

Deep and Adversarial Knowledge Transfer in Recommendation

by

Guangneng Hu

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
in Computer Science and Engineering

June 2021, Hong Kong, China

Acknowledgments

Firstly, I give my full thanks to my supervisor Prof. Qiang Yang. I still remember the first meeting in Buenos Aires, Argentina during IJCAI 2015 conference. His kindness and openness support me for my five-year study at HKUST, who helped me get the prestigious HKPFS scholarship and nominated me as the AAAI workflow co-chair.

I would like to thank my co-supervisor Prof. Lei Chen and all other committee members ranging from my PQE and thesis proposal to my thesis defense. Their critical thinking and comments make this thesis better.

I deeply thank Dr. Yu Zhang who made insightfully discussions and helped improve my writing style. I also deeply thank Prof. Xinyu Dai who helped me start my research study at Nanjing University and then provide continuing support when I pursue my PhD at HKUST.

I am happy to thank my roommates, friends, group members as well as many other folks who shared time with me and made a life better for me to enjoy.

Finally, I thank my parents and my elder sister. Without their support and understanding, I can not get to this point so far. I dedicate my thesis to them.

Contents

Title Page	i
Authorization Page	ii
Signature Page	iii
Acknowledgments	iv
Table of Contents	v
List of Figures	ix
List of Tables	xiii
Abstract	xv
1 Introduction	1
1.1 Motivation	2
1.2 Problem Description	7
1.2.1 Notation	8
1.2.2 Base Network	9
1.2.3 Transfer Unit	9
1.3 Research Challenges	10
1.4 Main Contributions	13
1.5 Thesis Organization	17
2 Background	20
2.1 Recommendation Techniques	20
2.1.1 Matrix Factorization	20
2.1.2 Deep Learning	21

2.2	Transfer Learning Approaches	22
2.2.1	Model-based Knowledge Transfer	22
2.2.2	Instance-based Knowledge Transfer	24
2.2.3	Feature-based Knowledge Transfer	24
2.2.4	Framework of deep transfer learning: An Anatomy	26
2.3	Adversarial Learning	26
3	Deep Model-based Knowledge Transfer in Recommendation	29
3.1	Introduction	29
3.2	Problem Description and Challenges	31
3.2.1	Notation	31
3.2.2	Base Network	32
3.2.3	Cross-stitch Networks	33
3.3	Methodology	34
3.3.1	Collaborative Cross Networks	34
3.3.2	Objective and Optimization	38
3.3.3	Complexity Analysis	39
3.4	Experiments	40
3.4.1	Data Sets and Evaluation Protocol	40
3.4.2	Baselines	42
3.4.3	Results	43
3.4.4	Analyses	48
3.5	Related Work	51
3.6	Conclusions	52
4	Deep Instance-based Knowledge Transfer in Recommendation	54
4.1	Introduction	54
4.2	Problem Description	56
4.2.1	Problem Formulation	56
4.2.2	A Basic Neural CF Network	57
4.3	Methodology	59
4.3.1	TransNet	59
4.3.2	Objective and Optimization	62

4.3.3	Complexity Analysis	63
4.4	Experiments	63
4.4.1	Data Sets and Evaluation Protocol	64
4.4.2	Baselines	66
4.4.3	Results	67
4.4.4	Analyses	69
4.5	Related Work	70
4.6	Conclusions	72
5	Deep Feature-based Knowledge Transfer in Heterogeneous User-Interest News Recommendation	74
5.1	Introduction	74
5.2	Problem Description	77
5.2.1	Problem Formulation	77
5.2.2	A Content-based Neural CF Base Model	78
5.3	Methodology	79
5.3.1	TrNews	79
5.3.2	Objective and Optimization	82
5.3.3	Inference for Unseen Users	83
5.4	Experiments	84
5.4.1	Data Sets and Evaluation Protocols	84
5.4.2	Results: Comparing Different Recommenders	86
5.4.3	Results: Comparing Different Transfer Strategies	87
5.4.4	Analyses	88
5.5	Related Work	94
5.6	Conclusions	95
6	Adversarial Knowledge Transfer in Recommendation	98
6.1	Introduction	98
6.2	Problem Description	100
6.2.1	Problem Formulation	100
6.2.2	Recommender	101
6.3	Methodology	104

6.3.1	PrivNet	104
6.3.2	Objective and Optimization	106
6.3.3	Complexity Analysis	107
6.4	Experiments	107
6.4.1	Datasets and Evaluation Protocols	107
6.4.2	Baseline	110
6.4.3	Results: Comparing Recommendation Performance	111
6.4.4	Results: Comparing Privacy Protection	112
6.4.5	Analyses	113
6.5	Related Work	116
6.6	Conclusion	118
7	Conclusion and Future Work	119
7.1	Conclusion	119
7.2	Future Work	120
7.3	List of Publications	121
	Reference	123

List of Figures

1.1	Illustration of Recommender Engine (left part) and the Data Sources (right part).	2
1.2	An Example of Deep Learning with Small Data: Reality in the Age of Big Data. The English domain has accumulated lots of rating data while the Chinese domain is facing the rating data sparsity issue.	3
1.3	The Ideal Performance of Deep Knowledge Transfer: Achieving Benefits of Deep Learning Meeting Transfer Learning.	4
1.4	Word Clouds in Top Two Categories of Microsoft News Corpora. The word distributions in the two domains are very different.	6
1.5	Infer User’s Gender from Their Watchings: An investigation on the MovieLens dataset.	7
1.6	Two Typical Recommendation Scenarios: Cross-Dataset Recommendation (left part) and Cross-Domain Recommendation (right part).	10
1.7	Illustration of the Privacy-Preserving Transfer Learning Setting. The transfer learning based recommender exploits the knowledge from the source domain, while the attacker tries to recover user private attributes contained in the transferred knowledge from the latent representation used by the recommender also.	12
1.8	Overview of Our Technical Contributions. We answer “what to transfer” in deep knowledge transfer for recommendation via the three proposed methods which are (a) model-based transfer, (b) instance-based transfer, and (c) feature-based transfer, respectively.	14
1.9	Adversarial knowledge transfer. It has three parties (source party, target party, and attacker). The privacy attacker infers user privacy from the transferred representations. Our proposed PrivNet method exploits the knowledge from the source domain with regularization from the adversary loss of the attacker via modelling them in an adversarial learning game.	18

1.10	The Organization of Thesis.	19
2.1	The Architecture of a Typical Neural Collaborative Filtering with Three Hidden Layers.	21
2.2	Three Model-based Transfer Learning Paradigms. (a) Hard Sharing: sharing hidden layers directly while keeping the task-specific layers. Soft sharing: No directly sharing of parameters and (b) using regularization terms to encourage parameters to be similar, and (c) modeling shared parameters by learning a linear combination of input activation maps from two networks.	23
2.3	Two Instance-based Transfer Learning Methods. (a) The method selects source instances by the model parameters itself (i.e., the instance weight is part of the parameter of the recommender model) and (b) the method selects source instances by the nearest neighbors of target examples found in the shared feature space.	25
2.4	Three Feature-based Transfer Learning Paradigms. Example methods of each paradigm are shown in Table 5.2. (a) The identity mapping equals to no adaptation. (b) The matrix symbol means the feature mapping is implemented by linear transfer. (c) The neural network icon mean the feature mapping is implemented by nonlinear transfer in a dilated way (i.e., the hidden layers have more number of neurons than the input/output layers.).	27
2.5	Framework of deep transfer learning: An Anatomy. It can be composed into two parts. The red part is the base network which is to model the individual domains (source domain and target domain). The blue part is the transfer unit which is to build the bridge across domains answering “what to transfer” (and “how to transfer”). This figure is adapted from [93].	28
3.1	The cross-stitch unit [83] (left) and the proposed cross connection unit (right). To enable knowledge transfer, the shared α_D in the cross-stitch network is a scalar while the shared H in the proposed collaborative cross network is a matrix.	34

3.2	The proposed collaborative cross networks (CoNet) architecture (a version of three hidden layers and two cross units). We adopt a multilayer FFNN as the base network (grey or blue part, see Sec. 4.2.2). The red dotted lines indicate the cross connections which enable the dual knowledge transfer across domains. A cross unit is illustrated in the dotted rectangle box (see Fig. 3.1b).	37
3.3	Impact of the sparsity on the Mobile dataset.	46
3.4	Impact of the sparsity on the Amazon dataset.	47
3.5	Sensitivity and Optimization	50
3.6	Ratio of zero entries	50
4.1	Architecture of the Proposed TransNet Model. The instance transfer component (rendered in green color) infers a variable-length transfer weight vector $\alpha = [\alpha_1, \dots, \alpha_s]$ based on the current target item i and source items $[j]^u = [j_1, \dots, j_s]$. A transfer vector c_{ui} is then computed as the weighted average, according to the weight vector α , over the source items.	61
4.2	The Missed Hit Users Distribution (not normalized) Over the Number of Training Examples on the Amazon (Top) Datasets and on the Mobile (Bottom).	71
5.1	Word clouds of two news corpora. Top: Cheetah Mobile data. Bottom: MIND data. Left and right parts represent different domains (categories).	76
5.2	Architecture of TrNews. There is a base network for each of the two domains. The shaded gray area in the target network is empty for unseen users. The translator enables knowledge transfer between source and target networks.	80
5.3	Translator. The translator enables knowledge transfer between source and target networks.	82
5.4	Impact of percentage (90%, 70%, 50%, 30%.) of shared users used to train the translator.	89
5.5	Impact of the history length (left) and embedding size (right).	92
5.6	Performance (right Y-axis in red cross) and loss (left Y-axis in blue circle) varying with training iterations.	93
6.1	t-SNE projection of transferred representations of users with (left) and without (right) training of PrivNet on the MovieLens-Gender dataset. (see Section 6.4.5 for details)	99

6.2	Architecture of PrivNet (a version of three layers). It has two components: the recommender and privacy attacker. The recommender (the left & right parts, see Section 6.2.2) is a representation-based transfer learning model where the red arrows indicate the representations transferred from the source domain to the target domain in a multilayer way. The privacy attacker (the middle part, see Section 6.3.1) marked by an avatar infers user privacy from the transferred representations. PrivNet (see Section 6.3.1) exploits the knowledge from the source domain with regularization from the adversary loss of the attacker indicated by the dotted box.	102
6.3	Impact of privacy component and public users. (FS-G: Foursquare-Gender, ML-G: MovieLens-Gender, ML-A: MovieLens-Age)	114
6.4	Parameter sensitivity for recommendation.	115
7.1	Summary of Our Deep and Adversarial Transfer Learning Models in Recommendation.	120

List of Tables

1.1	Notations and meanings.	8
2.1	Different Strategies in Feature-based Transfer Learning.	26
3.1	Datasets and Statistics.	40
3.2	Categorization of Baselines.	42
3.3	Comparison results of different methods on two datasets. The best results are boldfaced and the best baselines are marked with stars.	43
3.4	The performance when reducing training examples. Results with stars are inferior to MLP.	48
3.5	Performance Varying with Depth of Neural Models on Cheetah.	49
3.6	Performance Varying with Depth of Neural Models on Amazon.	49
4.1	Model Parameters of TransNet.	58
4.2	Datasets and Statistics.	64
4.3	Categorization of Baselines	66
4.4	Results on Amazon data. The last row is the relative improvement of TransNet over the best baseline.	68
4.5	Results on Mobile data. The last row is the relative improvement of TransNet over the best baseline.	68
5.1	Statistics of the datasets.	85
5.2	Different transfer learning strategies.	87
5.3	Impact of sharing word embeddings between source and target domains.	90
5.4	Training TrNews with alternating (Alter.) vs separating (Sepa.) strategies.	90
5.5	Training TrNews in two-stage vs end-to-end.	91
5.6	Example I: Some articles matter more while some are negligible. (No. 10 is the candidate news)	94

5.7	Comparison of different recommenders.	96
5.8	Comparison of different transfer strategies.	97
6.1	Statistics of datasets. (G=Gender, A=Age)	108
6.2	Setting of hyperparameters.	109
6.3	Categorization of comparing methods.	110
6.4	Comparison results of different methods on recommendation performance. The bold face indicates the best result while the star mark indicates the second best.	111
6.5	Comparison results on privacy protection. The bold face indicates the best result (the lower the better).	112
6.6	Necessity of adversary loss to regularize the recommender (lower value better privacy protection).	113
6.7	Example: Capturing short-/long-term user interests and high-level category relationship among items. No. 10 is the candidate new articles to be recommended while No. 0 to No. 9 are the historical news articles. Attn weight: Attention weight.	116

Deep and Adversarial Knowledge Transfer in Recommendation

by

Guangneng Hu

Department of Computer Science and Engineering, HKUST

Hong Kong University of Science and Technology

ABSTRACT

Recommendation is a basic service to filter information and to guide users from a large pool of items at various online systems, achieving both improved user satisfaction and increased corporate revenues. It works by learning user preferences on items from their historical interactions. Recent deep learning techniques bring in advancements of recommender models with the ability of learning representations of users and items from the interaction data. In real-world scenarios, however, interactions may well be sparse in a target domain of interest, and thus it hurts the huge success of deep models which are depending on large-scale labeled data. Transfer learning is studied to address the data sparsity by transferring the knowledge from auxiliary source domains.

