance and subsequently decreased the stability of the method across the number of hidden layers in contrast to the previous results on the *citeulike-a* dataset. In general, a single hidden layer architecture lacks the modeling capacity to capture these nonlinearlities where the two hidden layer architecture excelled. Additional layers lead to overfitting possibly due to the small size of the dataset. In future work, we plan to apply deeper architectures with large-scale test beds and exploit external knowledge such as pretrained word embeddings.

In the core architecture of NSPR, the DNN is approximating the item latent space and not a directly observable variable. One could view the item latent factor as an additional hidden layer connecting to the DNN. The initial error propagates to the latent item vector then we evaluate another error function with respect to the DNN output. We experimented with different combinations of activation functions ranging from the tanh, Logistic, Rectified Linear Unit (ReLU) [85] and identity. We found the best performance with the tanh function and the identity as the output. The Logistic function provided slightly deteriorated performance. One explanation may be the Logistic function is bound from $[0, 1]$ slowing learning by outputting a positive mean as inputs to subsequent hidden layers whereas the symmetry of tanh generally provides a zero centered mean typically leading to better convergence [69]. In our particular problem, we did not find ReLU's to enhance performance as demonstrated in Cheng et al. [15].

### 4.3.7 Impact of Item Variance

The item variance $\sigma_e^2$ in Eq.(4.2) models the interaction between the semantic learning of DNN from item content and latent factor learning from implicit feedback. In this section, we investigate the impact of $\sigma_e^2$ on the NSPR models and vary it from the default values specified in Section 4.3.3. We vary $\sigma_e^2$ by $\pm 5$ of our default values followed by more extrenous values. Table 4.9 demonstrates the effect of $\sigma_e^2$ on the *citeulike-a* dataset. As we can see, NSPR's performance is relatively robust over a broad range of values and generally, shows relatively subtle changes with the exception when $\sigma_e^2$ is small i.e. 0.1. In contrast, the *Yahoo! Movies* dataset shows more sensitivity to $\sigma_e^2$ in performance as shown in Table 4.10. When $\sigma_e^2$ is small, the DNN strongly influencing the item latent factor overfitting to item content. As the value of $\sigma_e^2$ increases, the DNN item content integration starts to diverge from the item latent factor and as $\sigma_e^2$ goes to infinity the

| $\sigma_e^2$ | 0.1 | 495 | 500 | 505 | 1000 |
|---|---|---|---|---|---|
| NSPR-P | 0.0875 | 0.6794 | 0.6862 | 0.6812 | 0.6626 |
| NSPR-L | 0.0802 | 0.6763 | 0.6859 | 0.6474 | 0.6410 |

TABLE 4.9: Recall@300 for different values of $\sigma_e^2$ on the *citeulike-a* dataset

| $\sigma_e^2$ | 0.1 | 195 | 200 | 205 | 250 |
|---|---|---|---|---|---|
| NSPR-P | 0.2619 | 0.4211 | 0.5179 | 0.4475 | 0.4696 |
| $\sigma_e^2$ | 0.1 | 4 | 9 | 13 | 100 |
| NSPR-L | 0.2889 | 0.4125 | 0.5266 | 0.4833 | 0.4873 |

TABLE 4.10: Recall@300 for different values of $\sigma_e^2$ on the *Yahoo! Movies* dataset

model degenerates to the BPR criterion. These results demonstrate the importance of keeping a balance between pairwise probability and latent semantic learning of DNN.

### 4.3.8 Qualitative Evaluation

To further investigate the effectiveness of NSPR, we compare the interpretability of the top 5 recommended items against baselines for a given user. Table 4.11 lists the recommended articles for *citeulike-a* by NSPR-L, CDL, and CTR. We might hypothesize that this user is interested in library and information sciences. CTR correctly recommends only two articles while four out of the five article titles recommended contain the root word 'science.' The remaining article is 'In a paperless world a new role for academic libraries: providing open access' which we can also expect the terms 'academic' and 'library' to co-occur with 'science' leading CTR astray. Similarly, CDL identified words 'technology' and 'publication' while correctly recommending one item. CDL incorrectly recommends the article 'The Molecular Biology Database Collection: 2005 update'. Upon inspecting the training data, the user does not have any interests in biology. CDL

| |
|---|
| NSPR-L |
| **1. Ten-Year Cross-Disciplinary Comparison of the Growth of Open Access and How it Increases Research Citation Impact** |
| **2. Peer Review in the Google Age: Is technology changing the way science is done and evaluated?** |
| 3. Defrosting the digital library: bibliographic tools for the next generation web. |
| 4. What are digital libraries? Competing visions |
| **5. A New Era in Citation and Bibliometric Analyses: Web of Science, Scopus, and Google Scholar** |
| CTR |
| 1. Do pressures to publish increase scientists' bias? An empirical support from US states data |
| **2. Unavailability of online supplementary scientific information from articles published in major journals** |
| 3. Strategic reading, ontologies, and the future of scientific publishing. |
| **4. In a paperless world a new role for academic libraries: providing open access** |
| 5. Universality of citation distributions: Toward an objective measure of scientific impact |
| CDL |
| 1. Where do educational technologists really publish? An examination of successful emerging scholars' publication outlets |
| 2. The Molecular Biology Database Collection: 2005 update |
| 3. Strategic reading, ontologies, and the future of scientific publishing. |
| **4. Peer Review in the Google Age: Is technology changing the way science is done and evaluated?** |
| 5. Déjà vu–a study of duplicate citations in Medline. |

Table 4.11: Top-5 recommended articles by NSPR-L, CDL and CTR. The positive items are highlighted.

may have identified 'database' as a term co-occurring with 'digital', 'libraries' and 'publications.' NSPR-L captures more semantics related to the users primary interests such as digital library and citation metrics.

The top 5 recommended movies from the *Yahoo! Movies* dataset is listed in Table 4.12. Analyzing the genres of each movie recommended we may speculate the user has diverse tastes in movies from a variety of genres spanning comedy, action, and adventure. CTR and CDL do not identify the action and adventure genre which comprises a significant

| NSPR-L |
|---|
| **1. American Wedding (2003)** |
| **2. Tarzan and the Lost City (1998)** |
| **3. Indiana Jones and the Last Crusade (1989)** |
| 4. Bloody Murder (1999) |
| 5. Bloody Murder 2 (2003) |
| CDL |
| 1. Virus (1980) |
| 2. Take This Job and Shove It (1981) |
| 3. Going Greek (2001) |
| 4. Shine (1997) |
| 5. Bless the Beasts and Children (1972) |
| CTR |
| 1. Ricochet River (2001) |
| 2. Buffalo Soldiers (1988) |
| 3. Tempest (1982) |
| 4. Two Hands (1999) |
| **5. JFK (1991)** |

Table 4.12: Top-5 recommended movies by NSPR-L, CDL and CTR. The positive items are highlighted.

portion of the user's preferences while NSPR-L discovered the association, particularly in recommending 'Indiana Jones and the Last Crusade.' It may seem odd that NSPR-L recommended the horror films 'Bloody Murder', however, inspecting the users library revealed additional horror movies such as 'Texas Chainsaw Massacre.'

## 4.4 Summary

Item cold-start and implicit user feedback present two of the greatest challenges to the real-world recommender systems. In this paper, we tackle the challenges by proposing a novel probabilistic generative modeling approach to integrate deep neural network (DNN) with three pairwise ranking variants. With the modeling power of deep learning, we can extract semantic representation of items and couple it with the latent factors learned from implicit feedback. The experiments show that the proposed approach significantly outperforms the competitive baselines on two real-world public datasets.

This work is just an initial step towards a promising new direction. In future work, we plan to incorporate other types of deep learning architectures such as Convolutional neural network (CNN) [67], Deep belief network (DBN) [44], and Recurrent neural network (RNN) [5]. Further performance boost may be possible when using such deep learning models since these models can explicitly take the context and ordering of words into account. Moreover, we plan to explore deeper architectures by applying data normalization techniques [50, 1] to help model stability and a better local optimum. Last but not the least, the proposed NSPR framework can be readily extended to handle the listwise preferences if ranked lists of items are given as ground truth for recommendations. The listwise learning to rank likelihood functions such as ListMLE and ListNet [76] can be directly plugged into the proposed generative framework. The listwise approach may be able to handle more complex user feedback than pairwise preferences.

# Chapter 5

# Attentive Contextual Denoising Autoencoder

## 5.1 Introduction

Personalized recommendation has become increasingly pervasive nowadays. Users receive recommendations on products, movies, point-of-interests and other online services. Traditional collaborative filtering techniques have demonstrated effectiveness in a wide range of recommendation tasks, but they are unable to capture complex relationships between users and items. There is a surge of interest in applying deep learning to recommender systems due to its nonlinear modeling capacity and recent success in other domains such as computer vision and speech recognition. However, prior work does not incorporate contexual information, which is usually largely available in many recommendation tasks. In this paper, we propose a deep learning based model for contexual recommendation. Specifically, the model consists of a denoising autoencoder neural network architecture augmented with a context-driven attention mechanism, referred to as *Attentive Contextual Denoising Autoencoder (ACDA)*. The attention mechanism is utilized to encode the contextual attributes into the hidden representation of the user's preference, which associates personalized context with each user's preference to provide recommendation targeted to that specific user. Experiments conducted on multiple real-world datasets from *Meetup* and *Movielens* on event and movie recommendations demonstrate the effectiveness of the proposed model over the state-of-the-art recommenders.

The information overload caused by the deluge of data has resulted in the need for recommender systems. The goal of a recommender system is to predict the unknown preferences of a user based on the known preferences of that user on certain items. Classic methods for recommender systems, such as content-based and collaborative filtering, have been effective in the past and they have offered a reasonable level of performance. However, these methods lack the ability to model complex nonlinear relationships that usually accompany the user-item interaction. With the recent success of deep learning in computer vision and speech recognition [34], there has been a surge of interest in applying deep learning methods to recommendation tasks [128]. The existing work in this domain is still quite limited, and furthermore, it does not utilize contextual information that is largely present in the real-world scenarios. Context provides additional information to the user-item interaction, which in turn improves the quality of the recommendation [22]. The attention mechanism [2] provides an intuitive way to incorporate context into the user-item interaction. Motivated by these factors, we propose a novel model for personalized recommendation based on the denoising autoencoder augmented with a context-driven attention mechanism. We call this model the *Attentive Contextual Denoising Autoencoder (ACDA).*

Autoencoders [34] are feed-forward neural networks capable of learning a representation of the input data, also known as codings. The codings typically learnt by an autoencoder are of much lower dimensionality than the original input. Denoising autoencoders [34] are a variant of the basic autoencoder that add noise to the input and train the network to recover the original input at the output layer. This forces the network to discover robust features in the data representation, and prevents the model from learning the trivial identity function. The autoencoder architecture makes it suitable for use in recommender systems as the hidden layer captures the latent representation of the data, allowing the model to learn the latent factors associated with the user-item interaction. It has been shown [121] that the denoising autoencoder architecture is a nonlinear generalization of latent factor models [64, 83], which have been widely used in recommender systems. Therefore, we utilize denoising autoencoder as the main building block for the proposed *ACDA* model.

Context provides an added dimension to real-world applications. Recommender systems for movies, products, point-of-interests and services utilize context to provide a meaningful personalized recommendation [22]. For example, genre such as horror, drama, thriller, comedy etc., is an important context for movie recommendation as people generally like the same type of movies. Location and time-of-day are useful context to consider while recommending point-of-interests. There is existing work in the literature that provides contextual recommendations [129, 94, 57]. The *ACDA* model incorporates contextual information via the attention mechanism for personalized recommendation. We apply the *ACDA* model to two real-world problems of event recommendation [52] and movie recommendation. For the event recommendation task, we use the user *group* and event *venue* as the contextual attributes, whereas the movie *genre* is used as the contextual attribute for the movie recommendation task.

The attention mechanism has been instrumental in dealing with structured problems, such as machine translation and caption generation [113, 2, 41]. The objective of the mechanism is to highlight, or focus attention on, a certain subset of the data. The attention mechanism accepts a certain input and a context that accompanies the input. The output of the attention mechanism is considered as a summary of the input focusing on the information linked to the provided context. The attention mechanism is generally applied for two reasons–first, to provide for efficient processing of a high-dimensional input by processing only subsets of the input, and second, to focus on specific parts of the input in terms of its relevance. A classical example of the use of the attention mechanism is image captioning, where the mechanism focuses on certain subsets of the image to generate the suitable caption. The *ACDA* model utilizes the attention mechanism to apply the contextual attributes to the hidden representation of the user's preference. This helps the model to associate personalized context with each user's preference to provide recommendation targeted to that specific user.

The *ADCA* model accepts the user's preference on existing items as input, which includes both positive and negative instances. The input is partially corrupted to learn a robust representation of the data. The input is mapped to an internal representation of lower dimensionality by the hidden layer, where the contextual parameters are applied via the attention mechanism to focus on the user-specific relevant context. The output of the model is the reconstructed user input, which is the predicted preference of the user. The model is trained to minimize the loss between the original corrupted input and the reconstructed input generated at the output layer. We use multiple real-world datasets to conduct comprehensive experiments for the proposed *ACDA* model. The datasets for the event recommendation task are obtained from *Meetup*[1], a popular Event-Based Social Network (EBSN). We use the publicly available *Movielens* 100K dataset for the movie recommendation task. The experimental results show that the proposed model performs better than current state-of-the-art baselines. The main contributions of this paper can be summarized as follows.

- We propose a novel *Attentive Contextual Denoising Autoencoder (ACDA)* model for recommendation. To the best of our knowledge, this is the first study that attempts to utilize the contextual information via attention mechanism in the deep architectures.

[1]http://www.meetup.com

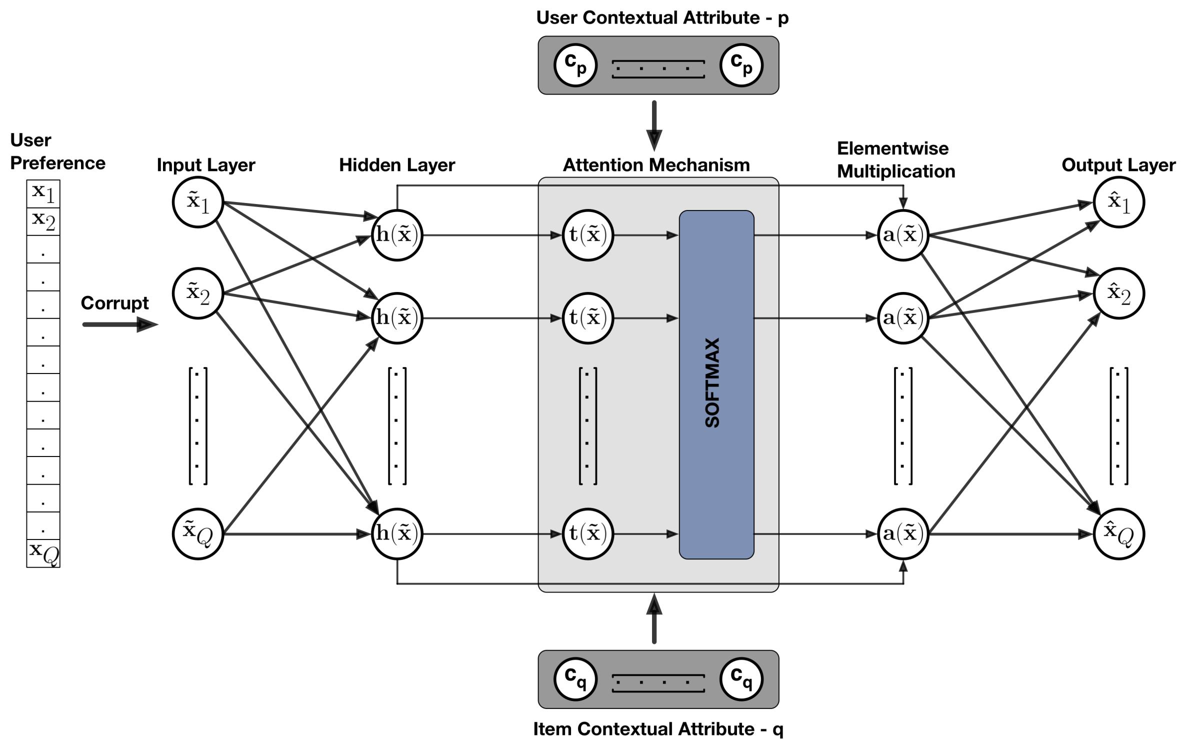


FIGURE 5.1: Attentive Contextual Denoising Autoencoder Architecture

- We thoroughly evaluate our proposed approach on real-world datasets from *Meetup* and *Movielens* on two different tasks: event recommendation and movie recommendation. The results demonstrate the effectiveness of the proposed model compared to the other state-of-the-art models. The code and data are available at `https://github.com/yjhamb/acda.git`.

## 5.2 Attentive Contextual Denoising Autoencoder

First, we present the *Attentive Contextual Denoising Autoencoder (ACDA)* model, which is a generic framework for contextual recommendation. Later we explain how this flexible framework is applied to the event recommendation and movie recommendation tasks.

### 5.2.1 The Architecture

The proposed model, as illustrated in Figure 5.1, is based on the denoising autoencoder neural network architecture. The model takes as input a vector indicating the preference of a user $u$ on all the items $i$ in the dataset. Assuming that there are $m$ users and $n$ items, the autoencoder takes as input a vector $\mathbf{x} \in \mathbb{R}^n$, which is the known preference of the user $u$ on the $n$ items. We corrupt the input vector $\mathbf{x}$ using mask-out/drop-out corruption to obtain $\tilde{\mathbf{x}}$. The corruption method randomly overwrites some of the dimensions of $\mathbf{x}$ with 0 using the probability $\rho$. To offset the effect of the corruption of certain dimensions, we scale the remaining dimensions by applying a factor $\delta$ to the original value: $P(\tilde{\mathbf{x}}_\theta = 0) = \rho$; $P(\tilde{\mathbf{x}}_{\bar{\theta}} = \delta\tilde{\mathbf{x}}) = 1 - \rho$, where $\delta = 1/(1-\rho)$. The symbol $\theta$ denotes the dimensions that are set to 0, whereas $\bar{\theta}$ denotes the dimensions that are scaled. This corruption method is similar to the one used in [121]

The corrupted vector $\tilde{\mathbf{x}}$ is fed into the model to generate the latent hidden representation $h \in \mathbb{R}^k$ using the encoding function $e(\cdot)$. The dimensionality of the hidden representation is represented by $k \ll n$, which is the number of hidden units in the model. The input user preference is corrupted only while training the model, and not during cross-validation and test.

$$h(\tilde{\mathbf{x}}) = e\big(\mathbf{W} \cdot \tilde{\mathbf{x}} + \mathbf{b}\big) \tag{5.1}$$

where $\mathbf{W} \in \mathbb{R}^{k \times n}$ is the weight matrix and $\mathbf{b} \in \mathbb{R}^k$ is a bias vector. The encoding function $e(\cdot)$ is set to the *ReLU* function [85] as it performs well due to its suitability for sparse data ($ReLU(x) = \max(0, x)$).

The hidden representation $h(\tilde{\mathbf{x}})$ is input into the attention mechanism layer, where the contextual attributes are applied. The objective of the attention mechanism is to summarize the input representation based on the provided context. The context may be associated with the user or item. Our model is flexible enough to accommodate as many contextual attributes as desired. However, we have specified two contextual attributes ($p$ and $q$) in our model for ease of presentation. The attention mechanism applies a weighted user-context $\mathbf{c}_p$ and item-based context $\mathbf{c}_q$ for any given contextual parameters $p$ and $q$ to the output of the hidden layer with a nonlinear activation function $f(\cdot)$. Mathematically, this is denoted as:

$$t(\tilde{\mathbf{x}}) = f\big(\mathbf{W}_h \cdot h(\tilde{\mathbf{x}}) + \mathbf{W}_p \cdot \mathbf{c}_p + \mathbf{W}_q \cdot \mathbf{c}_q\big) \tag{5.2}$$

where $\mathbf{W}_h$ is a $\mathbb{R}^{k \times k}$ weight matrix, where $k$ is the number of units in the attention layer, which is the same as the number of units in the hidden layer. $\mathbf{W}_p$ and $\mathbf{W}_q$ are weight matrices of dimensions $\mathbb{R}^{k \times |p|}$ and $\mathbb{R}^{k \times |q|}$ respectively, with $|p|$ and $|q|$ being the number of contextual parameters. $h(\tilde{\mathbf{x}})$ is the output of the hidden layer. We selected *tanh* as the attention mechanism activation function ($f(\cdot)$) as it gave us the best results ($tanh(x) = (\exp(x) - \exp(-x)) \,/\, (\exp(x) + \exp(-x))$).

The output of the attention activation function, $t(\tilde{\mathbf{x}})$, is then fed into a *softmax* layer. Finally, the *softmax* output is combined with the hidden layer output via element-wise multiplication ($\otimes$) to generate the final output of the attention mechanism, which is denoted as $a(\tilde{\mathbf{x}})$.

$$a(\tilde{\mathbf{x}}) = softmax\big(t(\tilde{\mathbf{x}})\big) \otimes h(\tilde{\mathbf{x}}) \tag{5.3}$$

where $t(\tilde{\mathbf{x}})$ is the output of the attention activation function, and $h(\tilde{\tilde{\mathbf{x}}})$ is the hidden layer output. The *softmax* function is defined as: $softmax(x_1, x_2, ..., x_n) = \exp(x_i) \,/\, \sum_{j=1}^{n} \exp(x_j)$.

Essentially, the attention mechanism serves to apply a weighted arithmetic mean to the hidden layer representation, with the weight representing the relevance based on the provided context.

The internal latent representation of the input with the applied context is reconstructed back to the original form using a decoding function $d(\cdot)$.

$$\hat{\mathbf{x}} = d\big(\mathbf{W}' \cdot a(\tilde{\mathbf{x}}) + \mathbf{b}'\big) \tag{5.4}$$

where the dimension of $\mathbf{W}'$ and $\mathbf{b}'$ is the same as $\mathbf{W}$ and $\mathbf{b}$. We would like to point out that the reconstruction of the original input, or the reverse mapping, may be constrained by sharing parameters $\mathbf{W}' = \mathbf{W}^T$. However, we did not do so as we got better results by having different $\mathbf{W}'$ and $\mathbf{b}'$ parameters at the decoding step.

We selected the *sigmoid* function ($\sigma(x) = 1 \,/\, 1 + \exp(-x)$) as the decoding function $d(\cdot)$ as it constraints an input to the $0 - 1$ output range. This gives us the probability associated with each item at the output, and we use this value for ranking in personalized recommendation. The *ADCA* model is generic and it can be applied to a rating prediction task by simply selecting any other nonlinear function as the decoding function $d(\cdot)$.

The parameters of the model are trained by minimizing the mean squared error between the original input vector $\mathbf{x}$ and the reconstructed vector $\hat{\mathbf{x}}$.

$$\min_{\mathbf{W},\mathbf{W}',\mathbf{W}_h,\mathbf{W}_p,\mathbf{W}_q,\mathbf{b},\mathbf{b}'} \frac{1}{m}\sum_{u\in U} \|\mathbf{x}_u - \hat{\mathbf{x}}_u\|^2 \tag{5.5}$$

The parameters are updated using the stochastic gradient descent variant ADAM optimizer [60]. We also used dropout [34] for regularization to prevent overfitting and improve generalization capacity. We set the dropout rate to be 0.2, which means that 20% of the hidden units are dropped at random at each training step to prevent co-adaptation.

### 5.2.2 Top-$n$ Recommendation

The proposed *ACDA* model can be applied to both rating prediction and top-$n$ recommendation by simply changing the decoding function $d(\cdot)$. We set the decoding function $d(\cdot)$ to the *sigmoid* function for top-$n$ recommendation, and apply the generic *ACDA* model to the event recommendation and movie recommendation tasks.

The event recommendation task utilizes the RSVP [2] data from *Meetup*. Users indicate their preference to an event by providing an RSVP, which is used to recommend future

[2]RSVP is a French expression, which means “please respond"

events to the user. For the event recommendation task, we utilize the user *group* and event *venue* as the contextual attributes. Users typically organize themselves into groups in an Event Based Social Network (EBSN) such as *Meetup*, and each event is hosted at a physical venue. The user's preference on existing events in the training set is input into the model as a binary $k$-hot encoded vector with a true value for the positive event preferences and false for the negative or unknown event preferences. The input preference is corrupted as discussed earlier in section 5.2.1. In addition to corrupting the input, we also include a fixed number of negative samples by encoding them as positive in the input vector. The negative samples are selected randomly from the training set and negative sample inclusion is only performed during training, not during evaluation on the cross-validation and test sets. The output of the model is the personalized top-$n$ event recommendation for the user.

For the event recommendation task, the contextual attributes of the model are set as: $\mathbf{c}_p = \mathbf{u}_g$ and $\mathbf{c}_q = \mathbf{i}_v$, where $\mathbf{u}_g \in \mathbb{R}^{|p|}$ denotes the groups that the user belongs to. The parameter $\mathbf{i}_v \in \mathbb{R}^{|q|}$ denotes the venues associated with the events. The parameters $|p|$ and $|q|$ are the number of groups and venues respectively.

We also apply the *ACDA* model to movie recommendation, which is also treated as a top-$n$ recommendation task. The *Movielens* dataset contains the movie ratings on a scale of $1 - 5$, which we convert to a binary scale. The movie binary scale indicates a user's preference on existing movies, which is used to recommend other movies to the user. We select the movie *genre* as a contextual attribute for our model. The *genre* is associated with each movie, and certain movies have multiple genres associated with them. Similar to the event recommendation task, the user's preference is partially corrupted and input into the model as a binary $k$-hot encoded vector. We also used negative samples during training. The output of the model is the personalized top-$n$ movie recommendation for the user.

Since we have only one item-related contextual attribute for the movie recommendation task, we update the model as: $\mathbf{c}_q = \mathbf{i}_r$ where $\mathbf{i}_r \in \mathbb{R}^{|q|}$ denotes the genres associated with the movies preferred by the user. The parameter $|q|$ is the number of genres.

## 5.3 Experimental Results

### 5.3.1 Datasets

We evaluate the proposed *Attentive Contextual Denoising Autoencoder (ACDA)* model on real-world datasets from *Meetup* and *Movielens.* The Meetup dataset is for events from *New York*, *San Francisco*, *Washington DC* and *Chicago.* These cities were selected as they are the major metropolitan areas in the United States and they have a vibrant Meetup community. The event data was collected by using the Meetup API[3] between January 2016 and May 2016. We also analyzed our model against the publicly available *Movielens (100K)* dataset. The *Movielens* dataset consists of movie ratings provided by the user on the $1-5$ scale. We converted the numeric rating score to a binary rating for the purpose of top-$n$ recommendation. A score of 5 is converted to a binary rating

[3]`http://www.meetup.com/meetup_api/`

Table 5.1: Data Statistics

| Dataset | Observations | Sparsity | Positive | Negative | Users | Items |
|---|---|---|---|---|---|---|
| Meetup-NYC | 73,816 | 0.9998 | 70,170 | 3,646 | 19,122 | 36,054 |
| Meetup-SFO | 48,972 | 0.9998 | 43,637 | 5,335 | 18,957 | 14,445 |
| Meetup-DC | 36,451 | 0.9998 | 33,541 | 2,901 | 10,384 | 12,359 |
| Meetup-Chicago | 22,915 | 0.9996 | 20,826 | 2,089 | 8,118 | 9,133 |
| Movielens | 100,004 | 0.9835 | 15,095 | 84,909 | 671 | 9,066 |

of 1, and anything less than a 5 is converted to 0. The statistics of the datasets used for the experiments are given in Table 5.1.