There is a privacy concern when the source domain shares their data with the target domain. This issue gets worse by the ever-increasing abuses of personal data and it is inevitable due to the enforcement of data protection regulations. Existing research work focuses on improving the recommendation performance while ignores the privacy leakage issue.

In this thesis, we investigate deep knowledge transfer in recommendation, of that the core idea is to answer what to transfer between domains. Specifically, we propose three models in different transfer learning approaches, i.e., deep model-based transfer (DMT), deep instance-based transfer (DIT), and deep feature-based transfer (DFT). Firstly, in DMT, we transfer parameters in lower layers and learn source and target networks in a multi-task way. The CoNet model is introduced to learn dual knowledge transfer across domains and is capable of selecting knowledge to transfer via the sparsity-induced regularization technique enforced on the transfer matrix. Secondly, in DIT, we transfer certain parts of instances in the source domain by adaptively re-weighting them to be used in the target domain. The TransNet model is introduced to learn an adaptive transfer vector to capture relations between the target item and source items. Next, in DFT, we

transfer a “good” feature representation that captures the invariant while reduces the difference between domains. The TrNews model is introduced to transfer heterogeneous user interests across domains and transfer item representations selectively. The proposed transfer models can be used for modeling both relational data (e.g., clicks), content data (e.g., news), and their combinations (hybrid data).

Finally, we investigate the adversarial knowledge transfer in recommendation to protect the private attributes in the source domain. Specifically, we propose the PrivNet model which improves the target performance as well as protects the source privacy, of that the core is to learn a privacy-aware neural representation. Through extensive experiments on real-world datasets, we validate the research on adversarial knowledge transfer. This thesis will also describe the research frontier and point out promising future work for investigation.

Chapter 1

Introduction

Recommender systems can satisfy user information need by providing personalized recommendation services and also increase corporate revenues by actively engaging with consumers. The core idea is to match user preferences with items properties by learning from their historical interactions. Deep learning is capable of learning representations of users and items, and thus it is widely used in recommendation. In real-world scenarios, however, interactions may well be sparse in the target domain of interest, and thus the data-hungry deep models face the challenges since they depend on large-scale labeled data. Fortunately, transfer learning has been studied to tackle the data sparsity issue by transferring knowledge from auxiliary source domains.

When the source data is fusing into the the target domain, the privacy concerns are raised. A malicious attacker can infer a user's private attributes (e.g., gender, age, and occupation) from their source ratings and recommendation results. Existing recommendation methods are mainly focusing on improving the target performance while ignoring the privacy protection during knowledge transfer.

In this thesis, we investigate deep knowledge transfer in recommendation, achieving both benefits from deep learning meeting transfer learning in recommender systems. The core idea is to answer what to transfer between domains based on the neural recommender networks. Comparing with traditional (shallow) knowledge transfer, the deep knowledge transfer tightly integrates the learning of knowledge transfer with the representation learning of users and items. This novel learning paradigm enables that the recommendation task learning couples with the knowledge transfer learning, and vice versa.

We also investigate adversarial knowledge transfer in recommendation, protecting the private attributes in the source domain while improving the target performance, of that the core is to learn a privacy-aware transferable representation. With thorough experiments on real-world datasets, we validate the research on adversarial-based privacy-preserving knowledge transfer.

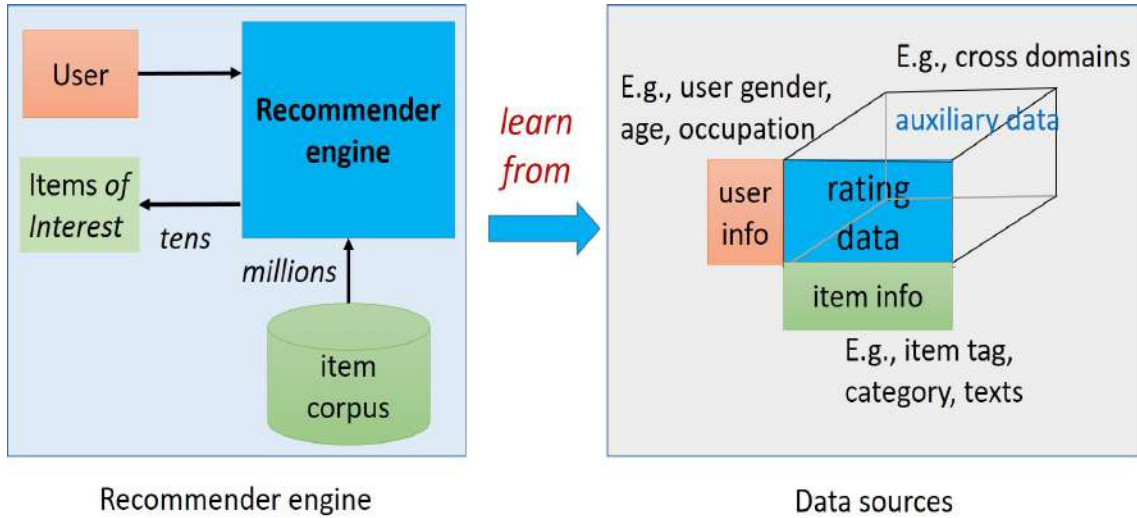


Figure 1.1: Illustration of Recommender Engine (left part) and the Data Sources (right part).

1.1 Motivation

Recommendation is a basic service at various online systems and industrial applications, including e-commerce websites of Amazon¹ and Taobao², news feeds platforms of Google News³ and Toutiao⁴. Recommender systems (RecSys) have been widely used for recommendations of products, articles, people, and ads, bringing in high revenues for companies and improving the customers' experiences [1]. As shown in Figure 1.1 (left part), the recommender engine can generate a small size of potential items to a user from millions of item corpus. RecSys achieves this service by learning users' preferences on items from their interactions, in the forms of user behaviors including clicks, ratings, purchases, watches, and installations.

With the emerging of deep learning and its great success in speech recognition, computer vision, and natural language processing [35, 109], deep learning is capable of learning representations of users and items, and thus it is widely used in recommendation [17]. Deep learning benefits from the big data age since the deep models are data hungry and need a lot of labeled examples, for example, the ImageNet dataset [19] and the Netflix grand prize [7] enabling the advancements in for computer vision and recommendation, respectively. One reason is that the number of parameters in deep models is much larger than the traditional learning algorithms such as matrix factorization techniques and support vector machines. Deep learning needs such large-scale data in order to learn a good representation and the overfitting is likely to occur when

¹<https://www.amazon.com/>

²<https://www.taobao.com/>

³<https://news.google.com/>

⁴<https://www.toutiao.com/>



Figure 1.2: An Example of Deep Learning with Small Data: Reality in the Age of Big Data. The English domain has accumulated lots of rating data while the Chinese domain is facing the rating data sparsity issue.

the model is very big while the training data is much small.

In some cases, however, there are not enough interaction data to learn a good recommendation model, when new users come or items are newly released. In such scenery, RecSys is difficult to understand users' true preferences, i.e., suffering from the data sparsity issues. Collecting data is sometimes highly cost and very time-consuming. It hinders the progress of RecSys research because of the property of data hungry and the need of large labeled data. Note that, the deep learning is not a panacea even in the big data age [78]. In real-world scenarios, as shown in Figure 1.2, we have accumulated lots of rating data in one domain (the English domain), while our target domain of interest (the Chinese domain) is fresh and so the rating data is sparse.

Luckily, transfer learning has been studied to reduce the need of massive labeled data in one domain by transferring the knowledge from another domain which has sufficient training data [93]. This learning paradigm is applicable to recommendation since the target user-item interactions may be sparse but there are relevant data sources in auxiliary domains. These data sources may come from the user-side (location, gender, social relations), item-side (tag, category, description texts), and the interaction-side (another user-item interaction matrix) (see Figure 1.1, right part).

During the past decade, transfer learning has been studied to address the data sparsity

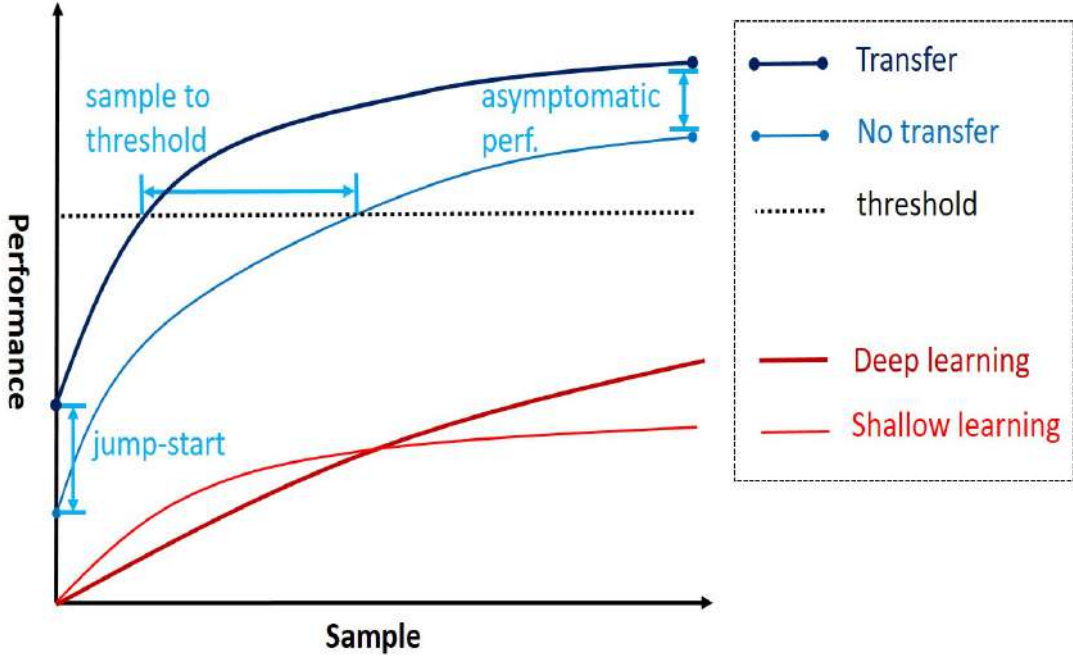


Figure 1.3: The Ideal Performance of Deep Knowledge Transfer: Achieving Benefits of Deep Learning Meeting Transfer Learning.

issue in recommendation and this multidisciplinary of the transfer in recommendation has been widely developed. The majority of existing transfer methods are built on the base of shallow recommendation models such as matrix factorization [61, 62, 95]. However, shallow methods are insufficient to learn complex relationships between users and items, or to extract useful features from semantic-rich content data. We aim to propose deep knowledge transfer for recommendation by benefiting from both deep learning with the ability of learning user and item representations and transfer learning with the ability of improving target performance by knowledge of source domains.

The motivation of deep knowledge transfer can be summarized in Figure 1.3. The figure is divided into two parts. The first part is about deep learning (in red) and the second part is about transfer learning (in blue). For the first part, the deep models surpass the shallow ones when a reasonable size of samples are fed to them since the deep learning can automatically extract features from the data rather than the human-involved feature engineering commonly used in shallow learning. For the second part, the transfer models are superior than the non-transfer ones in terms of three metrics: jump start, asymptomatic performance (abbreviated as asymptomatic perf. in the figure), and sample to threshold. The asymptomatic performance is the final performance of models in the target domain both with and without transfer learning. Since the training data is always finite in real-world recommender systems, a typical proxy is

the performance at the convergence point of the models. The jump-start performance is the initial performance in the target domain by comparing the transfer model with the non-transfer ones such as recommending random or the most popular items. When the single-domain SOTA (state-of-the-art) models are not able to work in cold-start recommendation due to the zero training examples in the target domain, the transfer model still works since it can exploit the knowledge from source domains. The sample-to-threshold is the number of training samples that the transfer model can reduce by comparing with non-transfer ones at achieving the same performance point. The more we can reduce, the better performance the transfer model is.

Deep knowledge transfer can be applied to enable knowledge transfer between many kinds of datasets in recommendation, ranging from cross-dataset (news feed and application installations), cross-domain (e-commerce Sports and Mens categories), to hybrid sources (online clicks and news content). This evolves relation data, content data, and hybrid data. We list a few deep knowledge transfer examples to be investigated in the following.

1. Cross-Dataset Deep Knowledge Transfer. In real-world life, a user typically participates several online systems to acquire different information services. For example, a user downloads applications in an app store (e.g., Apple Store, Google Play) and reads news from a website at the same time. Since users are overwhelmed in the million-scale news streaming⁵, it is necessary to filter out most of the news articles for individual users. This requires that the recommender system has made a precise user profile from their past history of reading news. When a user is just arriving at the system or their interactions are too few to model their user profile, the cold-start problem rises. It brings us an opportunity to improve the recommendation performance in the target service by learning across domains. Following the above example, we can transfer users' app installation feedback to benefit the learning of news recommendation. Naturally, we can also transfer users' reading news to benefit the learning of apps recommendation.

2. Cross-Domain Deep Knowledge Transfer. In E-commerce platforms, some category or domain is very popular and accumulates lots of rating reviews, say the Amazon Sports category. Some other domain is maybe newly deployed or it is a cold category, and hence there is few rating reviews for the users and items which are belonging to this domain, say the Amazon Men category. The recommender model can not reliably mine the user interests over items in the Amazon Men category, especially the recommender is a deep learning based model. Considering that the Men and Sports categories are both the review data in the same platform, the two domains

⁵<https://www.twingly.com/news-data>



(a) Watched by female only (total 50 movies).



(b) Watched by male only (total 350 movies).

Figure 1.5: Infer User’s Gender from Their Watchings: An investigation on the MovieLens dataset.

Masters (1982)), as shown in Figure 1.5. It implies that the occurrence of a rating, regardless of its numeric value (real or noisy), leaks the user privacy on revealing their genders. When these source rating examples are transferred to a target domain, the third party (the target domain or the malicious attacker) can infer the user privacy of the source domain from the rating examples, even though the source domain didn’t explicitly provide any user private attributes to the target domain. In this privacy-aware recommendation scenario, it is naturally to model the recommender and the attacker in an adversarial learning game.

1.2 Problem Description

Deep transfer learning aims to borrow knowledge in a source neural network into a target neural network so that the performance in the target domain can be improved. Since the neural architecture is usually similar between the source and target networks and the difference lies in learning parameters themselves, we first introduce a general, commonly used base network which can be used to instantiate them. Before that, we describe the notations and meanings used in this thesis.

Notation	Meaning
$\mathcal{S}, \mathcal{T}$	Source and target domains
$\mathcal{U}, \mathcal{I}_S, \mathcal{I}_T$	Set of users, source items, and target items
m, n_S, n_T	Size of users, source items, and target items
u, i	Indices for users ($u = 1, \dots, m$) and items ($i = 1, \dots, n$)
$\mathbf{R}_S, \mathbf{R}_T$	Interaction matrix of source and target domains
r_{uj}, r_{ui}	Ratings in source and target interaction matrix
$\mathbf{P}, \mathbf{Q}$	embedding matrices of users and items
d	dimensionality of embeddings
$\mathbf{W}, b$	weight and bias parameters of some neural network layer
$\mathbf{h}$	weight parameters of a logistic layer
λ	tradeoff hyperparameter between domains
$\mathbf{H}$	linear transformation matrix
$\Omega(\cdot)$	penalty functional on some parameter matrix
$\sigma(\cdot)$	some activation function
η	step size of gradient learning
L	Size of word vocabulary, or number of hidden layers
$[w_k]_{k=1}^l$	The word w_k sequence in some l -length document
$\mathbf{A}$	internal memory matrix
$\mathbf{m}_k$	The k -th memory slot
$[j_l]_{l=1}^s$	Rated item sequence with size s for some user
α_j, a_j	The normalized and unnormalized attention weight for item j
$\psi(\cdot), \phi(\cdot)$	The item content encoder, the user content encoder
$\mathcal{F}_{S \rightarrow T}$	The translator mapping from source to target representation
$\ \cdot\ _2^2$	The squared L_2 norm of some matrix
D_S, D_T	Training data set in source and target domains
$\mathbf{Y}^p \in \mathbb{R}^{m \times c_p}$	The p -th user private attribute (e.g., p ='Gender') and there are c_p choices
$\mathbf{Y} = \{\mathbf{Y}^p\}_{p=1}^n$	Denote all n private attributes data by (e.g., Gender, Age).

Table 1.1: Notations and meanings.

1.2.1 Notation

We list the commonly used notations in the thesis in Table 1.1. We are given two domains, a source domain $\mathcal{S}$ (e.g., news recommendation) and a target domain $\mathcal{T}$ (e.g., app recommendation). As a running example, we let app recommendation be the target domain and news recommendation be the source domain. The set of users in both domains are shared, denoted by $\mathcal{U}$ (of size $m = |\mathcal{U}|$). Denote the set of items in $\mathcal{S}$ and $\mathcal{T}$ by $\mathcal{I}_S$ and $\mathcal{I}_T$ (of size $n_S = |\mathcal{I}_S|$ and $n_T = |\mathcal{I}_T|$), respectively. Each domain is a problem of collaborative filtering for implicit feedback [50, 91]. For the target domain, let a binary matrix $\mathbf{R}_T \in \mathbb{R}^{m \times n_T}$ describe user-app installing interactions, where an entry $r_{ui} \in \{0, 1\}$ is 1 (observed entries) if user u has an interaction with app i and 0 (unobserved) otherwise. Similarly, for the source domain, let another binary matrix $\mathbf{R}_S \in \mathbb{R}^{m \times n_S}$ describe user-news reading interactions, where the entry $r_{uj} \in \{0, 1\}$ is 1

if user u has an interaction with news j and 0 otherwise. Usually the interaction matrix is very sparse since a user only consumed a very small subset of all items.

1.2.2 Base Network

A base network consists of four modules with the information flow from the input (u, i) (user-item pair) to the output $\hat{r}_{ui}$ (their matching score) as follows.

Input : $(u, i) \rightarrow \mathbf{x}_u, \mathbf{x}_i$. This module encodes user-item interaction indices. We adopt the one-hot encoding. It takes user u and item i , and maps them into one-hot encodings $\mathbf{x}_u \in \{0, 1\}^m$ and $\mathbf{x}_i \in \{0, 1\}^n$ where only the element corresponding to that index is 1 and all others are 0.

Embedding : $\mathbf{x}_u, \mathbf{x}_i \rightarrow \mathbf{x}_{ui}$. This module embeds one-hot encodings into continuous representations via two embedding matrices and then merges them as $\mathbf{x}_{ui} = [\mathbf{P}^T \mathbf{x}_u, \mathbf{Q}^T \mathbf{x}_i]$ to be the input of successive hidden layers.

Hidden layers: $\mathbf{x}_{ui} \rightsquigarrow \mathbf{z}_{ui}$. This module takes the continuous representations from the embedding module and then transforms, through multi-hop say L , to a final latent representation $\mathbf{z}_{ui} = \phi_L(\dots(\phi_1(\mathbf{x}_{ui})\dots)$. This module consists of multiple hidden layers to learn nonlinear interaction between users and items.

Output : $\mathbf{z}_{ui} \rightarrow \hat{r}_{ui}$. This module predicts the score $\hat{r}_{ui}$ for the given user-item pair based on the representation $\mathbf{z}_{ui}$ from the last layer of multi-hop module. Since we focus on one-class collaborative filtering, the output is the probability that the input pair is a positive interaction. This can be achieved by a softmax layer: $\hat{r}_{ui} = \phi_o(\mathbf{z}_{ui}) = \frac{1}{1 + \exp(-\mathbf{h}^T \mathbf{z}_{ui})}$, where $\mathbf{h}$ is the parameter of the logistic layer.

Note that, this kind of base network can be easily extended to modelling user info and item content for content-based/hybrid collaborative filtering. We just need to plug a content encoder enforced on input of user and item indices.

1.2.3 Transfer Unit

After defining the base network for source and target neural networks, the core step of deep transfer learning is to design the transfer unit between them to enable the knowledge transfer from the source domain to the target domain (or benefitting both tasks in multi-task learning). The design methodology of transfer unit can be guided by “what to transfer” between domains. We will follow it to design model-based, instance-based, and feature-based transfer units in this thesis. The model-based transfer unit can learn soft-sharing parameters via a linear combination

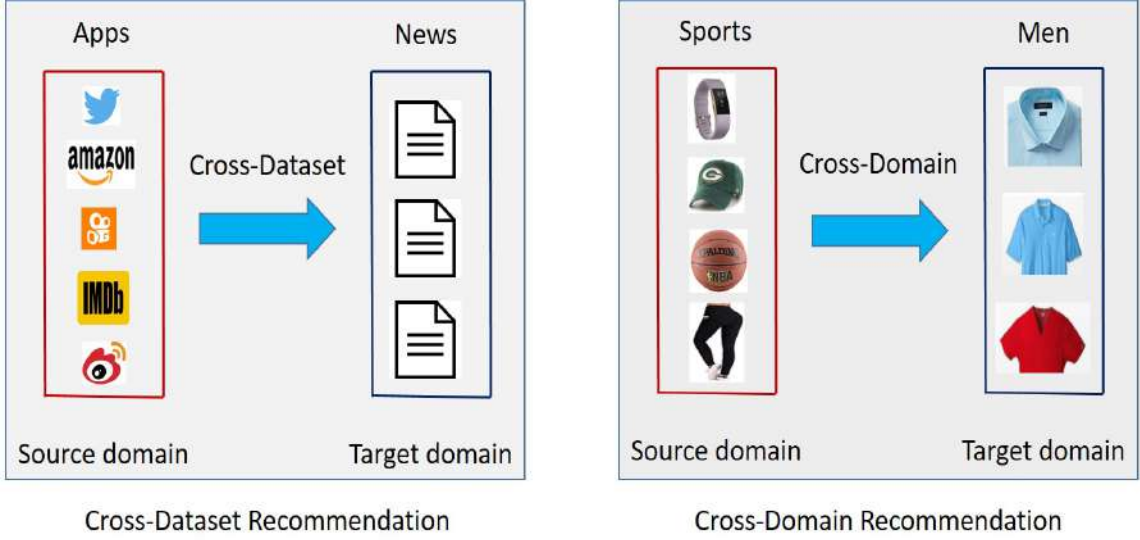


Figure 1.6: Two Typical Recommendation Scenarios: Cross-Dataset Recommendation (left part) and Cross-Domain Recommendation (right part).

between the source and target networks. The instance-based transfer unit can learn adaptive weights of source instances to be transferred into the target domain. The feature-based transfer unit can learn a mapping from source representation to the target space so that the cold-start recommendation can be resolved.

Together the source and target networks and the knowledge transfer unit enforced over them, the deep transfer learning problem is to optimize all of them to generate a good recommender system in the target domain.

1.3 Research Challenges

Deep knowledge transfer in recommendation is facing challenges on not only transfer learning but also deep transfer in recommendation problem. We list three research challenges in the following.

- Firstly, the deep transfer learning model must have the general ability of addressing cross-data, cross-domain, and hybrid sources recommendation scenarios. The key step of transfer learning is to build bridges between source and target domains so that the knowledge can be transferred from the source neural network to the target. These bridges can be guided by answering “what to transfer” and thus it can be in three kinds of manners: i) model-based transfer where part of parameters in source network are transferred to the target neural network, ii) instance-based transfer where part of examples in source domain

are transferred to the target domain, and iii) feature-based transfer where transferrable features are learned across the source and target domains. In real-world life, these transfer learning approaches need to match the recommendation scenario which is very diverse, as shown in Figure 1.6. It could be cross-dataset recommendation setting where the target dataset may come from a news reading platform while the source dataset is from an app store. It could also be cross-domain recommendation setting where the target domain may come from an e-commerce smaller category while the source domain is from a larger category.

- Secondly, the deep transfer learning model has to tackle both homogeneous and heterogeneous domains. The core of transfer learning is to construct a mapping from the source domain to the target. In real-world scenarios, the feature spaces of the two domains can be homogeneous. For example, we may transfer the knowledge from instances collected in old days to the target domain where the examples are coming in recent days. The representational structure is the same so that they can be directly operated and the shared knowledge is easily to identity to be transferred. In many other cases, we often face the heterogeneous transfer where the feature space is totally different (say the Apps and News domains as shown in Figure 1.6) or is much different (say the two new categories as shown in Figure 5.1). In light of the heterogeneity, the key supervised signal is the alignment between the two domains which is required to be annotated by humans. For example, regarding the cross-dataset recommendation as shown in Figure 1.6, we need to know that a subset of apps is corresponding to what part of the news. The supervised signal can be that the subset of apps and the part of news are both belonging to the same user, and then we can exploit such assignment to train the transfer learning algorithms. The possible weakly supervised signal is that the label⁸ of the source instances and the target instances is semantically the same or similar. For example, they both belong to the same topic “entertainment” in the Apps and News recommendation, or the category has hyponymy-hypernymy relationship (“clothes” vs “men clothes”) in the Amazon cross-domain recommendation.
- Finally, the deep transfer learning model is better to selectively transfer and to have effective neural architecture, so as to avoid the negative transfer issues. Usually we expect that the knowledge from the source is beneficial to improve the performance of the task

⁸The label space is the same between the source and the target domain in this case.