### 5.3.2 Experimental Setup

We split the datasets to use 60% as the training set, 20% as the cross-validation set, and 20% as the test set. The evaluation metrics include $P@5$, $P@10$, $R@5$, $R@10$, $NDCG@5$, $NDCG@10$, $MAP@5$ and $MAP@10$ [79]. These are common metrics for top-$n$ recommendations. We consider baselines methods from each of the following categories for comparison against the proposed *ACDA* model: Neighborhood-based Methods (*UserKNN*, *ItemKNN*), Model-based Methods (*BiasedMF*, *BPR-MF*, *SVD++*), and Deep Learning Methods (*CDAE*, *U-AutoRec*). The results are presented in this section and we discuss our findings in detail.

We use *Librec*[4], a recommender library, to obtain results for the neighborhood and model-based methods. We use our own implementation of the deep learning baseline models. The parameter values for the existing methods are similar to the proposed method (to the extent possible).

[4]`http://www.librec.net`

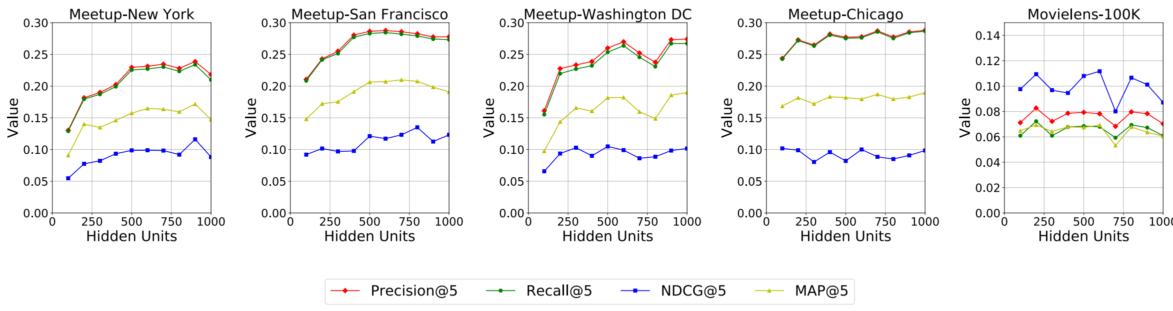


FIGURE 5.2: Hidden Unit Count Selection

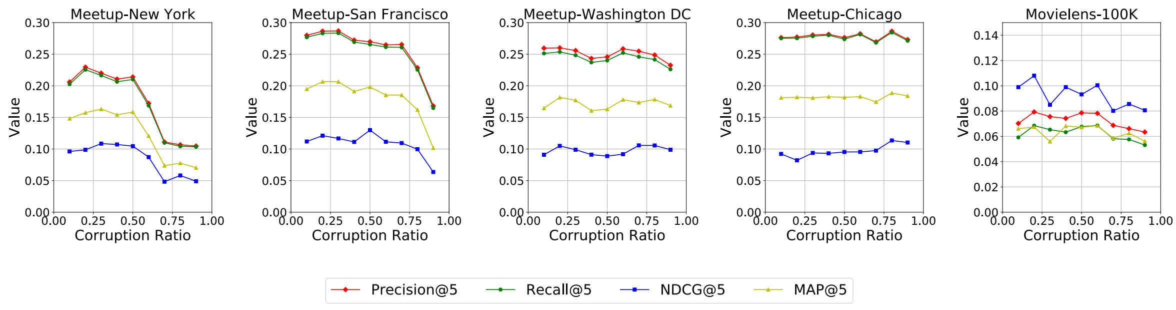


FIGURE 5.3: Corruption Ratio Selection

- *User-KNN*: User $k$-nearest neighborhood collaborative filtering method that predicts the user preference based on the similarity with the $k$ nearest users. We selected $k = 10$ as it gave the best results.

- *Item-KNN*: Item $k$-nearest neighborhood collaborative filtering method that predicts the user preference based on the similarity with the $k$ nearest items. We set the value of $k = 10$ to be consistent with *User-KNN*.

- *BPR-MF*: Bayesian personalized ranking method that utilizes pairwise loss to provide top-$n$ item recommendation using matrix factorization (MF). The latent factor count is set to $l = 50$ as it offered the best performance.

- *Biased-MF*: Basic matrix factorization that includes global mean, user bias and item bias. We set the latent factor count $l = 50$ to be consistent with the *BPR-MF* method.

- *SVD++*: State-of-the-art matrix factorization method that incorporates implicit feedback from the user into the baseline SVD model for better accuracy. We set the latent factor count $l = 50$ to be consistent with the *BPR-MF* method.

- *CDAE*: Collaborative filtering technique based on denoising autoencoders that incorporates the user latent factor as additional input [121].

- *U-AutoRec*: Collaborative filtering technique based on denoising autoencoders [103] that has two variants: *I-AutoRec*, which accepts the $k$-hot encoded item preference vector consisting of users as input, and *U-AutoRec* that accepts the $k$-hot encoded user preference vector of items. We compared against the *U-AutoRec* variant as it is similar to our proposed *ACDA* model in terms of the user preference on items being provided as input.

We evaluate the proposed models to incorporate the influence of the different contextual attributes for the event and movie recommendation tasks.

- *ACDA-V*: This is the variant of the proposed generic *ACDA* model that incorporates only the event *venue* as a contextual attribute for the event recommendation task.

- *ACDA-G*: A variant of the proposed generic *ACDA* model that just incorporates the user *group* as a contextual attribute for the event recommendation task.

- *ACDA-GV*: This model includes both the user *group* and event *venue* as contextual attributes of the *ACDA* model for the event recommendation task.
- *ACDA-R*: This model includes the movie *genre* as a contextual attribute of the *ACDA* model for the movie recommendation task.

We did not include the basic *ACDA* model (without the contextual attributes) into the comparison as that is basically the *U-AutoRec* model, which we have considered as a baseline method. The proposed *ACDA* models are trained on training set and then evaluated on the cross-validation set for selecting the appropriate values for the hyper-parameters. Finally, the model is evaluated on the test set, the results of which are published for comparison with the baselines. The proposed models are developed and trained using Google's tensorflow library[5]. We conducted additional experiments to determine the optimal value for the hidden unit size and corruption ratio hyper-parameters. The results of the additional experiments are provided in Section 5.3.3.

We selected the $epoch = 200$ during training as we found the model to converge at this point. We experimented with different learning rates $(0.1, 0.01, 0.05, 0.001, 0.005)$ and found the learning rate $\alpha = 0.001$ to work best. To prevent the model from just training on positive samples, we paired the positive samples of a user with a configurable number of negative or unknown samples for the user.

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

TABLE 5.2: Experimental Results - New York

| Method | P@5 | P@10 | R@5 | R@10 | NDCG@5 | NDCG@10 | MAP@5 | MAP@10 |
|---|---|---|---|---|---|---|---|---|
| UserKNN | 0.0069 | 0.0034 | 0.0220 | 0.0237 | 0.0223 | 0.0232 | 0.0186 | 0.0181 |
| ItemKNN | 0.0076 | 0.0038 | 0.0241 | 0.0254 | 0.0213 | 0.0222 | 0.0194 | 0.0191 |
| Biased-MF | 0.0003 | 0.0002 | 0.0001 | 0.0002 | 0.0002 | 0.0003 | 0.0008 | 0.0007 |
| BPR-MF | 0.0501 | 0.0342 | 0.1266 | 0.1524 | 0.0766 | 0.0825 | 0.1061 | 0.1089 |
| SVD++ | 0.0005 | 0.0004 | 0.0003 | 0.0006 | 0.0004 | 0.0006 | 0.0001 | 0.0001 |
| CDAE | 0.1035 | 0.1477 | 0.1123 | 0.1657 | 0.0707 | 0.0791 | 0.0788 | 0.0994 |
| U-AutoRec | 0.0527 | 0.0804 | 0.0508 | 0.0790 | 0.0272 | 0.0305 | 0.0334 | 0.0517 |
| ACDA-V | 0.1086 | 0.1844 | 0.1066 | 0.1834 | 0.0523 | 0.0541 | 0.0739 | 0.1151 |
| ACDA-G | 0.1781 | 0.2320 | 0.1760 | 0.2309 | 0.0860 | 0.0871 | 0.1337 | 0.1738 |
| ACDA-GV | **0.2295** | **0.2905** | **0.2255** | **0.2990** | **0.0987** | **0.0994** | **0.1574** | **0.2116** |

TABLE 5.3: Experimental results - San Francisco

| Method | P@5 | P@10 | R@5 | R@10 | NDCG@5 | NDCG@10 | MAP@5 | MAP@10 |
|---|---|---|---|---|---|---|---|---|
| UserKNN | 0.0166 | 0.0127 | 0.0401 | 0.0615 | 0.0317 | 0.0390 | 0.0232 | 0.0255 |
| ItemKNN | 0.0155 | 0.0131 | 0.0394 | 0.0591 | 0.0303 | 0.0381 | 0.0229 | 0.0245 |
| Biased-MF | 0.0004 | 0.0004 | 0.0001 | 0.0003 | 0.0005 | 0.0004 | 0.0002 | 0.0001 |
| BPR-MF | 0.0552 | 0.0376 | 0.1486 | 0.1809 | 0.0860 | 0.0977 | 0.1217 | 0.1254 |
| SVD++ | 0.0014 | 0.0009 | 0.0011 | 0.0017 | 0.0017 | 0.0016 | 0.0009 | 0.0007 |
| CDAE | 0.1109 | 0.1877 | 0.1098 | 0.1909 | 0.0519 | 0.0564 | 0.0804 | 0.1129 |
| U-AutoRec | 0.1045 | 0.1525 | 0.1020 | 0.1513 | 0.0634 | 0.0686 | 0.0730 | 0.1084 |
| ACDA-V | 0.1793 | 0.2623 | 0.1773 | 0.2616 | 0.0765 | 0.0789 | 0.1226 | 0.1770 |
| ACDA-G | 0.1879 | 0.3061 | 0.1649 | 0.3049 | 0.0602 | 0.0622 | 0.1004 | 0.1825 |
| ACDA-GV | **0.2864** | **0.3708** | **0.2830** | **0.3692** | **0.1211** | **0.1266** | **0.2065** | **0.2743** |

Table 5.4: Experimental results - Washington DC

| Method | P@5 | P@10 | R@5 | R@10 | NDCG@5 | NDCG@10 | MAP@5 | MAP@10 |
|---|---|---|---|---|---|---|---|---|
| UserKNN | 0.0013 | 0.0006 | 0.0039 | 0.0045 | 0.0023 | 0.0025 | 0.0016 | 0.0017 |
| ItemKNN | 0.0011 | 0.0005 | 0.0042 | 0.0042 | 0.0027 | 0.0027 | 0.0018 | 0.0019 |
| Biased-MF | 0.0003 | 0.0001 | 0.0003 | 0.0022 | 0.0002 | 0.0010 | 0.0007 | 0.0003 |
| BPR-MF | 0.0588 | 0.0466 | 0.1188 | 0.1509 | 0.0688 | 0.0753 | 0.1068 | 0.1098 |
| SVD++ | 0.0007 | 0.0007 | 0.0001 | 0.0006 | 0.0012 | 0.0011 | 0.0007 | 0.0004 |
| CDAE | 0.0983 | 0.1860 | 0.0914 | 0.1812 | 0.0459 | 0.0515 | 0.0663 | 0.1089 |
| U-AutoRec | 0.0905 | 0.1136 | 0.0847 | 0.1083 | 0.0560 | 0.0606 | 0.0731 | 0.0885 |
| ACDA-V | 0.1737 | 0.2498 | 0.1676 | 0.2455 | 0.0836 | 0.0863 | 0.1177 | 0.1713 |
| ACDA-G | 0.1956 | 0.2833 | 0.1886 | 0.2792 | 0.0777 | 0.0794 | 0.1198 | 0.1886 |
| ACDA-GV | **0.2600** | **0.3472** | **0.2536** | **0.3430** | **0.1049** | **0.1092** | **0.1816** | **0.2476** |

Table 5.5: Experimental results - Chicago

| Method | P@5 | P@10 | R@5 | R@10 | NDCG@5 | NDCG@10 | MAP@5 | MAP@10 |
|---|---|---|---|---|---|---|---|---|
| UserKNN | 0.0065 | 0.0041 | 0.0204 | 0.0213 | 0.0160 | 0.0170 | 0.0122 | 0.0125 |
| ItemKNN | 0.0062 | 0.0037 | 0.0193 | 0.0202 | 0.0157 | 0.0166 | 0.0118 | 0.0120 |
| Biased-MF | 0.0004 | 0.0003 | 0.0002 | 0.0006 | 0.0002 | 0.0005 | 0.0008 | 0.0001 |
| BPR-MF | 0.0498 | 0.0311 | 0.1640 | 0.1925 | 0.0952 | 0.1059 | 0.1277 | 0.1322 |
| SVD++ | 0.0020 | 0.0014 | 0.0005 | 0.0012 | 0.0020 | 0.0020 | 0.0010 | 0.0008 |
| CDAE | 0.1271 | 0.2097 | 0.1260 | 0.2095 | 0.0428 | 0.0438 | 0.0724 | 0.1297 |
| U-AutoRec | 0.0879 | 0.1209 | 0.0878 | 0.1209 | 0.0476 | 0.0479 | 0.0621 | 0.0855 |
| ACDA-V | 0.2354 | 0.3356 | 0.2339 | 0.3353 | 0.0805 | 0.0796 | 0.1493 | 0.2261 |
| ACDA-G | **0.2866** | **0.3994** | **0.2849** | **0.3988** | **0.1375** | **0.1395** | **0.2052** | **0.2835** |
| ACDA-GV | 0.2771 | 0.3856 | 0.2752 | 0.3849 | 0.0821 | 0.0847 | 0.1819 | 0.2655 |