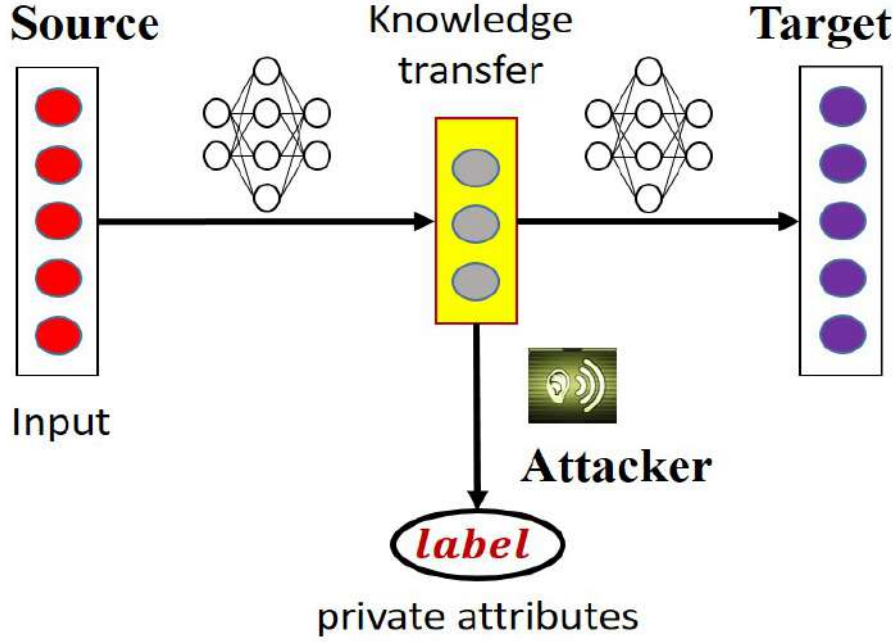


Figure 1.7: Illustration of the Privacy-Preserving Transfer Learning Setting. The transfer learning based recommender exploits the knowledge from the source domain, while the attacker tries to recover user private attributes contained in the transferred knowledge from the latent representation used by the recommender also.

in the target domain. The negative transfer issue happens when the knowledge transfer has a negative effect on the target task learning, i.e., hurting the performance in the target domain. Since the model complexity of deep learning is higher than that of shallow learning, the negative transfer is more complicated in deep transfer learning. It is better that the transfer model has the mechanism of selecting to transfer, including selecting parameters to transfer, selecting instances to transfer, and selecting features to transfer. The philosophy behind “selecting to transfer” is that many could be better than all.

The success of machine learning and deep learning algorithms is heavily relied on a big amount of labeled data to train a good good model. Though the aforementioned deep transfer learning can alleviate the issue of small data in the target domain by transferring the knowledge from a large source domain, the privacy concerns are raised when the source data is fusing into the target party. We describe the challenge that a privacy-preserving transfer learning algorithm faces in such scenario as shown in Figure 1.7.

- Firstly, a technical challenge for protecting user privacy in transfer learning is that the transferred knowledge has two-fold dilemma roles: its usefulness to improve target recommendation as well as its uselessness to infer source user privacy. Intuitively, it is beneficial

to improve the target domain performance when the transferable knowledge ‘carries’ the information of source domain since it can exploit the learned knowledge in the source domain. However, such information of source knowledge may unintentionally contain extra aspects of the source domain to recover the private attributes of the source user which is irrelevant to improve the target domain. As a result, a smart mechanism is needed to disentangle the transferable representations in terms of such dual roles.

- Secondly, another challenge is that the recommender in the target domain does not know the attackers and has no control over it during the test. The goal of the recommender model is to recommend ranked items to users such that any potential adversary cannot infer users’ private attributes (e.g., age, gender and occupation). However, a challenge is that the recommendation system does not know the malicious attacker’s model. The attacker can iteratively adapt its model regarding to existing recommender since the attacker can get the recommended results from the recommender (the recommended results are publicly visible to users) but not vice versa. In other words, the attacker is in the dark place which is not visible to the target recommender while the target recommender is in the light place which is visible to the attacker.

1.4 Main Contributions

In this thesis, we pursue to address the challenges as mentioned above via innovative research. With the guidance of answering “what to transfer”, we propose to investigate model-based, instance-based, and feature-based transfer learning algorithms, as shown in Figure 1.8, which are capable of addressing the aforementioned research challenges.

- In deep model-based transfer (DMT) (the left part in Figure 1.8), we transfer parameters in lower layers and learn source and target networks in a multi-task learning way. The CoNet model is proposed to learn dual knowledge transfer across domains and is capable of selecting knowledge to transfer via the sparsity-induced technique to avoid negative transfer. CoNet is effectively applicable to cross-dataset recommendation in heterogeneous transfer learning since it is a soft-sharing parameter transfer, i.e., learning a linear combination of lower-layer’s activation maps from the source and target networks. It is the task that supervises how much sharing is needed to learn. CoNet can decide to make certain layers

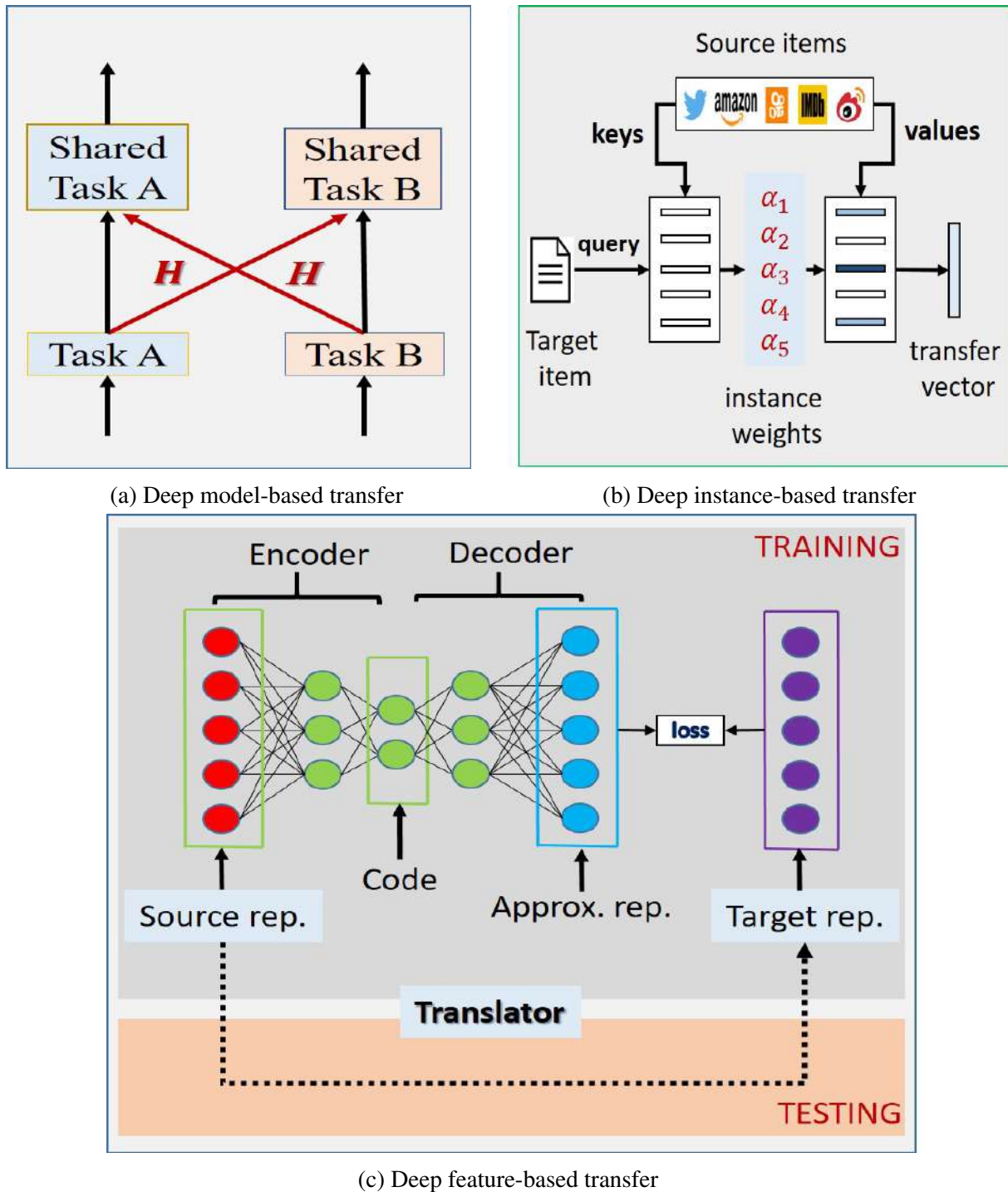


Figure 1.8: Overview of Our Technical Contributions. We answer “what to transfer” in deep knowledge transfer for recommendation via the three proposed methods which are (a) model-based transfer, (b) instance-based transfer, and (c) feature-based transfer, respectively.

more task specific or choose a more shared combination (i.e., more task general) via the learnable transfer matrix. In detail, the transfer unit in CoNet is implemented by:

$$\mathbf{a}_S^{(\ell+1)} = \sigma(\mathbf{W}_S \mathbf{a}_S^{(\ell)} + \mathbf{H} \mathbf{a}_T^{(\ell)}), \quad (1.1a)$$

$$\mathbf{a}_T^{(\ell+1)} = \sigma(\mathbf{W}_T \mathbf{a}_T^{(\ell)} + \mathbf{H} \mathbf{a}_S^{(\ell)}), \quad (1.1b)$$

where $\mathbf{a}_S^{(\ell)}$ and $\mathbf{a}_T^{(\ell)}$ are lower-layer activation maps while $\mathbf{a}_S^{(\ell+1)}$ and $\mathbf{a}_T^{(\ell+1)}$ are the corresponding higher-layer activations maps in source and target networks, $\mathbf{W}_S$ and $\mathbf{W}_T$ are weight matrices in source and target networks, and the transfer matrix $\mathbf{H}$ controls the information from source network to the target and vice versa. The knowledge transferring happens in two directions, from source to target and from target to source. When target domain data is sparse, the target network can still learn a good parameter from that of the source network through the transfer units. It only needs to learn “residual” target parameter with the reference of source parameter, making the target task learning easier.

- In deep instance-based transfer (DIT) (the middle part in Figure 1.8), we transfer certain parts of instances in the source domain by adaptively re-weighting them to be used in the target domain. The TransNet model is proposed to learn an adaptive transfer vector to capture relations between the target item and source items and is capable of selecting knowledge to transfer via the attention mechanism to avoid negative transfer. TransNet can decide to make certain instances more task specific or choose more shared instances (i.e., more task general) via the learnable instance-transfer weights. In detail, the transfer unit in TransNet is implemented by:

$$\mathbf{c}_{ui} = \sigma\left(\sum_j \alpha_j^{(i)} \mathbf{x}_j\right), \quad (1.2)$$

where $\mathbf{c}_{ui}$ is the achieved transfer vector over source item instances summarizing the knowledge from the source domain, $\mathbf{x}_j$ is the source instance, and α_j is its instance-weight instantiated by the normalized attention weight computed by the attention function $\text{Att}(\cdot, \cdot)$:

$$\alpha_j^{(i)} = \text{Att}(\mathbf{x}_i, \mathbf{x}_j), \quad (1.3a)$$

$$\alpha_j^{(i)} = \text{softmax}(\alpha_j^{(i)}). \quad (1.3b)$$

TransNet sharpens the idea that the transfer component can selectively transfer source items with the guidance of target user-item interactions. This is achieved by attentive

weights $\alpha_j^{(i)}$. When the source item j is highly relevant to the target item i given some user, then the knowledge from the source domain is easily flowing into the target domain with a high influence weight. When the source item j is irrelevant to the target item i given some user, then the knowledge from source domain is hard to flow into the target domain with a small effect weight.