Table 5.6: Experimental results - Movielens 100K

| Method | P@5 | P@10 | R@5 | R@10 | NDCG@5 | NDCG@10 | MAP@5 | MAP@10 |
|---|---|---|---|---|---|---|---|---|
| UserKNN | 0.0200 | 0.0185 | 0.0052 | 0.0082 | 0.0200 | 0.0204 | 0.0109 | 0.0089 |
| ItemKNN | 0.0195 | 0.0164 | 0.0055 | 0.0079 | 0.0205 | 0.0194 | 0.0097 | 0.0081 |
| Biased-MF | 0.0008 | 0.0008 | 0.0002 | 0.0004 | 0.0007 | 0.0008 | 0.0002 | 0.0002 |
| BPR-MF | 0.0761 | 0.0603 | **0.1022** | **0.1477** | 0.0907 | 0.0997 | 0.0611 | 0.0691 |
| SVD++ | 0.0017 | 0.0011 | 0.0001 | 0.0001 | 0.0016 | 0.0013 | 0.0010 | 0.0006 |
| CDAE | 0.0595 | 0.0887 | 0.0533 | 0.0869 | 0.0664 | 0.0782 | 0.0428 | 0.0613 |
| U-AutoRec | 0.0691 | 0.0986 | 0.0609 | 0.0962 | 0.0932 | 0.1017 | 0.0631 | 0.0718 |
| ACDA-R | **0.0827** | **0.1106** | 0.0723 | 0.1070 | **0.1094** | **0.1189** | **0.0694** | **0.0835** |

### 5.3.3 The Effect of Hidden Units and Corruption Ratio

To investigate the effect of the number of hidden units on the performance, we experimented with different values of $k$, ranging from 100 to 1000 in increments of 100. The results are provided in Figure 5.2. As observed from the plots, we found that the performance of the model plateaus after $k = 500$, with higher values offering no significant gain in performance at a cost of increased training time. While there are certain metrics, such as the *NDCG@5*, that perform slightly better at higher values $k$, we set $k = 500$ as a default choice.

We also experimented with different values of the corruption ratio ranging from 0.1 to 0.9 in increments of 0.1. The results, depicted in Figure 5.3, indicate that the performance degrades with higher values of the corruption ratio. The only exception to this is the *Meetup-Chicago* dataset, which does not have a observable degradation in performance at higher values of the corruption ratio. Therefore, we default the value of the corruption ratio $\rho = 0.2$.

### 5.3.4 Baseline Comparisons

Tables 5.2, 5.3, 5.4, 5.5, 5.6 contains the results of the different methods, with the best results highlighted in boldface. A general observation is that, other than a few exceptions, the results on the *precision*, *recall*, *NDCG* and *MAP* metrics were consistent across all the datasets. The proposed model performed well on the *Meetup* and *Movielens* datasets, which demonstrates its effectiveness on top-$n$ recommendation tasks.

First, we discuss the performance of the baseline methods. We considered three different categories of the baseline methods:

neighborhood-based, model-based and deep learning based methods. Among these categories, we found the deep learning based baseline methods to perform better than the others. In general, across the baseline methods, the *CDAE* deep learning based method performs better on the precision and recall metrics. The *BPR-MF* is better on the *NDCG* metric. The *CDAE* method is based on the denoising autoencoder, and the results signify its importance to recommender systems. The good performance of the *BPR-MF* method may be attributed to the use of the pairwise loss function. We also observe good results for the *BPR-MF* method against the *Movielens* dataset. However, we found *U-AutoRec* to perform better than *CDAE* against the *Movielens* dataset. This suggests that the user latent factor included in the *CDAE* model does not help to improve the performance against the *Movielens* dataset, but it does so against the *Meetup* dataset. When considering the neighborhood methods, we found both (*UserKNN* and *ItemKNN*) to be similar in performance.

Comparing the baseline methods to the proposed models for the event recommendation task, we observed that all three variants of the proposed *ACDA* model perform better than the baselines. While a variant of the *ACDA* model with some of the contextual attributes may perform better, in general the model with more contextual attributes offers the better performance. As we can see for the *Meetup* datasets, the *ACDA-GV* model offers a better performance in three of the four cities. We also observe that the significance of the contextual attributes is not equal. The influence of the user *group* contextual attribute is higher than the event *venue* attribute, and the model *ACDA-G* performs better than the *ACDA-V* model. This implies that additional contextual parameters may improve the performance further in some cases, however, this may not be always true. With regard to the movie recommendation task, we utilize the movie *genre* as a contextual attribute. The model *ACDA-R* performs better on all metrics except the *recall*. The *BPR-MF* method is better on the *recall* metric, perhaps due to the fact that it uses a pairwise loss function. We intend to evaluate the performance of the proposed models using pairwise loss in the future. The results, which are consistent across all datasets, reinforce our assertion that the proposed *ACDA* model performs well on recommendation tasks.

## 5.4 Summary

We propose a deep learning architecture for contextual recommendation based on denoising autoencoder augmented with a context-driven attention mechanism. We also perform comprehensive experiments demonstrating that the proposed *ACDA* model outperforms the state-of-the-art baselines on event recommendation and movie recommendation tasks.

We understand that this preliminary study can be extended in many directions and we plan to do so in future work. First of all, we will investigate other types of loss functions including pairwise and listwise losses which demonstrate good performance for ranking tasks especially with implicit user feedback [76]. Secondly, we will explore deeper architectures by adding more layers and experimenting with different activation functions. Last, we will conduct experiments in other domains of contextual recommendation.

# Chapter 6

# Neural Citation Network

## 6.1 Introduction

Authors establish credibility, honesty, and authority by providing accurate and relevant citations. The vast plethora of scientific literature makes searching for relevant work time consuming and highly keyword dependent. On the other hand, following the proceedings of well-known conferences restricts the scope of related work. Ideally, we desire a personalized, curated list of high-quality recommendations. We focus on the task of context-aware citation recommendation, where given a citation context (query), we recommend a list of high-quality candidate papers to fill the citation placeholder. A citation context comprises a small window of words surrounding a placeholder denoting where the citation should appear [49, 47, 78, 36, 4], see Figure 6.1 for an example. We assume the surrounding text of a placeholder provides a short and concise summary of the paper's content.

Traditional information retrieval techniques rely heavily on keyword overlap, but identifying the critical structures in abstract ideas requires additional levels of semantic relations. For example, “deep learning" was previously known as “cybernetics" in its infancy and “connectionism" in its second resurgence [34]. As language evolves over time, new terms emerge while others become less frequently used. Similarly, the denotative meaning of words are generally fixed, perhaps more importantly, the connotative meaning changes throughout time. The words “deep" and “learning" treated independently as a bag-of-words lacks conceptual interpretation but modeling the conditional probability of the words together produces a clear concept. The word usage between the content in the citation context and corresponding cited document lead to a vocabulary gap [78, 47, 49] causing a mismatch between keywords leading to poor performance with standard information retrieval (IR) methods. In addition, existing methods cannot easily incorporate metadata without additional feature engineering or explicitly linked data [4].

We propose Neural Citation Network (NCN)[1] an encoder-decoder framework inspired by the success of neural machine translation (NMT) [16, 2, 124] which can learn relations between parallel pairs of variable-length text. Consequently, NCN is capable of characterizing the semantic composition of citation contexts and corresponding cited

[1]Source code: `https://github.com/tebesu/NeuralCitationNetwork`.

| **Citation Context** |
|---|
| . . . the original language modeling approach proposed in [●]; that is, we first estimate a document language model and then compute . . . |
| **Cited Paper** |
| Jay M. Ponte and W. Bruce Croft. 1998. A Language Modeling Approach to Information Retrieval. (SIGIR '98). |

FIGURE 6.1: Example citation context with a placeholder denoted by [●] indicating where the corresponding cited paper would appear.

documents title by exploiting author relations. The encoder capitalizes on the computational advantages of a max time delay neural network [17] while the decoder leverages the capacity of recurrent neural networks (RNN) influenced by both the author networks and attention mechanism. As each composer of literature has her own writing style, grammatical structure, word usage and citation preference. NCN leverages these associated attributes with each author by utilizing only their name, producing significant performance gains. Furthermore, NCN can generalize to new papers not present in the training set. To the best of our knowledge, no prior work has addressed citation recommendation with the encoder-decoder framework. Experimental results on the CiteSeer dataset demonstrate NCN produces a significant improvement Recall, Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) and Normalized Discounted Cumulative Gain (NDCG) over baseline methods. Qualitative results demonstrate the effectiveness of the proposed end-to-end neural network.

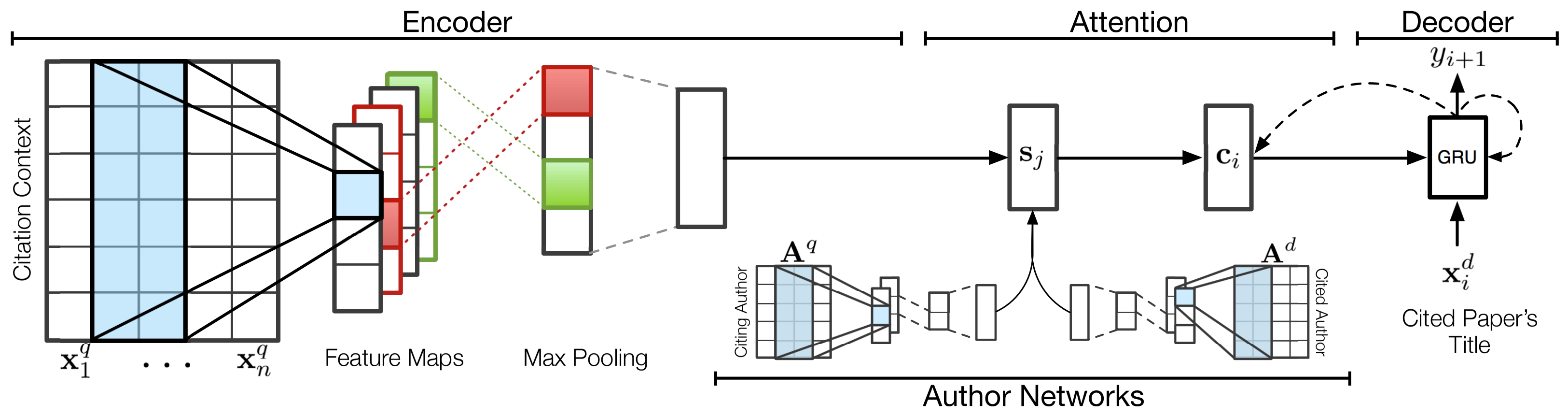


FIGURE 6.2: The proposed architecture of Neural Citation Network (NCN) with the attention mechanism and author networks. The dashed arrows represent recurrent dependencies.

## 6.2 Neural Citation Network

We introduce Neural Citation Network (NCN) an architecture unifying the strengths of TDNN and RNNs under the encoder-decoder framework. The fused architecture is capable of capturing the semantic representations of the citation context and authors' conditioned on the corresponding cited paper's title in an end-to-end fashion. The proposed model is based on the encoder-decoder architecture with the attention mechanism [2] to integrate complementary author information and learn rich feature representations. An illustration of the proposed architecture is presented in Figure 6.2. We would like to note that this Chapter's notations slightly deviates from those used in earlier parts of this thesis.

### 6.2.1 Encoder

In our encoder we leverage the TDNN [17] a CNN variant designed to capture long-term dependencies with a 1-dimension convolution over all possible word windows for a given context. A non-linear projection coupled with max-pooling extracts rich feature representations from each convolved word window. Specifically, given a citation context of length $n$, let $\mathbf{x}_t^q$ be a $g$ dimensional word embedding corresponding to the $t^{th}$ word in the citation context and $\mathbf{x}_{1:n}^q = \mathbf{x}_1^q \oplus \ldots \oplus \mathbf{x}_n^q$ denote the concatenation of the embeddings from 1 to $n$. A convolutional filter $\mathbf{w} \in \mathbb{R}^{l \cdot g}$ slides over $l$ words or regions at a time over all possible window lengths $\{\mathbf{x}_{1:l}^q, \mathbf{x}_{2:l+1}^q, \ldots, \mathbf{x}_{n-l+1:n}^q\}$, see Figure 6.2. We define the convolutional layer as:

$$o_k = \text{ReLU}(\mathbf{w}^\mathsf{T}\mathbf{x}^q_{k:k+l-1} + b_k) \tag{6.1}$$

$$\hat{o} = \max\{o_1, \ldots, o_{n-l+1}\} \tag{6.2}$$

where ReLU is the nonlinear activation function $\max(0, x)$ and $o_k$ is the $k^{th}$ feature map, $\mathbf{o} \in \mathbb{R}^{n-l+1}$. The max-pooling over time yields a scalar representing the relevant feature $\hat{o}$ detected for the given set of feature maps subsequently converting the variable length sequence to a fixed one. In order to capture more complex relations the process is repeated $p$ times with different filter weights yielding $\hat{\mathbf{o}}_j \in \mathbb{R}^p$. Finally, a fully connected layer allow interactions between the various phrase level feature maps extracted from the max-pooling layer, leading to:

$$\mathbf{s}_j = \tanh(\mathbf{U}_{s_j}\hat{\mathbf{o}}_j + \mathbf{b}_{s_j}) \tag{6.3}$$

where the TDNN aims to project the raw citation context $\mathbf{X}^q$, to a fixed summary representation $\mathbf{s}_j$ over feature maps of the $j^{th}$ sliding region size of $l_j$. The final transformation $f(\mathbf{X}^q)$ applies a set of variable region size filters $L = \{l_1, \ldots, l_{|L|}\}$ to capture different granularity of phrases e.g. bigrams, trigrams. The TDNN exploits the property of parallelism allowing all feature maps to be computed in parallel yet obtaining competitive performance with an RNN encoder (Section 6.3.2). The phrase level representation obtained by the TDNN provides a trade-off between capturing semantics and computational time.

### 6.2.2 Decoder

Since the title of a manuscript is short but more concise, we require a finer grain representation than the phrase level of the TDNN. We adopt an RNN to represent the decoder with its large capacity to condition each word on all previous words in the sequence while considering its internal state and the encoder's representation. Let $\mathbf{x}_i^d$ be a $e$ dimensional embedding corresponding to the $i^{th}$ word of the cited document's title of length $m$. The RNN conditions the input sequence over the entire encoder representation and all previous recurrent hidden states. We utilize the Gated Recurrent Unit (GRU) [16] to help prevent the vanishing or exploding gradient problem, formally:

$$\begin{aligned}
\mathbf{z}_i &= \sigma(\mathbf{W}_z\mathbf{x}_i^d + \mathbf{V}_z\mathbf{c}_i + \mathbf{U}_z\mathbf{h}_{i-1}) \\
\mathbf{r}_i &= \sigma(\mathbf{W}_r\mathbf{x}_i^d + \mathbf{V}_r\mathbf{c}_i + \mathbf{U}_r\mathbf{h}_{i-1}) \\
\tilde{\mathbf{h}}_i &= \tanh(\mathbf{W}_o\mathbf{x}_i^d + \mathbf{V}_o\mathbf{c}_i + \mathbf{r}_i \circ \mathbf{U}_o\mathbf{h}_{i-1}) \\
\mathbf{h}_i &= (1 - \mathbf{z}_i) \circ \tilde{\mathbf{h}}_i + \mathbf{z}_i \circ \mathbf{h}_{i-1}
\end{aligned} \tag{6.4}$$

where $\mathbf{W}_{[z,r,o]}, \mathbf{V}_{[z,r,o]}, \mathbf{U}_{[z,r,o]}$ are weight matrices to be learned, $\tilde{\mathbf{h}}_i$ is the new updated hidden state, $\mathbf{z}_i$ is the update gate, $\mathbf{r}_i$ is the reset gate, $\sigma(\cdot)$ is the sigmoid function and $\circ$ is the element wise product.

Although the max pooling layer obtains the most relevant features present for a given filter, it treats each feature map with uniform importance and words on the margins of the sequence are neglected. The attention mechanism learns a weighted interpolation $\mathbf{c}_i$ dependent on all of the encoder's representation conditioned on previous decoder states obtaining a richer representation with:

$$\mathbf{c}_i = \sum_j \alpha_{ij}\mathbf{s}_j \tag{6.5}$$

$$\alpha_{ij} = \text{softmax}(\mathbf{v}^{\mathsf{T}} \tanh(\mathbf{W}_a\mathbf{h}_{i-1} + \mathbf{U}_a\mathbf{s}_j)) \tag{6.6}$$

where $\alpha_{ij}$ is the alignment between the $i^{th}$ word and the $j^{th}$ output from the encoder parametrized as a feedforward neural network followed by a softmax function [34]. Figure 6.2 illustrates these recurrent dependencies with dashed arrows.

### 6.2.3 Author Networks

The author(s) of a manuscript may have a large impact on the audience, popularity, and citations. Frequently, one may follow specific researchers or groups with similar interests. The lead author of a paper may hold the most authority. On the other hand, the most influential author may not necessarily be the first author. To capture the most prominent author, we consider both the citing (context) and cited (title) manuscript authors with a shared embedding space, but learn two separate TDNNs. Intuitively, the author's characteristics may remain static hence the shared embedding space but the author has no direct control over if she will be cited or not (with the exception of self-citation). For example, a popular author may be frequently cited yet citations may not be reciprocated leading to distinct roles. We treat each author as a token by denoting $\mathbf{A}^q$ and $\mathbf{A}^d$ as the embeddings of the citation context (query) and cited paper's (document) author(s), respectively. Similar to the encoder representation presented in Section 6.2.1, we exploit the TDNN to learn higher level joint author interactions with:

$$\mathbf{s}_j = [f(\mathbf{X}^q) \oplus f(\mathbf{A}^q) \oplus f(\mathbf{A}^d)]_j \tag{6.7}$$

By concatenating the citation context summary with the author's representation, the attention mechanism conditions on the author networks in addition to the encoder's output. Hence an interaction between the composition of the context and author takes place over the course of the decoding process. The final output from the RNN decoder is projected into a softmax layer producing a probability over the vocabulary:

$$P(y_i|y_{\leq i}, \mathbf{s}) = \text{softmax}(\mathbf{V}\mathbf{h}_i) \tag{6.8}$$

where $P(y_i|y_{\leq i}, \mathbf{s})$ denotes the conditional probability of all previous words in the cited papers title prior to $i$. Since the entire architecture is differentiable, we jointly training the encoder-decoder via stochastic gradient descent (SGD) [34] maximizing the following:

$$\log P(\mathbf{y}|\mathbf{X^q}, \mathbf{X^d}, \mathbf{A^q}, \mathbf{A^d}) = \sum_{i}^{m} \log P(y_i|y_{\leq i}, \mathbf{s}) \tag{6.9}$$

| | **Recall** | **MAP** | **MRR** | **NDCG** |
|---|---|---|---|---|
| BM-25 | 0.1007 | 0.0556 | 0.0606 | 0.0676 |
| CTM | 0.1288 | 0.0726 | 0.0777 | 0.0875 |
| RNN-to-RNN | 0.1590 | 0.0958 | 0.1054 | 0.1134 |
| TDNN-to-RNN | 0.1579 | 0.0935 | 0.1032 | 0.1114 |
| Neural Citation Network | **0.2910** | **0.2418** | **0.2667** | **0.2592** |

Table 6.1: Performance comparison of the top 10 recommendations on Recall, MAP, MRR, and NDCG. (NCN is statistically significant from all baselines on a paired $t$-test $p < 0.001$)

Once the network is fully trained we can score a cited document $\mathbf{y}$ given a citation context $\mathbf{X}^q$ and author information $\mathbf{A}^q, \mathbf{A}^d$ with Equation 6.9.

## 6.3 Experimental Results

### 6.3.1 Setup

We evaluate NCN on the RefSeer dataset [2] [49]. After preprocessing invalid entries, we obtain 4,549,267 context pairs with 855,735 papers in a citation-cited relation. Similar to [49], we divide the data by year, where papers before, after, and equal to 2013 yield 4,258,383 training; 148,927 testing; and 141,957 validation citation contexts respectively. For text preprocessing, we perform tokenization, lemmatization and take the top 20K most frequent terms on the encoder and decoder sides, where words not on this list are replaced with a special <UNK> token. We also take top 20K most frequently cited authors by name and consider the first 5 authors per paper for simplicity. Authors not on the short list are replaced with a with a special $\text{<UNK>}_{\text{Author}}$.

[2] `http://refseer.ist.psu.edu/data/`

All hyperparameters are determined according to the validation set. For clarity, we set all embedding sizes, batch sizes, RNN memory cell sizes and feature maps to 64. We apply gradient clipping at 5, dropout probability to 0.2 and the number of recurrent layers to two for both the encoder (when applicable) and decoder. For the NCN encoder, convolutional filters use region sizes: 4, 4, 5 and author networks use region sizes: 1, 2. We use the Adam optimizer [34] for a total of 5 training iterations, A learning rate of 0.001, using the Adam optimizer with $\beta_1 = 0.9$, and $\beta_2 = 0.999$. taking approximately 10 hours to train NCN on a NVIDIA Titan X.

We report the following metrics: Recall, Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) and Normalized discounted cumulative gain (NDCG) on the test set. For NCN, we rerank the top 2048 documents retrieved by BM-25 with Equation 6.9 and include the ground truth if it is not present.

### 6.3.2 Baselines

We validate the effectiveness of NCN against four baselines:

- BM-25 is a standard information retrieval baseline
- Citation Translation Model (CTM) [47]: learns a translation model between the citation contexts and cited papers title using the GIZA++ toolkit[3].
- TDNN-to-RNN: follows the NCN formulation excluding author networks.

[3]https://github.com/moses-smt/giza-pp

- RNN-to-RNN is identical to TDNN-to-RNN but utilizes a RNN as the encoder.

Table 6.1 demonstrates NCN outperforms all baselines on every metric by 13-16%. BM-25 displays the poorest performance verifying the existence of the vocabulary gap while CTM[4] improves upon standard IR methods but the bag-of-words assumption lacks sufficient capacity to capture complex relations. Since NCN without author content degenerates to the TDNN-to-RNN model, we clearly see the advantages of incorporating author information. RNN-to-RNN marginally outperforms the TDNN-to-RNN model, however, the additional computational overhead may not justify the 0.3% increase in performance taking 11 hours to train yet NCN produces superior performance in less time. We observe smaller performance gains on position aware metrics in NCN when varying the number of recommendations. An improvement of 1.6% on NDCG, 2.4% on MAP and MRR when cutting off the number of recommendations at 10 versus 1 as illustrated in Figure 6.3.

### 6.3.3 Qualitative Study

The top three recommendations by NCN, CTM and RNN-to-RNN for the context (query) are listed in Table 6.2. Both baselines correctly recommend one item and NCN provides two correct recommendations; however, the incorrect recommendation (2) appears to be a plausible citation. We noticed the recommendations produced by NCN all have common authors[5]. Recommendations 1 and 3 contain M. Jordan as an author and

[4]Performance is less than reported in [49] due to significantly reduced vocabulary size.

[5]Authors omitted due to space constraints.

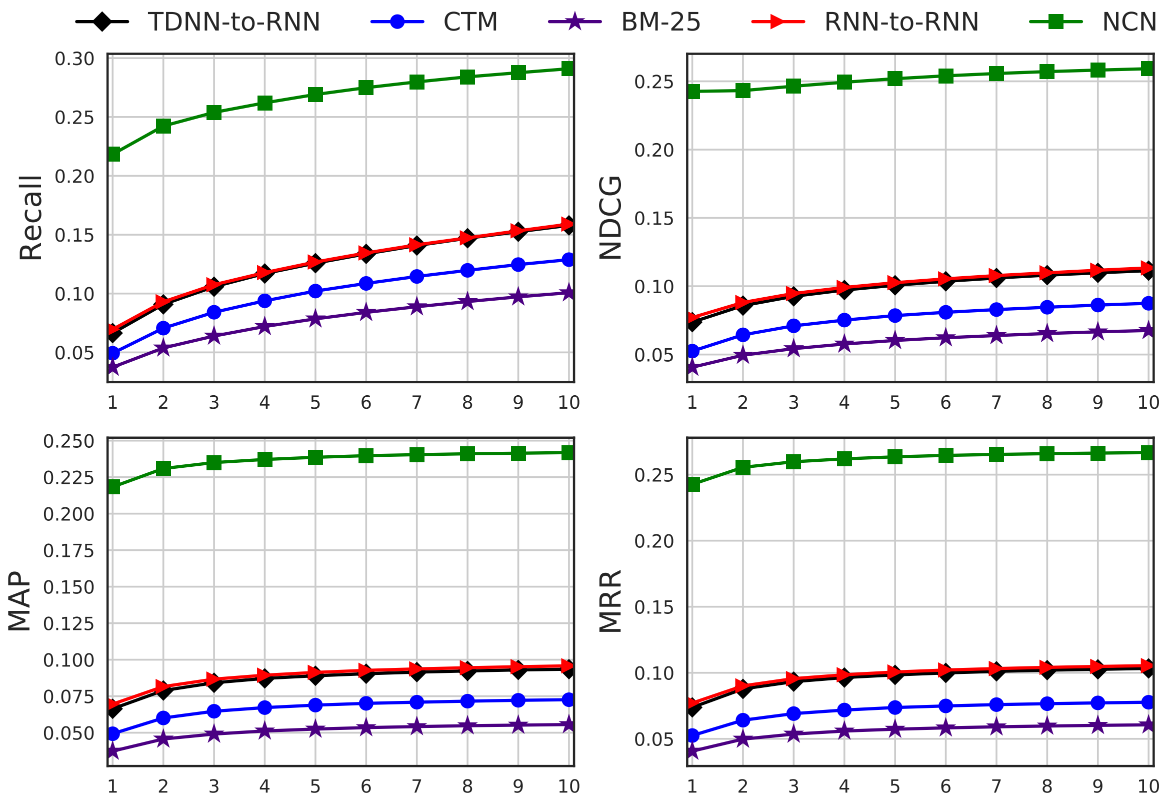


FIGURE 6.3: Recall, NDCG, MAP, and MRR as the number of recommendations vary from 1 to 10.

recommendations 2 and 3 shares the author Z. Ghahramani further portraying NCNs successful integration of author information to produce relevant recommendations.