- In deep feature-based transfer (DFT) (the right part in Figure 1.8), we transfer a “good” feature representation which captures the invariant while reduces the difference between domains. The TrNews model is proposed to transfer heterogeneous user interests across domains and transfer item representations selectively. TrNews can be used for modeling both relational data (e.g., clicks), content data (e.g., news), and their combinations (hybrid data). In detail, the transfer unit in TrNews is implemented by:

$$\mathcal{L}_{\mathcal{F}} = \frac{1}{|\mathcal{U}_0|} \sum_{u \in \mathcal{U}_0} \|\tilde{\phi}_S(u) - \phi_T(u)\|_2^2, \quad (1.4)$$

where $\mathcal{U}_0$ is the shared user set of source and target domains. The transfer unit takes a user’s source representation $\phi_S(u)$ as the input, and maps it to a hidden representation (called “code”) via an encoder parameterized by θ , and then gets a approximated representation $\tilde{\phi}_S(u)$ from the code via a decoder parameterized by θ' . The learned mapping function $\mathcal{F} = \{\theta, \theta'\}$ is then used for inferring representations of unseen users in the target domain. It fulfills knowledge transfer from the source domain to the target via a supervised learning process.

- In adversarial knowledge transfer (our model dubbed as PrivNet) (as shown in Figure 1.9), we learn such a transferable representation that it is beneficial to improve the target performance while it protects the user private attributes in the source domain. The PrivNet model achieves this goal by modelling the recommender and the attacker in an adversarial learning game. PrivNet can be easily extended in a plug-and-play way by replacing the recommender model (shallow or deep) and replacing the attacker model (linear or nonlinear).

The transfer unit in PrivNet is implemented by as follows. Let $\mathbf{x}_{u|\#}^\ell$ where $\# \in \{S, T\}$ be user u ’s source/target representation in the ℓ -th layer ($\ell = 1, 2, \dots, L - 1$) where $\mathbf{x}_{u|S}^1 = [\mathbf{x}_u, \mathbf{x}_j]$ and $\mathbf{x}_{u|T}^1 = [\mathbf{x}_u, \mathbf{x}_i]$. The transferred representation is computed by projecting the source representation to the space of target representations with a translation

matrix $\mathbf{H}^\ell$:

$$\mathbf{x}_{u|trans}^\ell = \mathbf{H}^\ell \mathbf{x}_{u|S}^\ell, \quad (1.5)$$

The attacker model in PrivNet predicts the private user attribute from their source representation sent to the target domain:

$$\hat{y}_{u,p} = P(y_{u,p} | \mathbf{x}_{u|trans}; \theta_p) = f_p(\mathbf{x}_{u|trans}; \theta_p), \quad (1.6)$$

where $\hat{y}_{u,p}$ is the predicted value of user u 's p -th private attribute and $p = 1, \dots, n$. f_p is the prediction model parameterized by θ_p . For all n private user attributes, the attacker model minimizes the multitask loss:

$$\mathcal{L}(\Theta) = -\frac{1}{n} \sum_p \sum_{D_p} \log P(y_{u,p} | \mathbf{x}_{u|trans}; \theta_p), \quad (1.7)$$

where $\Theta = \{\theta_p\}_{p=1}^n$ and D_p is training examples for the p -th attribute.

The adversarial learning game in PrivNet works as follows. The generator is a privacy attacker which tries to accurately infer the user privacy, while the discriminator is an recommender which learns user preferences and deceives the adversary. The recommender of PrivNet minimizes:

$$\tilde{\mathcal{L}}(\theta) = \mathcal{L}(\theta) - \lambda \mathcal{L}(\Theta), \quad (1.8)$$

where the hyperparameter λ controls the influence from the attacker component. PrivNet seeks to improve the recommendation quality (the first term on the right-hand side) and fools the adversary by maximizing the loss of the adversary (the second term,). PrivNet reduces to privacy-agnostic transfer model when $\lambda = 0$.

1.5 Thesis Organization

The thesis is organized into seven chapters as shown in Figure 1.10. In Chapter 2, we first survey recommendation methods w.r.t. shallow matrix factorization deep learning techniques, and transfer learning works w.r.t model-based transfer, instance-based transfer and feature-based transfer, and then we summarize some closely related works on adversarial learning. In chapters 3, 4, and 5, we respectively study each of the three different deep knowledge transfer problem settings in terms of model-based, instance-base, feature-based. In Chapter 6, we introduce our work on learning privacy-preserving representations via adversarial knowledge transfer.

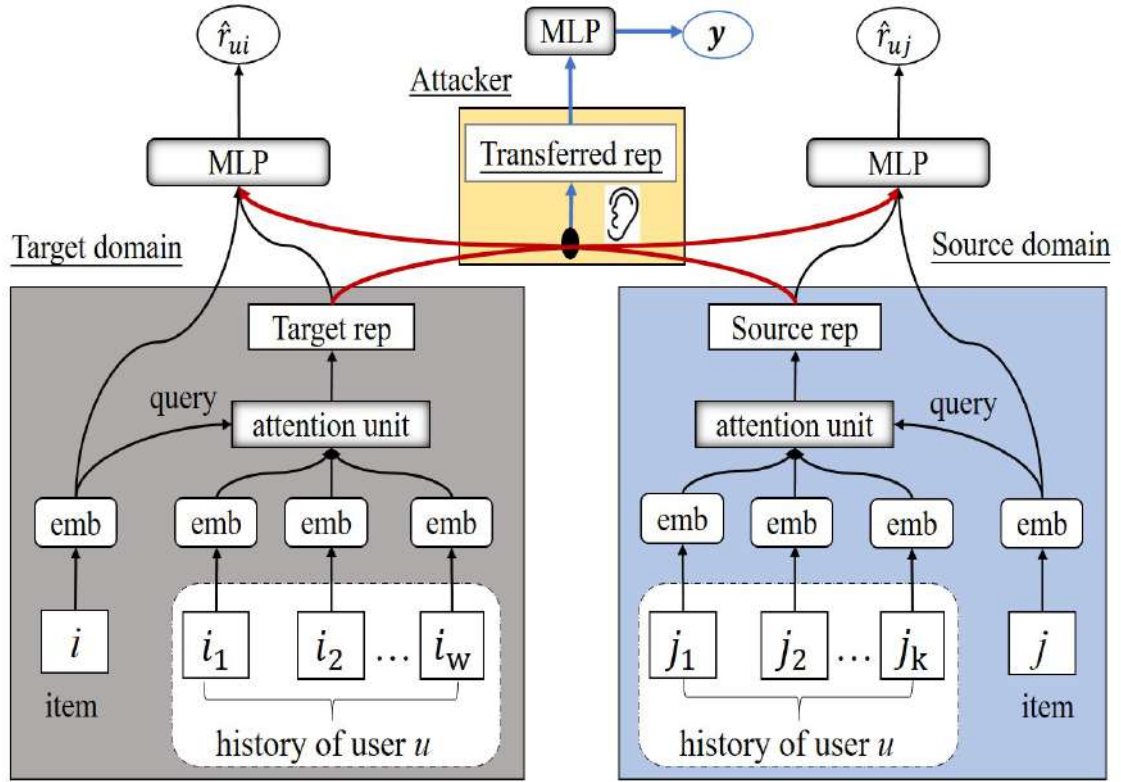


Figure 1.9: Adversarial knowledge transfer. It has three parties (source party, target party, and attacker). The privacy attacker infers user privacy from the transferred representations. Our proposed PrivNet method exploits the knowledge from the source domain with regularization from the adversary loss of the attacker via modelling them in an adversarial learning game.

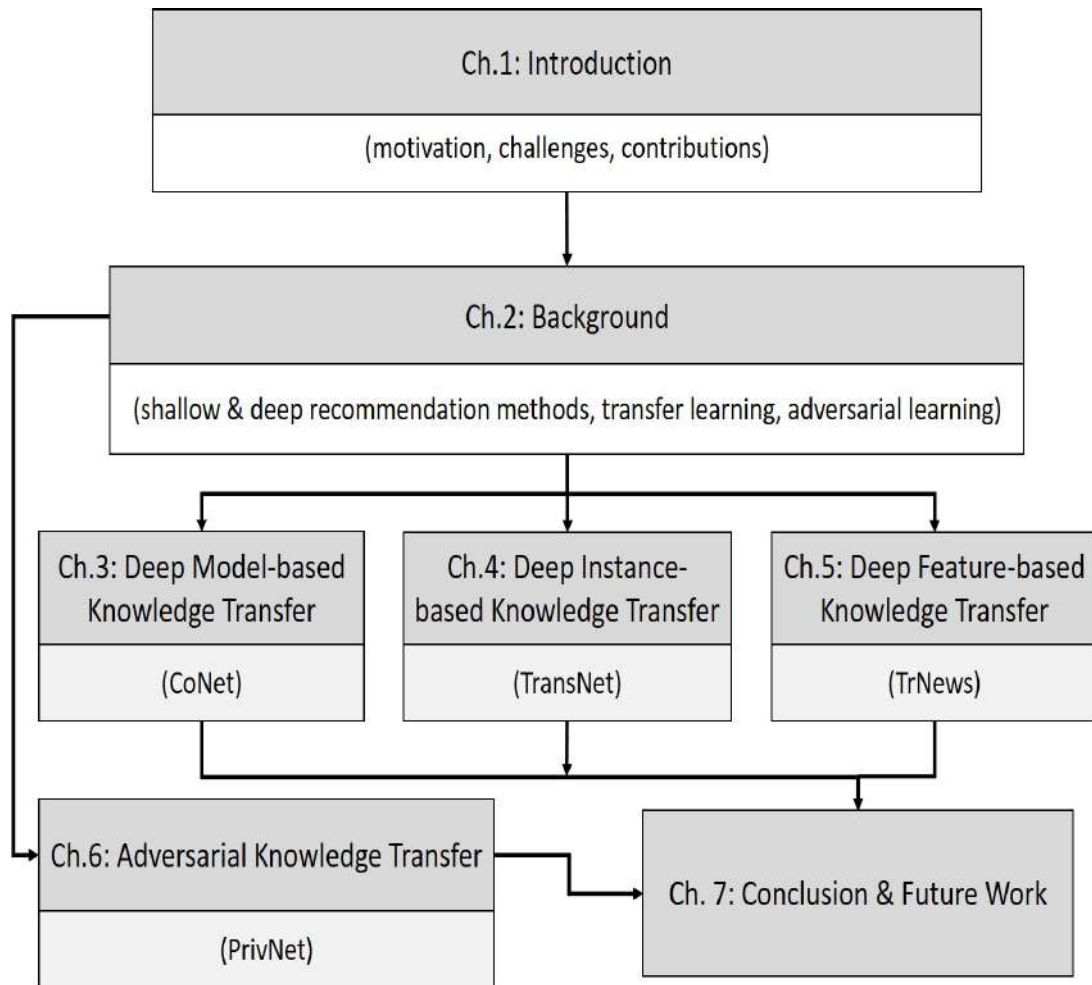


Figure 1.10: The Organization of Thesis.

Finally, in Chapter 7, we conclude our work with and then we describe potential future research directions.

Chapter 2

Background

In this chapter, we will first review some most recent developments in recommendation techniques especially the deep learning based models, and then three transfer learning approaches including model-based transfer, instance-based transfer, and feature-based transfer.

2.1 Recommendation Techniques

We introduce both shallow and deep recommender models in this section. The typical example of shallow model is the matrix factorization technique while the typical example of deep model is the neural collaborative filtering.

2.1.1 Matrix Factorization

Rating scores are the explicit user feedback and matrix factorization (MF) is a state-of-the-art recommender method to exploit this rating information. MF techniques have gained popularity and become the standard recommender approaches due to their accuracy and scalability. They have probabilistic interpretation with Gaussian noise and are very flexible to add side data sources for recommender such as reviews content and social relations introduced in the following subsections.

MF based RSs are mainly to find the latent user-specific matrix $U = [U_1, \dots, U_m] \in \mathbb{R}^{d \times m}$ and item-specific matrix $V = [V_1, \dots, V_n] \in \mathbb{R}^{d \times n}$, where d is the number of latent factors, obtained by solving the following problem:

$$\min_{U, V} \sum_{r_{i,j} \neq 0} (r_{i,j} - \hat{r}_{i,j})^2 + \lambda(\|U\|_F^2 + \|V\|_F^2), \quad (2.1)$$

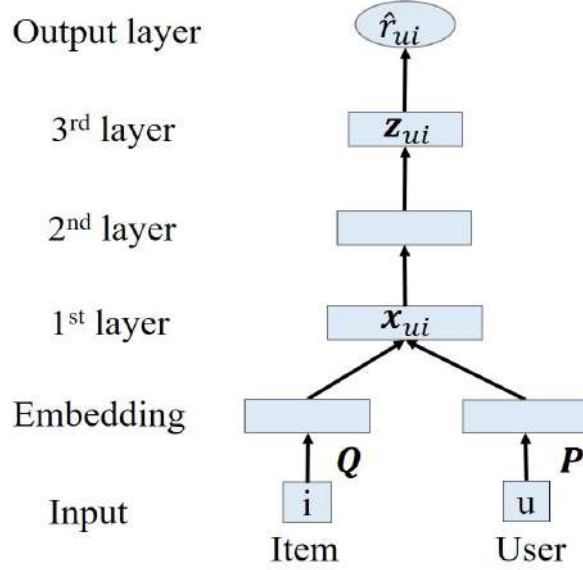


Figure 2.1: The Architecture of a Typical Neural Collaborative Filtering with Three Hidden Layers.

where the predicted rating is computed by:

$$\hat{r}_{i,j} = \mu + b_i + b_j + U_i^T V_j, \quad (2.2)$$

and regularization parameter λ controls over-fitting. The rating mean is captured by μ ; b_i and b_j are rating biases of u_i and of v_j . The d -dimensional feature vectors U_i and V_j represent preferences for user i and characteristics for item j , respectively. The dot products $U_i^T V_j$ capture the interaction or match degree between users and items.

2.1.2 Deep Learning

Neural networks are proposed to push the learning of feature vectors towards non-linear representations, including the neural network matrix factorization (NNMF) and multilayer perceptron (MLP) [23, 42]. The basic MLP architecture is extended to regularize the factors of users and items by social and geographical information [132]. Other neural approaches learn from the explicit feedback for rating prediction task [11, 147]. We focus on learning from the implicit feedback for top-N recommendation [127]. CF models, however, suffer from the data sparsity issue.

A basic neural network is similar to the Deep model in [15, 17] and the MLP model in [42]. The base network consists of four modules which has been introduced in Chapter 1.2.2. We illustrate a three hidden-layers base network in Figure 2.1.

The basic neural network generalizes matrix factorization. Let $\mathbf{P}^T \mathbf{x}_i$ be that of latent user factors $\underline{U}_i$ in MF, and $\mathbf{Q}^T \mathbf{x}_j$ be that of latent item factors $\underline{V}_j$ in MF. MF computes the predicted score by $\hat{r}_{ij} = \underline{U}_i^T \underline{V}_j$ (ignoring the biases). We replace the concatenation in the embedding module with element-wise multiplication:

$$\mathbf{x}_{ij} = (\mathbf{P}^T \mathbf{x}_i) \odot (\mathbf{Q}^T \mathbf{x}_j). \quad (2.3)$$

It requires that the dimensions of latent features of users and items are the same in this case (denoted as d). We do not perform computation in hidden layers and let the output layer be the identity mapping. In this way, the prediction is the same whatever it is computed by matrix factorization or neural network.

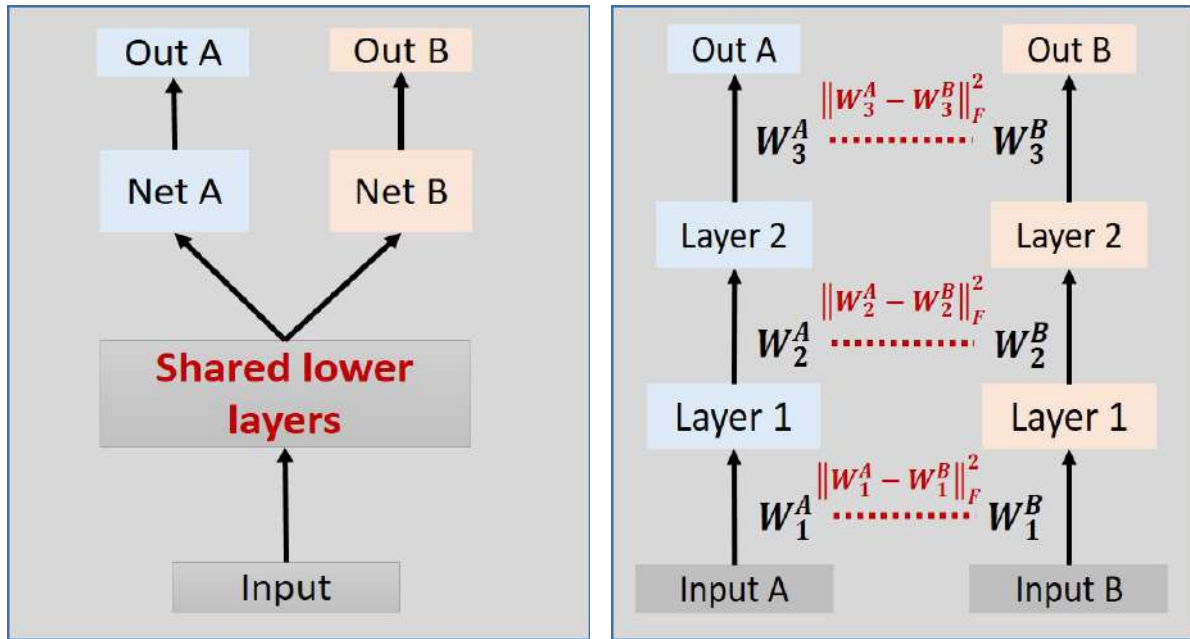
2.2 Transfer Learning Approaches

Transfer learning (TL) aims at improving the performance of the target domain by exploiting knowledge from source domains [93]. We introduce three transfer learning approaches including model-based transfer, instance-based transfer, and feature-based transfer.

2.2.1 Model-based Knowledge Transfer

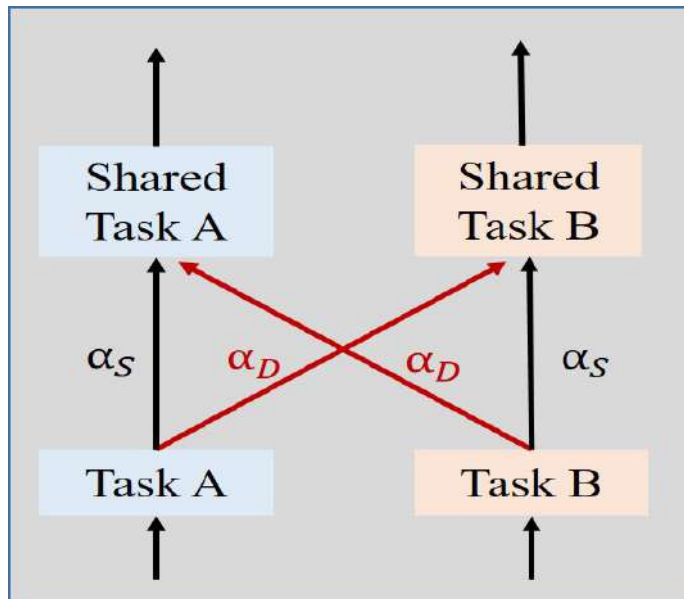
Transfer learning in recommendation [9] is an effective technique to alleviate the data sparsity issue in one domain by exploiting the knowledge from other domains. Typical methods apply matrix factorization [95, 111, 133] and representation learning [29, 75, 77, 132, 143] on each domain and share the user (item) factors, or learn a cluster level rating pattern [61, 142]. Transfer learning is to improve the target performance by exploiting knowledge from auxiliary domains [13, 24, 30, 93, 144]. One transfer strategy (two-stage) is to initialize a target network with transferred representations from a pre-trained source network [90, 140]. Another transfer strategy (end-to-end) is to transfer knowledge in a mutual way such that the source and target networks benefit from each other during the training, with examples including the cross-stitch networks [83] and collaborative cross networks [43].

The typical TL technique in neural networks is two-step: initialize a target network with transferred features from a pre-trained source network [90, 140]. Similar to TL, the multitask learning (MTL) is to leverage useful knowledge in multiple related tasks to help each other [10, 144]. Multi-view learning [24] is closely related to MTL. The cross-stitch network (CSN) [83] enables information sharing between two base networks.



(a) Hard Sharing

(b) Soft sharing by regularization



(c) Soft sharing by linear combination via scalar multiplication

Figure 2.2: Three Model-based Transfer Learning Paradigms. (a) Hard Sharing: sharing hidden layers directly while keeping the task-specific layers. Soft sharing: No directly sharing of parameters and (b) using regularization terms to encourage parameters to be similar, and (c) modeling shared parameters by learning a linear combination of input activation maps from two networks.

We illustrate three typical model-based transfer learning approaches in Figure 2.2.

2.2.2 Instance-based Knowledge Transfer

Transfer learning (TL) aims at improving the performance of the target domain by exploiting knowledge from source domains [93]. Similar to TL, the multitask learning (MTL) is to leverage useful knowledge in multiple related tasks to help each other [10, 144]. The cross-stitch network [83] and its sparse variant [43] enable information sharing between two base networks for each domain in a deep way. Robust learning is also considered during knowledge transfer [40]. These methods treat knowledge transfer as a global process with shared global parameters and do not match source items with the specific target item given a user.

The K-nearest neighbors method is used to find relevant source instances to be transferred into the target domain [33]. It first projects both source and target instances into the shared feature space, and then it selects nearest neighbors instances in the source domain for each target domain training sample in such low-level feature space (see the right part as shown in Figure 2.3).

We illustrate two typical instance-based transfer learning approaches in Figure 2.3.

2.2.3 Feature-based Knowledge Transfer

Transfer learning aims at improving the performance of a target domain by exploiting knowledge from source domains [93]. A special setting is domain adaptation where a source domain provides labeled training examples while the target domain provides instances on which the model is meant to be deployed [34, 66]. The coordinate system transfer (CST) [95] firstly learns the principle coordinate of users in the source domain, and then transfers it to the target domain in the way of warm-start initialization. This is equivalent to an identity mapping from users' source representations to their corresponding target representations. TCB [69] learns a linear mapping to translate target feature examples to the source feature space because there are many labelled data in the source domain. This linear strategy is also used in CoNet [43] and DDTCDR [65] which transforms the source representations to the target domain by a translation matrix. Nonlinear mapping strategy [28, 77, 149] is to learn a supervised mapping function between source and target latent factors by using neural networks. SSCDR [54] extends them to the semi-supervised mapping setting. Our translator is general to accommodate these identity, linear, and nonlinear transfer-learning strategies.

Instance	User			Target Item			Source Items				Feedback
x1	1	0	0	1	0	0	1	1	0	0	y1
x2	1	0	0	0	1	0	1	1	1	0	y2
x3	0	1	0	0	0	1	0	1	1	0	y3
x4	0	0	1	1	0	0	0	0	1	1	y4
x5	0	0	1	0	0	1	0	0	0	1	y5













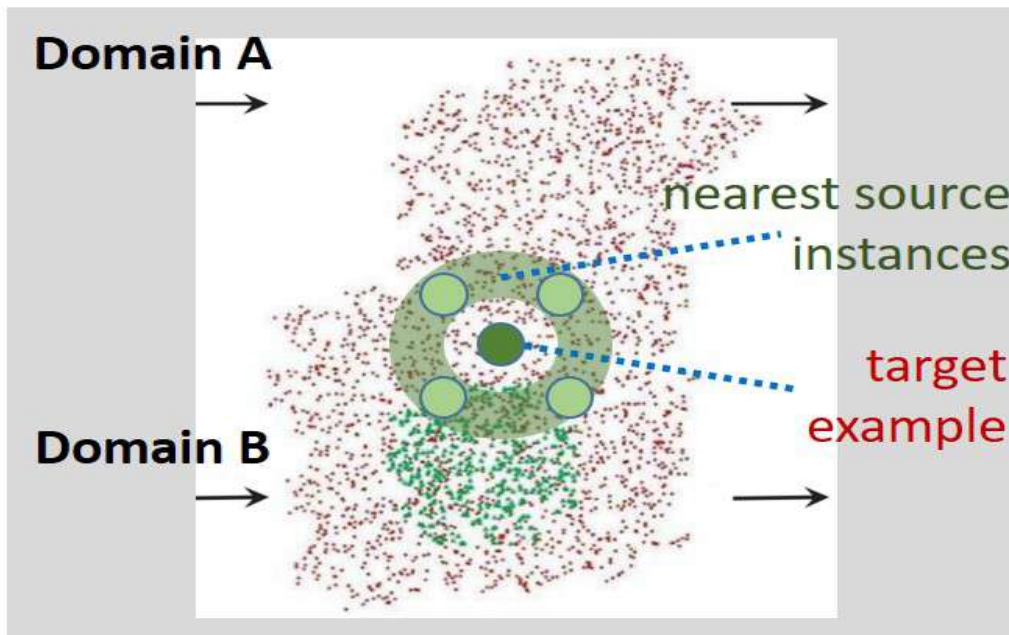








(a) Weight of source instance is part of the model parameters



(b) Target example's nearest neighbors as source transferred instances in shared feature space

Figure 2.3: Two Instance-based Transfer Learning Methods. (a) The method selects source instances by the model parameters itself (i.e., the instance weight is part of the parameter of the recommender model) and (b) the method selects source instances by the nearest neighbors of target examples found in the shared feature space.