## 6.4 Summary

We have introduced NCN, a flexible architecture capable of incorporating author metadata and highlight a promising new direction for context-aware citation recommendation. In future work, we plan to explore temporal aspects, and the large hyperparameters

| **Context:** "find a distribution over the latent variables that is close to the posterior of interest. Variational methods provide effective approximations in topic models and nonparametric Bayesian models" |
|---|
| **Neural Citation Network** |
| 1. **Graphical models, exponential families, and variational inference**<br>2. Graphical models and variational methods<br>3. **An introduction to variational methods for graphical models** |
| **CTM** |
| 1. Indexing by latent semantic analysis<br>2. **An introduction to variational methods for graphical models**<br>3. Bayesian data analysis |
| **RNN-to-RNN** |
| 1. **An introduction to variational methods for graphical models**<br>2. The variational formulation of the Fokker–Planck equation<br>3. A Bayesian analysis of the multinomial probit model with fully identified parameters |

Table 6.2: Top 3 recommendations for NCN, CTM and RNN-to-RNN for the citation context (query), correct recommendations are in bold.

space such as filter strides, wide convolutions, dynamic $k$-max pooling and multi-channel convolutions.

# Chapter 7

# Conversational Recommendation

## 7.1 Introduction

Recently, virtual assistants such as Alexa, Google Home or Siri are becoming an increasingly common part of many user's daily routines. Natural language serves as a convenient interface to computing systems and is a natural form to communicate our preferences with others. Hence, conversations serve as an excellent interactive medium to provide recommendations.

We consider the setting of conversational recommendations where two parties are interacting with one another centered around movies. The first party expresses his/her movie preferences and asks for relevant movie suggestions from the second party who acts as

| | |
|---|---|
| Seeker: | Hi |
| Recommender: | Hello |
| Seeker: | How are you? I would like to watch an animated movie. Maybe something like **Monsters, Inc.** |
| Recommender: | Oh! I love animated movies. Have you seen **Toy Story**, That is good, but equally as good in that series is **Toy Story 3** I also loved watching **Ice Age** Or **Shrek** .... |
| Seeker: | .... **Ice Age** and **Shrek** are such funny movies! |
| Recommender: | Those were really good movies. Have you seen **Coco** ? THat's a new movie you might like. |
| Seeker: | No! I haven't yet. I've heard about it but never got a chance to watch it .... |

FIGURE 7.1: Example dialogue between the Seeker asking for recommendations and the Recommender providing suggestions from the ReDial dataset. The movies mentions are in bold.

the recommender. This recommender's goal is to understand the user and provide personalized movie recommendations based on the conversation. An example dialogue is shown in Figure 7.1.

Previous work on conversational recommendations used synthetic data [20, 108] or assumed a entity tagger was present [72] which may not be available in practice. To the best of our knowledge, this work is the first to address conversational recommendations in a setting where no explicit feedback or tagged entities exist. We rely solely on extracting user preferences from natural language text simultaneously addressing the cold-start problem using transfer learning.

We tackle performing movie recommendations in a natural chit-chat conversational setting and propose a novel framework to address the cold-start problem in the conversational setting through transfer learning [89]. Specifically, we transfer the learned item preferences from a large dataset (e.g. MovieLens) to the smaller conversational domain dataset. For each conversation we learn a new user latent factor in an online fashion which is updated throughout the conversation. To facilitate learning a new user latent factor we extract user preferences directly from natural language text using transfer learning and construct a mixture of item latent factors. Our approach does not rely on a mapping between natural language text to unique movie identifiers. We focus solely on improving the recommendation engine which can be easily integrated as a submodule to existing dialogue system such as in Li et al. [72].

## 7.2 Recommendations through Conversational Transfer

In this section, we describe a generic framework to perform movie recommendations in a conversational chit-chat setting. We assume no entities are tagged in the conversation (i.e. movies names are not mapped to items). While having this information would be useful it poses its own set of challenges such as word sense and entity disambiguation, hence we leave to future work. Our setting poses several challenges such as the cold start problem where no past user interaction history exists, limited data, and extracting user preferences from natural language.

We solve the aforementioned problems using two different transfer learning approaches and learn a new user latent factor in an online setting by inferring user preference from natural language conversations. The proposed framework consists of three components.

The first is a pretrained interaction function which maps a user latent factor and item latent factor to a shared space measuring the user's affinity for a given item (e.g. matrix factorization). The second component is a pretrained encoder function that maps variable length sequences of text to a single fixed length representation in semantic space. The final component instantiates a mixture of the existing pretrained item latent factors based on the natural language conversation allowing the optimization of the loss function, hence propagating the gradient to update the user latent factor.

### 7.2.1 Interaction Function

First we require an interaction function to measure the user's level of interest in a given item. The objective is to estimate a user's preferences given a ratings matrix $\mathbf{R} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$ where $\mathcal{U}$ denotes the entire set of users and similarly, $\mathcal{I}$ the entire set of items. Each user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$ are represented in a shared $d$ dimensional space, denoted as $\mathbf{m}_u \in \mathbf{M}$ and $\mathbf{e}_i \in \mathbf{E}$, respectively where the entire set of all user latent factors is $\mathbf{M} \in \mathbb{R}^{|\mathcal{U}| \times d}$ and all item latent factors as $\mathbf{E} \in \mathbb{R}^{|\mathcal{I}| \times d}$. We require an interaction function to measure the user's level of interest or lack of to the given item. Formally, the ranking score for a given user $u$ and item $i$ is

$$\hat{r}_{ui} = f(\mathbf{m}_u, \mathbf{e}_i) \tag{7.1}$$

The interaction function $f(\cdot)$ can be a simple linear function such as the inner product yielding matrix factorization. More complex nonlinear functions may also be used instead such as deep neural networks. We assume the interaction function to already be trained on some large dataset (e.g. MovieLens). We defer the exact definition of $f(\cdot)$ to Section 7.2.6.

### 7.2.2 Encoder

Each utterance in the conversation at a given turn $t$ is denoted as $X_t = \{w_1, \ldots, w_{|X_t|}\}$, where $w_l$ indexes the $l^{th}$ word in the utterance and the entire conversation is denoted as $X_t \in \mathcal{X}$. Similarly, the $i^{th}$ movie's plot as $D_i = \{w_1, w_2, \ldots, w_{|D_i|}\}$. Next, we assume an encoder $\Phi(\cdot)$ which maps a variable length textual document (e.g. $X_t, D_i$) to a $v$ dimensional semantic vector. The encoding function is typically a pretrained language model and can be selected from any recent state of the art pretrained language models such as BERT [19], ELMo [95], Universal Sentence Encoder [10] or simpler neural bag of words methods such as Word2Vec [82] or Glove [93]. Note, once the encoder is pretrained on a large corpus the parameters remain frozen during our training process.

Our goal is to identify movies the user may be interested in based on their past utterances. This is done by computing the similarity of the given utterance and each movie's plot embedding. Specifically, each movie's plot $D_i$ is embedded with the encoding function $\mathbf{c}_i = \Phi(D_i)$. The set of encoded movies plots as a $|\mathcal{I}| \times v$ matrix $\mathbf{c}_i \in \mathbf{C}$.

Similarly, each utterance $X_t$ at turn $t$ is embedded to its corresponding semantic vector $\mathbf{x}_t = \Phi(X_t)$. Using the same encoder allows the movie's plot and each utterance to be mapped into the same space and hence semantically comparable. The similarity between the utterance $X_t$ and all movie embeddings are

$$\mathbf{q}_i^t = \mathbf{c}_i \mathbf{x}_t \quad \forall\, i \in \mathcal{I} \tag{7.2}$$

where $\mathbf{q}_i^t$ denotes the user's level of interest in the $i^{th}$ movie based on the encoded movie's summary and utterance as measured by the inner product.

### 7.2.3 Mixture of Item Latent Factors

Previously, we discussed how to obtain the user's interest vector $\mathbf{q}^t$. We now detail the instantiation of the mixture of item latent factors based on the user interaction vector extracted from the conversation. We construct this item latent factor as a weighted combination or mixture of existing item latent factors (preferences) from the dialogue. Thus the new user latent factors are learned directly from user feedback in the form of natural language and updated in an online fashion.

To obtain the instantiated item latent factor we first normalize the similarity with the softmax function

$$\mathbf{p}^t = softmax(\mathbf{q}^t) \tag{7.3}$$

where $\mathbf{p}^t$ represents the current user's preference distribution over each item from their utterance at turn $t$ and $softmax(\mathbf{x})_i = \exp(x_i)/\sum_j \exp(x_j)$. We can also interpret this as an attention mechanism, where the attention places higher weight on the most relevant movies. Finally, to obtain the mixture of item latent factors we perform a weighted linear combination of the item latent factors

$$\mathbf{z}^t = \sum_{j \in \mathcal{I}} p_j^t \mathbf{e}_j \tag{7.4}$$

We now have our estimated item latent factor extracted from the user utterance $X_t$ at turn $t$ and can instantiate our interaction function by substituting the item latent factor $\mathbf{e}_*$ with the estimated item latent factor $\mathbf{z}^t$ leading to

$$\hat{r}_u^t = f(\tilde{\mathbf{m}}_u, \mathbf{z}^t) \tag{7.5}$$

where $\tilde{\mathbf{m}}_u$ is a new randomly initialized user latent factor for this conversation. Intuitively, we estimate the ranking score of user $u$'s utterance $X_t$ at turn $t$ as a latent mixture of item preferences $\mathbf{z}^t$. In other words, we can view it as treating the user's utterance as an item where the plot consists of the user's utterance. Since we lack a corresponding item latent factor we extract a combination from the known item latent factors based on their content similarity.

### 7.2.4 Parameter Estimation

The nature of the data is similar to the implicit feedback setting therefore we take the pairwise assumption that a given user $u$ prefers the observed item $i^+$ over the unobserved item $i^-$. In our case, we assume the user prefers to discuss movies they enjoy. The intuition is as follows, user $u$ prefers movies related to the topics they converse with over topics they dislike where similarity is measured in the encoder's low dimensional embedding space. We opt for the commonly used Bayesian Personalized Ranking (BPR) [99] as our loss function. However, to optimize the BPR criterion, positive observed items and negative unobserved items are required. Hence with no interaction data we cannot use standard techniques to perform the sampling. Instead, the positive item is inferred from the conversation with the user as we detailed in the previous section instantiating $\mathbf{z}^t$ from utterance $X_t$. The interaction vector is truncated to the top-$k$ most similar movies according to Equation 7.2 thus $\mathbf{q}^t$ becomes a $k$ dimensional vector. Similarly, we randomly sample the negative items from the least similar top-$k$ movies with respect to the current utterance (i.e. the movies with the lowest similarity score according to Equation 7.2 are regarded as the negative items). Our training data consists of a tuple for each user utterance $X_t \in \mathcal{X}$ and sampled negative items $i^-$ then we minimize the following objective

$$\mathcal{L} = - \sum_{(X_t,\, i^-)}^{T} \log \sigma(\hat{r}_u^t - \hat{r}_{ui^-}) \tag{7.6}$$

The only parameter learned is the new user's latent factor $\tilde{\mathbf{m}}_u$ while all other parameters are held fixed. The loss function is optimized in an online fashion using stochastic gradient descent (SGD) which allows updating the user latent factor throughout the conversation. Note only the utterance requires the encoder function $\Phi(\cdot)$ during the online setting and the movie summaries can be encoded and cached ahead of time. See Algorithm 1 for the procedure on performing conversational recommendations.

### 7.2.5 Recommendation

The instantiated mixture of item latent factors do not directly correspond to a single item therefore cannot recommend items. Equation 7.5 is only used during the estimation of the new user $u$'s latent factor $\tilde{\mathbf{m}}_u$. The ranking score for the new user $u$ and item $i$

$$\hat{r}_{ui} = f(\tilde{\mathbf{m}}_u, \mathbf{e}_i) \tag{7.7}$$

Note that we use the newly learned user latent factor $\tilde{\mathbf{m}}_u$ and the true item latent factors $\mathbf{e}_i$ not the estimated mixture. The top-$n$ movies with the highest ranking scores are presented to the user.

### 7.2.6 Choice of Interaction Function

In this section, we define multiple forms of the interaction function $f(\cdot)$ to demonstrate the flexibility of our proposed framework. The first interaction function we define is a

**Algorithm 1:** Conversational Recommendation

```
Input: Conversation $\mathcal{X}$, Encoder $\Phi(\cdot)$, interaction function $f(\cdot)$, learning rate $\alpha$,
       pretrained item latent factors $\mathbf{E}$ and content matrix $\mathbf{C}$

Randomly initialize new user latent factor $\tilde{\mathbf{m}_u}$
for $X_t \in \mathcal{X}$ do
    if Speaker==Recommender then
        Recommend Movies (Equation 7.7)
    else Speaker == Seeker
        Encode utterance $\mathbf{x}_t \leftarrow \Phi(X_t)$
        Instantiate mixture of item latent factor $\mathbf{z}^t$ (Equation 7.4)
        Sample negative item $i^-$
        Update $\tilde{\mathbf{m}_u} \leftarrow \tilde{\mathbf{m}_u} + \alpha \nabla_{\tilde{\mathbf{m}_u}} \mathcal{L}(X_t, i^-)$
    end
end
```

linear version via the inner product yielding matrix factorization (MF).

$$f(\mathbf{m}_u, \mathbf{e}_i) = \mathbf{m}_u^\intercal \mathbf{e}_i = \sum_{j=1}^{d} \mathbf{m}_{uj} \mathbf{e}_{ij} \tag{7.8}$$

However, a linear function may lack the flexibility to disentangle complex user preferences and generalize to another dataset. Thus we use a nonlinear variant generalized matrix factorization (GMF) [39]

$$f(\mathbf{m}_u, \mathbf{e}_i) = \mathbf{v}^\intercal \phi(\mathbf{m}_u \odot \mathbf{e}_i) \tag{7.9}$$

where $\odot$ is the elementwise product; $\mathbf{v} \in \mathbb{R}^d$ is an additional parameters to be learned and $\phi(\cdot)$ is a nonlinear activation function. We adopt the rectified linear unit (ReLU) function $\phi(x) = \max(0, x)$ as our nonlinear function due to its nonsaturating behavior

| Dataset Statistics | |
|---|---|
| Conversations | 11,348 |
| Utterances | 206,102 |
| Average Dialogue Length | 18.16 |
| Items | 6,925 |

TABLE 7.1: Statistics of the ReDial dataset

[84]. GMF can also degenerate to matrix factorization if we set the nonlinear activation function $\phi(\cdot)$ to the identity function and constrain the vector $\mathbf{v}$ to the $\mathbb{1}$ vector of all ones.