Approach	Transfer strategy	Formulation
CST [95]	Identity mapping	$\phi_T(u) = \phi_S(u)$
TCB [69] DDTCDR [65]	Linear mapping	$\phi_T(u) = H\phi_S(u)$ H is orthogonal
EMCDR [77]	Nonlinear mapping	$\phi_T(u) = \text{MLP}(\phi_S(u))$

Table 2.1: Different Strategies in Feature-based Transfer Learning.

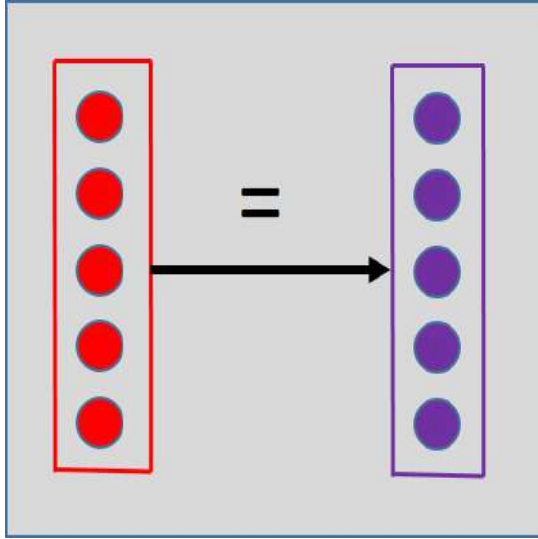
We summarize three strategies in feature-based transfer learning approaches in Table 5.2 and illustrate them in Figure 2.4.

2.2.4 Framework of deep transfer learning: An Anatomy

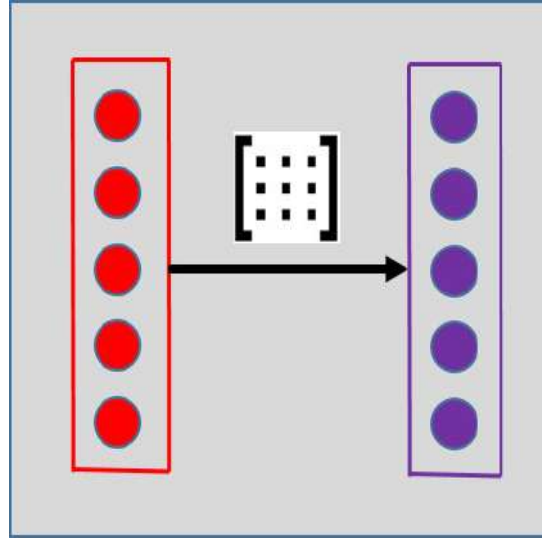
As shown in Figure 2.5, a deep transfer learning model can be decomposed into two basic parts. The first basic part is to define what the base network is. This is to answer how to model the individual domains, i.e., the source domain and target domain. The instantiation of a base network can arbitrarily implemented by any deep models including multi-layer feedforward neural networks, recurrent neural networks, and convolutional networks. The other part is the transfer unit which is the core of knowledge transfer. It builds the bridge across domains and enables the knowledge transfer from the source domain to the target domain. The model-based, instance-based, and feature-based transfer learning approaches can be instantiated by varying with the different transfer units.

2.3 Adversarial Learning

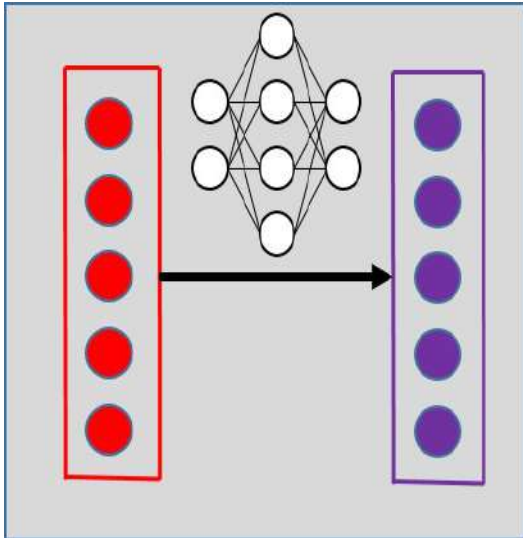
Existing privacy-preserving techniques mainly belong to three research threads. One thread adds noise (e.g., differential privacy [22]) to the released data or the output of recommender systems [53, 81, 82, 119, 121]. One thread perturbs user profiles such as adding (or deleting/changing) dummy items to the user history so that it hides the user’s actual ratings [100, 122]. Adding noise and perturbing ratings may still suffer from privacy inference attacks when the attacker can successfully distinguish the true profiles from the noisy/perturbed ones. Furthermore, they may degrade performance since data is corrupted. Another thread uses adversary loss [5, 106] to formulate the privacy attacker and the recommender system as an adversarial learning problem. However, they face the data sparsity issues. A recent work [103] trains linear classifiers to predict a protected attribute and then remove it by projecting the representation



(a) Identity mapping, i.e., no adaptation



(b) Linear Mapping



(c) Nonlinear mapping in a dilated way

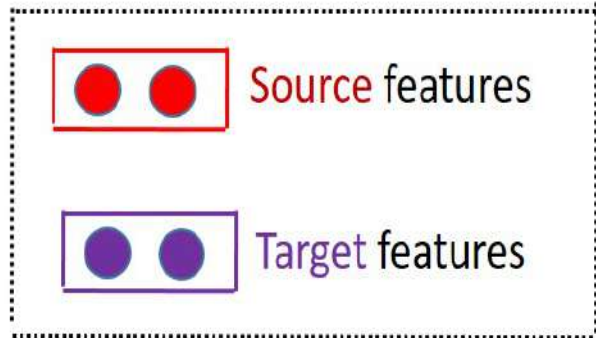


Figure 2.4: Three Feature-based Transfer Learning Paradigms. Example methods of each paradigm are shown in Table 5.2. (a) The identity mapping equals to no adaptation. (b) The matrix symbol means the feature mapping is implemented by linear transfer. (c) The neural network icon mean the feature mapping is implemented by nonlinear transfer in a dilated way (i.e., the hidden layers have more number of neurons than the input/output layers.).

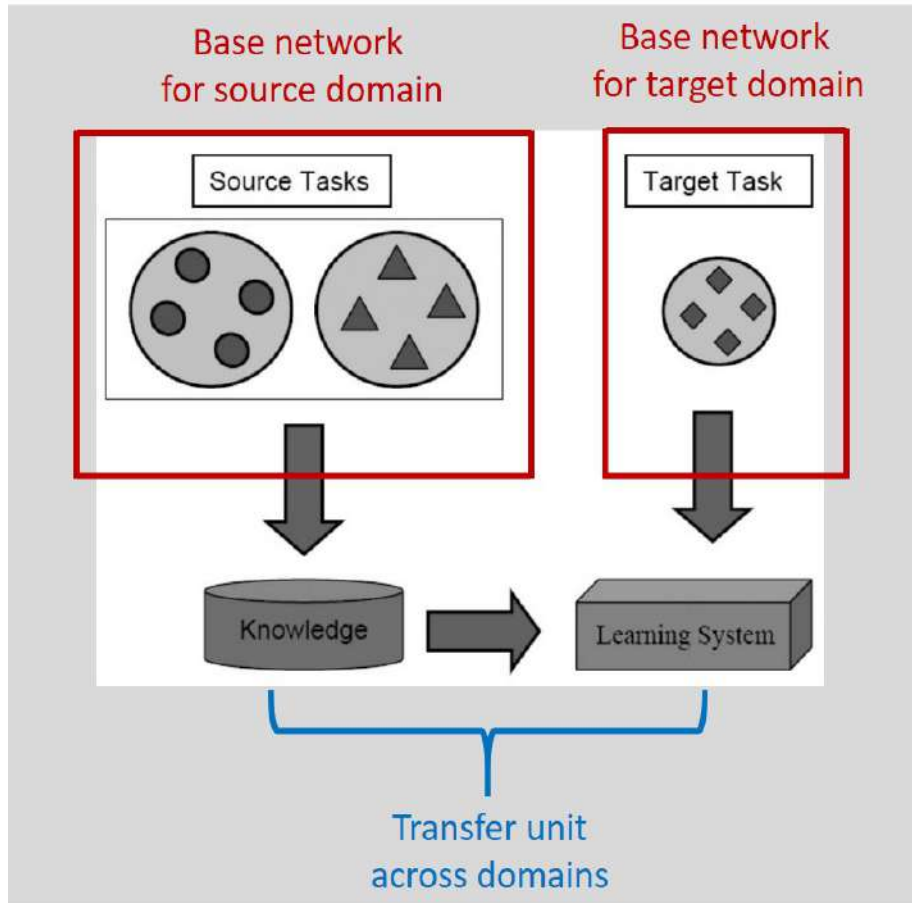


Figure 2.5: Framework of deep transfer learning: An Anatomy. It can be composed into two parts. The red part is the base network which is to model the individual domains (source domain and target domain). The blue part is the transfer unit which is to build the bridge across domains answering “what to transfer” (and “how to transfer”). This figure is adapted from [93].

on its null-space. Some other work uses encryption and federated learning so as to protect the personal data without affecting performance [14, 87, 118]. They suffer from efficiency and scalability due to high cost of computation and communication.

Chapter 3

Deep Model-based Knowledge Transfer in Recommendation

The cross-domain recommendation technique is an effective way of alleviating the data sparse issue in recommender systems by leveraging the knowledge from relevant domains. Transfer learning is a class of algorithms underlying these techniques. In this chapter, we propose a novel transfer learning approach for cross-domain recommendation by using neural networks as the base model. In contrast to the matrix factorization based cross-domain techniques, our method is deep transfer learning, which can learn complex user-item interaction relationships. We assume that hidden layers in two base networks are connected by cross mappings, leading to the collaborative cross networks (CoNet). CoNet enables dual knowledge transfer across domains by introducing cross connections from one base network to another and vice versa. CoNet is achieved in multi-layer feedforward networks by adding dual connections and joint loss functions, which can be trained efficiently by back-propagation. The proposed model is thoroughly evaluated on two large real-world datasets. It outperforms baselines by relative improvements of 7.84% in NDCG. We demonstrate the necessity of adaptively selecting representations to transfer. Our model can reduce tens of thousands training examples comparing with non-transfer methods and still has the competitive performance with them.

3.1 Introduction

Collaborative filtering (CF) approaches, which model the preference of users on items based on their past interactions such as product ratings, are the corner stone for recommender systems. Matrix factorization (MF) is a class of CF methods which learn user latent factors and item latent

factors by factorizing their interaction matrix [59, 84]. Neural collaborative filtering is another class of CF methods which use neural networks to learn the complex user-item interaction function [15, 23, 42]. Neural networks have the ability to learn highly nonlinear function, which is suitable to learn the complex user-item interaction. Both traditional matrix factorization and neural collaborative filtering, however, suffer from the cold-start and data sparse issues.

One effective solution is to transfer the knowledge from relevant domains and the cross-domain recommendation techniques address such problems [8, 9, 61, 94]. In real life, a user typically participates several systems to acquire different information services. For example, a user installs applications in an app store and reads news from a website at the same time. It brings us an opportunity to improve the recommendation performance in the target service (or all services) by learning across domains. Following the above example, we can represent the app installation feedback using a binary matrix where the entries indicate whether a user has installed an app. Similarly, we use another binary matrix to indicate whether a user has read a news article. Typically these two matrices are highly sparse, and it is beneficial to learn them simultaneously. This idea is sharpened into the collective matrix factorization (CMF) [111] approach which jointly factorizes these two matrices by sharing the user latent factors. It combines CF on a target domain and another CF on an auxiliary domain, enabling knowledge transfer [91, 144]. CMF, however, is a shallow model and has the difficulty in learning the complex user-item interaction function [23, 42]. Moreover, its knowledge sharing is only limited in the lower level of user latent factors.

Motivated by benefitting from both knowledge transfer learning and learning interaction function, we propose a novel deep transfer learning approach for cross-domain recommendation using neural networks as the base model. Though neural CF approaches are proposed for single domain recommendation [42], there are few related works to study knowledge transfer learning for cross-domain recommendation using neural networks. Instead, neural networks have been used as the base model in natural language processing [16, 138] and computer vision [21, 83, 140]. We explore how to use a neural network as the base model for each domain and enable the knowledge transfer on the entire network across domains. Then a few questions and challenges are raised: 1) What to transfer/share between these individual networks for each domain? 2) How to transfer/share during the learning of these individual networks for each domain? and 3) How is the performance compared with single domain neural learning and shallow cross-domain models?