## 7.3 Experimental Results

### 7.3.1 Dataset

We validate our proposed framework Recommendations through Conversational Transfer using the recommendations through dialogue (ReDial)[1] dataset [72] which consists of 11,38 dialogues where two users conversing about movies, see Table 7.1 for statistics. The dialogues are crowd sourced and collected from Amazon Mechanical Turk where two users are paired up to converse around the topic of movies. Each user plays a specific role. The first user known as the seeker tries to explain their movie preferences and asks for appropriate movie suggestions. The second user known as the recommender tries to understand the seeker and provide appropriate movie recommendations. When a movie is mentioned the users are asked to tag it and select the corresponding movie from a list

[1]`https://redialdata.github.io/website/`

sourced from DBPedia. This ensures the exact tagging of movies and disambiguation of movies with same name but released in different years. Additional information was collected from users separately from the discussion to validate the data such as who suggested the movie (seeker or recommender) and if the seeker Liked the movie or not. We only keep movie suggestions by the recommender which was not marked as dislike by the seeker. We use the provided training and testing splits released by the authors yielding 10,006 and 1,342 dialogues respectively. A separate held-out validation set of 1k examples from the training set is used for cross-validation.

Movie plots are collected from IMDB[2] and we use the latest version of MovieLens[3] consisting of 27M ratings from 283k users over 53k items for pretraining our interaction function. We split 90% of the ratings for training and 10% for testing purposes. Hyperparameters were not tuned for the pretrained interaction function. Following [72] we match up the movies between the ReDial dataset and the MovieLens dataset with simple text matching using the authors' provided implementation[4]. After preprocessing and removing invalid entries we obtain 5,053 movies.

### 7.3.2 Evaluation Metrics

We use standard recommendation system evaluation metrics for top-$n$ ranking, Normalized discounted cumulative gain (NDCG), Recall (R) and Mean reciprocal rank (MRR) [79]. Users are generally interested in only a few top-ranked movies, NDCG@$n$ and

[2]`https://www.imdb.com`
[3]`https://grouplens.org/datasets/movielens/latest/`
[4]`https://github.com/RaymondLi0/conversational-recommendations/`

MRR@$n$ are used to compare the top-$n$ recommendation performance. We use Recall as the data resembles an implicit feedback setting and measures the level of user feedback extracted from the conversation. Rank aware metrics such as NDCG and MRR alone may be insufficient since a negative entry could adversely impact the metric but in reality the user may not be aware of the item's existence. Specifically, we treat the movies mentioned by the recommender in utterance $X_t$ at turn $t$ as the ground truth where the seeker did not give the recommendation 'dislike'. Note the recommendation is performed prior to observing the utterance with the ground truth and the model has only seen the utterances prior to turn $t$. The reported metrics are averaged over all utterances in the test set.

### 7.3.3 Baselines and Settings

We validate the effectiveness of our model against two baseline methods.

- Random: Movies are uniformly at randomly presented to the user.
- Most Popular: The movies with the highest popularity that occur in the MovieLens training set are presented to the user.

We first pretrain our interaction function on the MovieLens dataset. The hyperparameters are as follows, we set the latent dimensions to be 16, apply an $L_2$ penalty of $1e-6$ and optimize the BPR criterion [99] using the stochastic gradient descent variant Adam

| | Random | Most Popular | MF | GMF |
|---|---|---|---|---|
| R@25 | 0.0053 | 0.0219 | 0.0312 | 0.0497 |
| R@50 | 0.0114 | 0.0503 | 0.0540 | 0.0862 |
| R@100 | 0.0234 | 0.0663 | 0.0908 | 0.1319 |
| NDCG@25 | 0.0016 | 0.0093 | 0.0114 | 0.0190 |
| NDCG@50 | 0.0028 | 0.0151 | 0.0161 | 0.0264 |
| NDCG@100 | 0.0048 | 0.0180 | 0.0223 | 0.0342 |
| MRR@25 | 0.0007 | 0.0064 | 0.0068 | 0.0120 |
| MRR@50 | 0.0009 | 0.0074 | 0.0076 | 0.0133 |
| MRR@100 | 0.0011 | 0.0077 | 0.0082 | 0.0140 |

Table 7.2: Experimental results for different methods reporting Recall (R), normalized discounted cumulative gain (NDCG) and mean reciprocal rank (MRR) at cut offs at 25, 50 and 100.

[60] with a standard learning rate of 0.001. Once we obtain the pretrained model (interaction function) the parameters are held fixed with the exception of each new user latent factor $\tilde{\mathbf{m}}_u$ which is randomly initialized from a Gaussian distribution with mean and variance computed from the existing pretrained user latent factors. This maintains the relative scale with respect to other parameters during training. During the conversation we use stochastic gradient descent with an initial learning rate of 0.1 and momentum of 0.9 to optimize the new user latent factor while all other parameters are held fixed. We set $k = 300$, truncating $\mathbf{q}^t$ to the top 300 most similar movies to the given utterance. At each turn $t$ we update the parameters $\tilde{\mathbf{m}}_u$ and randomly sample 10 negative items from the 300 least similar movies. For our encoder we use the Transformer [112] Universal Sentence Encoder [10] unless otherwise specified and in Section 7.3.5, we explore different variants.

| | R@25 | R@50 | R@100 | NDCG@25 | NDCG@50 | NDCG@100 |
|---|---|---|---|---|---|---|
| MF | 0.0312 | 0.0540 | 0.0908 | 0.0114 | 0.0161 | 0.0223 |
| MF-DAN | 0.0314 | 0.0577 | 0.0916 | 0.0117 | 0.0171 | 0.0230 |
| MF-ELMo | 0.0188 | 0.0319 | 0.0532 | 0.0064 | 0.0091 | 0.0127 |
| MF-BERT | 0.0272 | 0.0544 | 0.0877 | 0.0105 | 0.0161 | 0.0219 |
| GMF | 0.0497 | 0.0862 | 0.1319 | 0.0190 | 0.0264 | 0.0342 |
| GMF-DAN | 0.0407 | 0.0634 | 0.1079 | 0.0146 | 0.0192 | 0.0269 |
| GMF-ELMo | 0.0371 | 0.0608 | 0.1035 | 0.0134 | 0.0181 | 0.0255 |
| GMF-BERT | 0.0269 | 0.0465 | 0.0799 | 0.0099 | 0.0139 | 0.0197 |

Table 7.3: Experimental results for different encoder functions reporting Recall (R), normalized discounted cumulative gain (NDCG) at cut offs at 25, 50 and 100.

### 7.3.4 Baseline Comparison

Table 7.2 presents the results of the baseline methods against our proposed transfer learning approach using the two different interaction functions. Unsurprisingly the random baseline uniformly performs the worst. Matrix factorization (MF) narrowly outperforms the most popular baseline method demonstrating our framework does indeed provide some level of personalization. GMF performs the best overall which suggests the presence of more complex nonlinear interactions may be required to disentangle user preferences and transfer the knowledge to another dataset. Most notably the recall at higher cut offs 50 and 100 show larger increases than the baseline methods indicating the user preferences are being integrated but may have difficulty in ranking the relevant items further up in the list due to limited interaction data.

### 7.3.5 Effect of Encoder

To better understand the flexibility of the framework and its dependence on the encoder we compare several state of the art language models including the deep averaging network (DAN) version of the universal sentence encoder [10], ELMo [95] and BERT [19]. The results for matrix factorization (MF) and generalized matrix factorization (GMF) are listed in Table 7.3 reporting Recall and NDCG at cut offs at 25, 50 and 100. MF using the DAN universal sentence encoder slightly outperforms the transformer variant, however, for GMF the opposite is true, the transformer version outperforms the DAN variant. A similar phenomenon is seen between BERT and ELMo. Each encoders appears to capture various semantic information in the conversation leading to extracting different user feedback provided to the interaction function. Overall, the universal sentence encoders produce better results regardless of its transformer or DAN model architecture. Since BERT is also a transformer, the performance differences in the encoders may be due to the differences in training methodology and datasets used for pretraining.

## 7.4 Summary

In this work, we tackle the challenge of performing movie recommendations in the conversational setting with a generic framework using transfer learning. Specifically, we transfer existing learned item preferences from the MovieLens dataset and apply it to the conversational domain. Since no user interaction history exists we infer user preferences from the natural language conversation and learn a new user latent factor in an online fashion. Experimental results with two different interaction functions confirm the benefits of our framework. In addition, we examine the effect between the two interaction functions and multiple state of the art language model encoders. This work is just an initial step towards a promising new direction. Additional performance may also be achieved by using more complex interaction functions which explicitly integrate content information or leverage fine-tuning existing encoder language models on a dataset similar to the ReDial conversational dataset. We have also noticed users often request for movies by names of famous directors or actors. Integrating this additional information will help improve the quality of the recommendations. Additional challenges exist such as resolving the user's sentiment towards a suggested movie, understanding the intricate relation between the encoder and interaction functions as well as dialogue tasks like state tracking.

# Chapter 8

# Conclusion

## 8.1 Summary

In this thesis, we explored five different approaches to applying deep learning techniques to recommender systems addressing top-$n$ recommendation in the traditional collaborative filtering (CF) setting; item-cold start CF scenario; CF with contextual attributes; context-aware citation recommendation; and conversational movie recommendation. The five approaches are summarized below.

The first approach leverages the latest advances in memory networks and neural attention mechanisms to enhance traditional CF methods with implicit feedback. We propose a novel hybrid architecture, Collaborative Memory Network (CMN), unifying the two classes of CF models capitalizing on the strengths of the global structure of latent factor model and local neighborhood-based structure in a nonlinear fashion yielding a unified nonlinear hybrid model. Comprehensive experiments on three different public datasets demonstrate significant performance gains over seven competitive baselines. Qualitative visualization of the attention weights provide insight into the model's recommendation process and suggest the presence of higher order interactions. The second model, Neural Semantic Personalized Ranking (NSPR), addresses the item cold-start problem by tightly integrating a deep neural network to learn effective feature representations from item content and the pairwise latent factor model. Comprehensive experiments on two real-world datasets demonstrated the effectiveness of NSPR to address unseen items leveraging the inferred item feature representations extracted from the content information. We then proposed a generic top-$n$ recommendation framework called Attentive Contextual Denoising Autoencoder (ACDA) to leverage arbitrary user and item contextual information via an attention mechanism. The context-driven attention mechanism encodes the attributes into the user's hidden representation producing user specific recommendations from the personalized context. ACDA demonstrates outperforming competitive baselines on multiple datasets on event and movie recommendation tasks. Next, we addressed context-aware citation recommendation with a flexible encoder-decoder architecture called Neural Citation Network (NCN) embodying a powerful max time delay neural network encoder and recurrent neural network decoder, augmented with an attention mechanism and author networks. NCN is capable of characterizing the semantic composition of citation contexts and corresponding cited documents by exploiting author relations capturing additional preferences such as writing style, grammatical structure, word usage and citation preference. Quantitative results on the large-scale CiteSeer dataset reveal NCN cultivates a significant improvement over competitive baselines. Lastly, we tackle the challenge of performing movie recommendations in the conversational setting with a generic framework using transfer learning. Specifically, we transfer existing learned item preferences from the MovieLens dataset and apply it to the conversational domain. Since no user interaction history exists we infer user preferences from the natural language conversation and learn a new user latent factor in an online fashion. Experimental results with two different interaction functions confirm the benefits of our framework. In addition, we examine the effect between the interaction functions and multiple state of the art language model encoders.

We describe a few key takeaways from this thesis. First, the integration of nonlinear user-item interactions improves upon the representations captured over linear models. These learned mapping from high-dimensional heterogeneous data into low-dimensional dense feature representations automatically capture more complex user-item interactions yielding better recommendation performance. In addition, selecting an appropriate deep learning architecture for the task at hand and type of data available induces a prior which can substantially reduce the amount of data or compute required. For instance, recurrent neural networks are excellent at modeling fine grain temporal sequences, however in some cases such granularity may not be desired. A more suitable architecture may be a max time delay convolutional neural network which captures the sequential nature of the data on a coarser level and significantly reduces the amount of computation time required due to parallelization. Finally, the attention mechanism demonstrates the ability to incorporate personalized preferences into hidden representations. In this thesis, we have demonstrated the flexibility and widespread applications of the attention mechanism to integrate user personalization in neighborhood representations, contextual attributes, and hidden state representations. Visualization of the attention weights provide some level of interpretability into the model's recommendation process.

## 8.2 Future Work

As countless extensions are possible to each of the five methodologies proposed in this thesis. Instead, we discuss longer term future directions and challenges in the application of deep learning to recommender systems. Each of these future directions may serve as a guideline for extending any of the work proposed in this thesis.

The data hungry nature of deep learning increases its susceptibility to overfitting. The high level of sparsity associated with user feedback further contributes to this problem. Learning multiple related tasks in parallel demonstrate the ability to reduce overfitting and increase generalization through inductive bias and shared representations known as multitask learning [9]. For example, consider the setting where the first task is movie recommendation and the secondary task is to classify the genre of the correspond movie. This learning of joint tasks together provides additional supervisory signals and inductive bias for the model to learn more robust and generalizable representation. Despite its appeal, multitask learning typically requires careful hand tuned design decisions such as what should be shared among tasks and where to share them. It is still unclear how to properly integrate appropriate deep learning architectures into multiple tasks with a proper objective function that considers the complex interactions between tasks.

An area closely related to multitask learning is transfer learning. The key idea behind transfer learning is to 'transfer' knowledge from a domain where large amounts of labeled data exists to a new target domain where less data may exist [89]. Humans are particularly good at generalizing and transferring existing knowledge to a new problem. Computer vision and natural language processing communities extensively use this technique to handle new domains with limited training data. Although such domain knowledge can be transferred about images and text, the transfer of existing user and item preferences is much more unexplored. Consider the initialization of a model with some level of prior knowledge (pretrained), this should yield a more suitable local minima and better performance than a model from random initialization.

Explainability is a fundamental challenge in recommender systems and machine learning in general. Governments, regulatory bodies and users want to know why a specific item was recommended. Currently, deep learning models are treated as a black box with limited insight into its decision making process. The ability to justify the model's recommendation process will instill confidence in both the users and businesses.

Finally, applying deep learning to recommender systems indeed improves performance over traditional linear methods but often at a heavy computational cost. A large factor in determining the computational requirements of the model depends upon the location of the user and item interactions. If the interaction occurs in the upper layers, the computed item representations can be cached for all users substantially reducing the computational requirements. On the other hand, if the interaction occurs in the lower layers little to no computation may be reused across users. For deep learning-based recommendation algorithms to be successfully deployed in real-world scenarios researchers must consider obtaining real-time inference and the ability to scale to millions of users and items.

# Bibliography

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. URL `https://arxiv.org/pdf/1607.06450v1.pdf`.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[3] Zeynep Batmaz, Ali Yurekli, Alper Bilge, and Cihan Kaleli. A review on deep learning for recommender systems: challenges and remedies. *Artificial Intelligence Review*, pages 1–37, 2018.

[4] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. Research-paper recommender systems: a literature survey. *IJDL*, 17(4):305–338, 2016. ISSN 1432-1300.

[5] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *journal of machine learning research*, 3(Feb):1137–1155, 2003.

[6] James Bennett and Stan Lanning. The netflix prize. In *SIGKDD Cup*, volume 2007, page 35, 2007.

[7] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed Huai hsin Chi. Latent cross: Making use of context in recurrent recommender systems. In *WSDM*, 2018.

[8] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, pages 177–186. Springer, 2010.

[9] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[10] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[11] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. Attentive collaborative filtering: Multimedia recommendation with feature-and item-level attention. In *SIGIR*, 2017.

[12] Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. Svdfeature: a toolkit for feature-based collaborative filtering. *JMLR*, 13(1): 3619–3622, 2012.

[13] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. Sequential recommendation with user memory networks. In *WSDM*, 2018.

[14] Chen Cheng, Haiqin Yang, Irwin King, and Michael R Lyu. Fused matrix factorization with geographical and social influence in location-based social networks. In *AAAI*, 2012.

[15] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *RecSys*, 2016.

[16] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

[17] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, 2008.

[18] Li Deng and Dong Yu. Deep learning: methods and applications. *Foundations and Trends in Signal Processing*, 7(3–4):197–387, 2014.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[20] Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. Evaluating prerequisite qualities for learning end-to-end dialog systems. In *ICLR*, 2016.

[21] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. Sequential user-based recurrent neural network recommendations. In *RecSys*, 2017.

[22] Rong Du, Zhiwen Yu, Tao Mei, Zhitao Wang, Zhu Wang, and Bin Guo. Predicting activity attendance in event-based social networks: Content, context and social influence. In *UbiComp*, 2014.

[23] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011.

[24] Gintare Karolina Dziugaite and Daniel M. Roy. Neural network matrix factorization. *CoRR*, abs/1511.06443, 2015. URL `http://arxiv.org/abs/1511.06443`.

[25] Travis Ebesu and Yi Fang. Neural citation network for context-aware citation recommendation. In *SIGIR*, 2017.

[26] Travis Ebesu and Yi Fang. Neural semantic personalized ranking for item cold-start recommendation. In *Information Retrieval Journal*, pages 109–131. Springer, 2017.

[27] Travis Ebesu, Bin Shen, and Yi Fang. Collaborative memory network for recommendation systems. In *SIGIR*, 2018.

[28] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. Bayesian personalized ranking for non-uniformly sampled items. *JMLR*, 18, 2012.

[29] Xue Geng, Hanwang Zhang, Jingwen Bian, and Tat-Seng Chua. Learning image and user features for recommendation in social networks. In *ICCV*, 2015.

[30] Kostadin Georgiev and Preslav Nakov. A non-iid framework for collaborative filtering with restricted boltzmann machines. In *ICML*, pages 1148–1156, 2013.

[31] Carlos A Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):13, 2016.

[32] Yuyun Gong and Qi Zhang. Hashtag recommendation using attention-based convolutional neural network. In *IJCAI*, 2016.

[33] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[34] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[35] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[36] Jing He, Jian-Yun Nie, Yang Lu, and Wayne Xin Zhao. Position-aligned translation model for citation recommendation. In *SPIRE*, 2012.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *CVPR*, 2015.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[39] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, 2017.

[40] Jon Herlocker, Joseph A Konstan, and John Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. In *Information Retrieval Journal*, pages 287–310. Springer, 2002.

[41] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *NIPS*, 2015.

[42] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *ICLR*, 2016.

[43] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29 (6):82–97, 2012.

[44] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[45] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272. IEEE, 2008.

[46] Haoran Huang, Qi Zhang, Yeyun Gong, and Xuanjing Huang. Hashtag recommendation using end-to-end memory networks with hierarchical attention. In *COLING*, 2016.

[47] Wenyi Huang, Saurabh Kataria, Cornelia Caragea, Prasenjit Mitra, C. Lee Giles, and Lior Rokach. Recommending citations: Translating papers into references. In *CIKM*, 2012.

[48] Wenyi Huang, Zhaohui Wu, Prasenjit Mitra, and C. Lee Giles. Refseer: A citation recommendation system. *IEEE/ACM Joint Conference on Digital Libraries*, pages 371–374, 2014.

[49] Wenyi Huang, Zhaohui Wu, Chen Liang, Prasenjit Mitra, and C. Lee Giles. A neural probabilistic model for context based citation recommendation. In *AAAI*, 2015.

[50] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[51] Dietmar Jannach and Malte Ludewig. When recurrent neural networks meet the neighborhood for session-based recommendation. In *RecSys*, 2017.

[52] Yogesh Jhamb and Yi Fang. A dual-perspective latent factor model for group-aware social event recommendation. *Information Processing & Management*, 53 (3), 2017.

[53] Yogesh Jhamb, Travis Ebesu, and Yi Fang. Attentive contextual denoising autoencoder for recommendation. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval*, 2018.

[54] How Jing and Alexander J. Smola. Neural survival recommender. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 515–524, New York, NY, USA, 2017. ACM.

[55] Santosh Kabbur, Xia Ning, and George Karypis. Fism: factored item similarity models for top-n recommender systems. In *SIGKDD*, 2013.

[56] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *ACL*, 2014.

[57] Houda Khrouf and Raphaël Troncy. Hybrid event recommendation using linked data and user diversity. In *RecSys*, 2013.

[58] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. Convolutional matrix factorization for document context-aware recommendation. In *RecSys*, 2016.

[59] Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, 2014.

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

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

[62] Yehuda Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *TKDD*, 4(1):1, 2010.

[63] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender systems handbook*, pages 77–118. Springer, 2015.

[64] Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42, 2009.

[65] Artus Krohn-Grimberghe, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. Multi-relational matrix factorization using bayesian personalized ranking for social network data. In *WSDM*, pages 173–182. ACM, 2012.

[66] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*, 2016.

[67] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[68] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521 (7553):436–444, 2015.

[69] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[70] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *NIPS*, 2014.

[71] Cheng Li, Xiaoxiao Guo, and Qiaozhu Mei. Deep memory networks for attitude identification. In *WSDM*, 2017.

[72] Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. Towards deep conversational recommendations. In *Advances in Neural Information Processing Systems 31 (NIPS 2018)*, 2018.

[73] Sheng Li, Jaya Kawale, and Yun Fu. Deep Collaborative Filtering via Marginalized Denoising Auto-encoder. In *CIKM*, pages 811–820. ACM, 2015. ISBN 978-1-4503-3794-6.

[74] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M Blei. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 59–66. ACM, 2016.

[75] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. In *IEEE Internet Computing*, 2003.

[76] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[77] Pasquale Lops, Marco Degemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, 2011.

[78] Yang Lu, Jing He, Dongdong Shan, and Hongfei Yan. Recommending citations with translation model. In *CIKM*, 2011.

[79] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval.* Cambridge university press Cambridge, 2008.

[80] Paolo Massa and Paolo Avesani. Trust-aware recommender systems. In *RecSys*, 2007.

[81] Peter McCullagh and John A Nelder. *Generalized linear models*, volume 37. CRC press, 1989.

[82] T Mikolov and J Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 2013.

[83] Andriy Mnih and Ruslan Salakhutdinov. Probabilistic matrix factorization. In *NIPS*, 2007.

[84] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[85] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[86] Xia Ning and George Karypis. Slim: Sparse linear methods for top-n recommender systems. In *ICDM*, 2011.

[87] Xia Ning, Christian Desrosiers, and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender Systems Handbook*, 2015.

[88] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *ICDM*, pages 502–511. IEEE, 2008.

[89] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. In *IEEE Transactions on knowledge and data engineering*. IEEE, 2009.

[90] Weike Pan and Li Chen. Gbpr: Group preference based bayesian personalized ranking for one-class collaborative filtering. In *IJCAI*, volume 13, pages 2691–2697, 2013.

[91] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The Adaptive Web*, pages 325–341. Springer, 2007.

[92] Wenjie Pei, Jie Yang, Zhu Sun, Jie Zhang, Alessandro Bozzon, and David M. J. Tax. Interacting attention-gated recurrent networks for recommendation. In *CIKM*, 2017.

[93] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[94] Maria S Pera and Yiu-Kai Ng. A group recommender for movies based on content similarity and popularity. *IPM*, 2013.

[95] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. Deep contextualized word representations. In *NAACL-HLT*, 2018.

[96] Steffen Rendle. Factorization machines. In *ICDM*, 2010.

[97] Steffen Rendle and Christoph Freudenthaler. Improving pairwise learning for item recommendation from implicit feedback. In *WSDM*, pages 273–282, New York, New York, USA, feb 2014. ACM Press. ISBN 9781450323512.

[98] Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM*, pages 81–90. ACM, 2010.

[99] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-thieme. BPR : Bayesian Personalized Ranking from Implicit Feedback. *UAI*, 2009.

[100] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

[101] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive Modeling*, 5(3):1, 1988.

[102] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML*, 2007.

[103] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *WWW*, 2015.

[104] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *RecSys*, 2017.

[105] Ajit P Singh and Geoffrey J Gordon. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 650–658. ACM, 2008.

[106] Brent Smith and Greg Linden. Two decades of recommender systems at amazon.com. *Ieee internet computing*, 21(3):12–18, 2017.

[107] Florian Strub and Mary Jeremie. Collaborative Filtering with Stacked Denoising AutoEncoders and Sparse Inputs. In *NIPS Workshop on Machine Learning for eCommerce*, Montreal, Canada, 2015.

[108] Alessandro Suglia, Claudio Greco, Pierpaolo Basile, Giovanni Semeraro, and Annalina Caputo. An automatic procedure for generating datasets for conversational recommender systems. In *CLEF (Working Notes)*, 2017.

[109] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *NIPS*, 2015.

[110] Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. Learning to recommend quotes for writing. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[111] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *NIPS*, pages 2643–2651, 2013.

[112] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[113] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *CVPR*, 2015.

[114] Maksims Volkovs, Guang Wei Yu, and Tomi Poutanen. Dropoutnet: Addressing cold start in recommender systems. In *NIPS*, 2017.

[115] Chong Wang and David M Blei. Collaborative topic modeling for recommending scientific articles. In *SIGKDD*, 2011.

[116] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *SIGKDD*, 2015.

[117] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *SIGIR*, 2017.

[118] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *ICLR*, 2015.

[119] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. Recurrent recommender networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 495–503, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4675-7.

[120] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*, 2016.

[121] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*, 2016.

[122] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *IJCAI*, 2017.

[123] Caiming Xiong, Stephen Merity, and Richard Socher. Dynamic memory networks for visual and textual question answering. In *ICML*, 2016.

[124] Kelvin Xu, Jimmy Ba, Jamie Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[125] Haochao Ying, Liang Chen, Yuwen Xiong, and Jian Wu. Collaborative Deep Ranking : a Hybrid Pair-wise Recommendation Algorithm with Implicit Feedback. *PAKDD*, 2016.

[126] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 353–362. ACM, 2016.

[127] Shuai Zhang, Lina Yao, and Xiwei Xu. Autosvd++: An efficient hybrid collaborative filtering model via contractive auto-encoders. In *SIGIR*, 2017.

[128] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52 (1):5, 2019.

[129] Wei Zhang, Jianyong Wang, and Wei Feng. Combining latent factor model with location features for event-based group recommendation. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 910–918. ACM, 2013.

[130] Weinan Zhang, Tianming Du, and Jun Wang. Deep learning over multi-field categorical data. In *European Conference on Information Retrieval*, pages 45–57. Springer, 2016.

[131] Lei Zheng, Vahid Noroozi, and Philip S. Yu. 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 '17, pages 425–434, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4675-7. doi: 10.1145/3018661.3018665. URL http://doi.acm.org/10.1145/3018661.3018665.

[132] Yin Zheng, Cailiang Liu, Bangsheng Tang, and Hanning Zhou. Neural autoregressive collaborative filtering for implicit feedback. In *DLRS*, 2016.

[133] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. What to do next: Modeling user behaviors by time-lstm. In *IJCAI*, 2017.