This chapter aims at proposing a novel deep transfer learning approach by answering these questions under cross-domain recommendation scenario. The usual transfer learning approach is to train a base network and then copy its first several layers to the corresponding first layers of a target network with fine-tuning or parameter frozen [140]. This way of transferring has possibly two weak points. Firstly, the shared-layer assumption is strong in practice as we find that it does not work well on real-world cross-domain datasets. Secondly, the knowledge transfer happens in one direction, i.e., only from source to target. Instead, we assume that hidden layers in two base networks are connected by dual mappings, which do not require them to be identical. We enable dual knowledge transfer across domains by introducing cross connections from one base network to another and vice versa, letting them benefit from each other. These ideas are sharpened into the proposed collaborative cross networks (CoNet). CoNet is achieved in simple multi-layer feedforward networks by using dual shortcut connections and joint loss functions, which can be trained efficiently by back-propagation.

3.2 Problem Description and Challenges

We first give the notations and describe the problem setting (Sec. 3.2.1). We then review a multi-layer feedforward neural network as the base network for collaborative filtering (Sec. 4.2.2).

3.2.1 Notation

We are given two domains, a source domain $\mathcal{S}$ (e.g., news recommendation) and a target domain $\mathcal{T}$ (e.g., app recommendation). As a running example, we let app recommendation be the target domain and news recommendation be the source domain. The set of users in both domains are shared, denoted by $\mathcal{U}$ (of size $m = |\mathcal{U}|$). Denote the set of items in $\mathcal{S}$ and $\mathcal{T}$ by $\mathcal{I}_S$ and $\mathcal{I}_T$ (of size $n_S = |\mathcal{I}_S|$ and $n_T = |\mathcal{I}_T|$), respectively. Each domain is a problem of collaborative filtering for implicit feedback [50, 91]. For the target domain, let a binary matrix $\mathbf{R}_T \in \mathbb{R}^{m \times n_T}$ describe user-app installing interactions, where an entry $r_{ui} \in \{0, 1\}$ is 1 (observed entries) if user u has an interaction with app i and 0 (unobserved) otherwise. Similarly, for the source domain, let another binary matrix $\mathbf{R}_S \in \mathbb{R}^{m \times n_S}$ describe user-news reading interactions, where the entry $r_{uj} \in \{0, 1\}$ is 1 if user u has an interaction with news j and 0 otherwise. Usually the interaction matrix is very sparse since a user only consumed a very small subset of all items.

For the task of item recommendation, each user is only interested in identifying top-N items.

The items are ranked by their predicted scores:

$$\hat{r}_{ui} = f(u, i | \Theta), \quad (3.1)$$

where f is the interaction function and Θ are model parameters. For matrix factorization (MF) techniques, the match function is the fixed dot product:

$$\hat{r}_{ui} = \mathbf{P}_u^T \mathbf{Q}_i, \quad (3.2)$$

and parameters are latent vectors of users and items $\Theta = \{\mathbf{P}, \mathbf{Q}\}$ where $\mathbf{P} \in \mathbb{R}^{m \times d}$, $\mathbf{Q} \in \mathbb{R}^{n \times d}$ and d is the dimension. For neural CF approaches, neural networks are used to parameterize function f and learn it from interactions:

$$f(\mathbf{x}_{ui} | \mathbf{P}, \mathbf{Q}, \theta_f) = \phi_o(\phi_L(\dots(\phi_1(\mathbf{x}_{ui}))\dots)), \quad (3.3)$$

where the input $\mathbf{x}_{ui} = [\mathbf{P}^T \mathbf{x}_u, \mathbf{Q}^T \mathbf{x}_i]$ is merged from projections of the user and the item, and the projections are based on their one-hot encodings $\mathbf{x}_u \in \{0, 1\}^m$, $\mathbf{x}_i \in \{0, 1\}^n$ and embedding matrices $\mathbf{P} \in \mathbb{R}^{m \times d}$, $\mathbf{Q} \in \mathbb{R}^{n \times d}$. The output and hidden layers are computed by ϕ_o and $\{\phi_l\}$ in a multilayer feedforward neural network (FFNN), and the connection weight matrices and biases are denoted by θ_f .

In our transfer/multitask learning approach for cross-domain recommendation, each domain is modelled by a neural network and these networks are jointly learned to improve the performance through mutual knowledge transfer. We review the base network in the following subsection before introducing the proposed model.

3.2.2 Base Network

We adopt an FFNN as the base network to parameterize the interaction function (see Eq.(4.2)). The base network is similar to the Deep model in [15, 17] and the MLP model in [42]. The base network, as shown in Figure 3.2 (the gray part or the blue part), consists of four modules with the information flow from the input (u, i) to the output $\hat{r}_{ui}$ as follows.

Input : $(u, i) \rightarrow \mathbf{x}_u, \mathbf{x}_i$. This module encodes user-item interaction indices. We adopt the one-hot encoding. It takes user u and item i , and maps them into one-hot encodings $\mathbf{x}_u \in \{0, 1\}^m$ and $\mathbf{x}_i \in \{0, 1\}^n$ where only the element corresponding to that index is 1 and all others are 0.

Embedding : $\mathbf{x}_u, \mathbf{x}_i \rightarrow \mathbf{x}_{ui}$. This module embeds one-hot encodings into continuous representations via two embedding matrices and then merges them as $\mathbf{x}_{ui} = [\mathbf{P}^T \mathbf{x}_u, \mathbf{Q}^T \mathbf{x}_i]$ to be the input of successive hidden layers.

Hidden layers: $\mathbf{x}_{ui} \rightsquigarrow \mathbf{z}_{ui}$. This module takes the continuous representations from the embedding module and then transforms, through multi-hop say L , to a final latent representation $\mathbf{z}_{ui} = \phi_L(\dots(\phi_1(\mathbf{x}_{ui})\dots))$. This module consists of multiple hidden layers to learn nonlinear interaction between users and items.

Output : $\mathbf{z}_{ui} \rightarrow \hat{r}_{ui}$. This module predicts the score $\hat{r}_{ui}$ for the given user-item pair based on the representation $\mathbf{z}_{ui}$ from the last layer of multi-hop module. Since we focus on one-class collaborative filtering, the output is the probability that the input pair is a positive interaction. This can be achieved by a softmax layer:

$$\hat{r}_{ui} = \phi_o(\mathbf{z}_{ui}) = \frac{1}{1 + \exp(-\mathbf{h}^T \mathbf{z}_{ui})}, \quad (3.4)$$

where $\mathbf{h}$ is the parameter.

3.2.3 Cross-stitch Networks

We first introduce an intuitive model to realize cross-domain recommendation using neural networks, and point out several intrinsic strong assumptions limiting its use, which inspire the design of our model in the next section.

Given two activation maps a_A and a_B from the l -th layer for two tasks A and B , cross-stitch convolutional networks (CSN) [83] learn linear combinations $\tilde{a}_A, \tilde{a}_B$ of both the input activations and feed these combinations as input to the successive layers' filters (see Fig. 3.1a):

$$\tilde{a}_A^{ij} = \alpha_S a_A^{ij} + \alpha_D a_B^{ij}, \quad (3.5a)$$

$$\tilde{a}_B^{ij} = \alpha_S a_B^{ij} + \alpha_D a_A^{ij}, \quad (3.5b)$$

where the shared parameter α_D controls information shared/ transferred from the other network, α_S controls information from the task-specific network, and (i, j) is the location in the activation map.

Although the cross-stitch unit indeed incorporates knowledge from the source domain (and target domain vice versa), there are several limitations of this simple stitch unit. Firstly, cross-stitch networks cannot process the case that the dimensions of contiguous layers are different. In other words, it assumes that the activations in successive layers are in the *same vector space*. This is not an issue in convolutional networks for computer vision since the activation maps of contiguous layers are in the same space [60]. For collaborative filtering, however, it is not the case in typical multi-layer FFNNs where the architecture follows a tower pattern: the lower

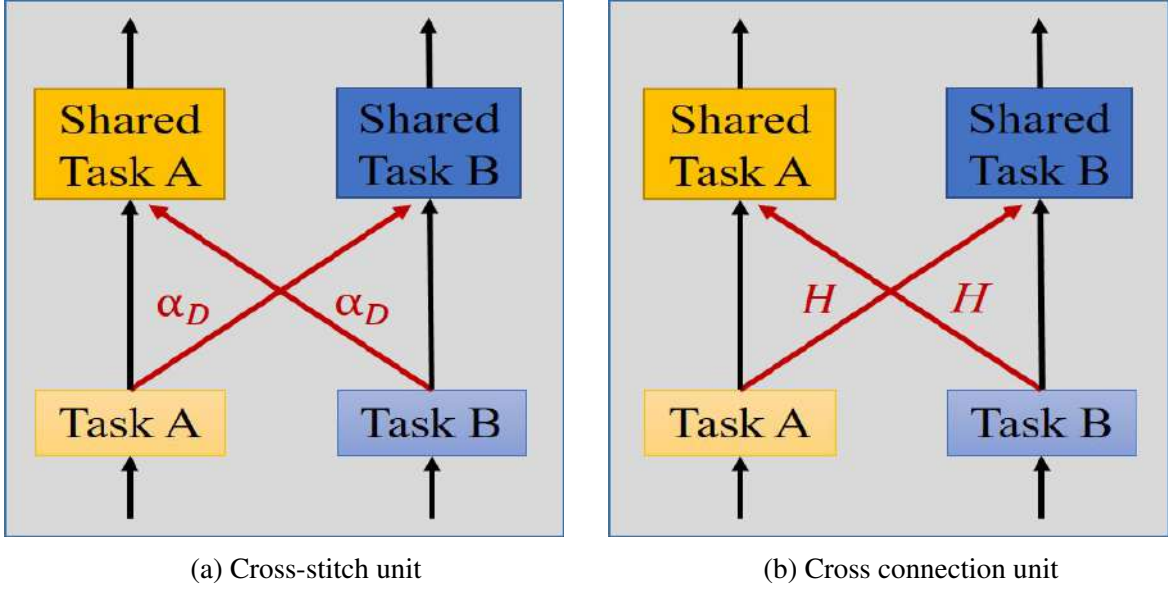


Figure 3.1: The cross-stitch unit [83] (left) and the proposed cross connection unit (right). To enable knowledge transfer, the shared α_D in the cross-stitch network is a scalar while the shared H in the proposed collaborative cross network is a matrix.

layers are wider and higher layers have smaller number of neurons [17, 42]. Secondly, it assumes that the representations from other networks are *equally important* with weights being all the same scalar α_D . Some features, however, are more useful and predictive and it should be learned attentively from data [130]. Thirdly, it assumes that the representations from other networks are *all useful* since it transfers activations from every location in a dense way. The sparse structure, however, plays a key role in general learning paradigm [21]. Instead, our model can be extended to learn the sparse structure on the task relationship matrices which are defined in Eq. (3.9), with the help of the existing sparsity-induced regularization. As we will see in the experiments (see Table 3.3 and Figures 3.3 and 3.4), the sparse structure is necessary for generalization performance.

3.3 Methodology

3.3.1 Collaborative Cross Networks

To alleviate the limitations of the cross-stitch networks, we propose collaborative cross networks (CoNet) to transfer knowledge for the cross-domain recommendation. The core component is cross connection units (Sec. 3.3.1). Our cross unit generalizes the cross-stitch (Sec. 3.3.1) and exploits the sparse structure (Sec. 3.3.1). We describe the model learning from implicit

feedback datasets and the optimization process of the joint loss. A complexity analysis is also given (Sec. 4.3.3).

Cross Connections Unit

In this section, we present a novel soft-sharing approach for transferring knowledge for cross-domain recommendation. It relaxes the hard-sharing assumption [140] and is motivated by the cross-stitch networks [83].

We now introduce the cross connections unit to enable dual knowledge transfer as shown in Fig. 3.1b. The central idea is simple, using a matrix rather than a scalar to transfer. Similarly to the cross-stitch network, the target network receives information from the source network and vice versa. In detail, let $\mathbf{a}_{app}$ be the representations of the l -th hidden layer and $\tilde{\mathbf{a}}_{app}$ be the input to the $l + 1$ -th in the app network, respectively. Similarly, they are $\mathbf{a}_{news}$ and $\tilde{\mathbf{a}}_{news}$ in the news network. The cross unit implements as follows:

$$\tilde{\mathbf{a}}_{app} = \mathbf{W}_{app}\mathbf{a}_{app} + \mathbf{H}\mathbf{a}_{news}, \quad (3.6a)$$

$$\tilde{\mathbf{a}}_{news} = \mathbf{W}_{news}\mathbf{a}_{news} + \mathbf{H}\mathbf{a}_{app}, \quad (3.6b)$$

where $\mathbf{W}_{app}$ and $\mathbf{W}_{news}$ are weight matrices, and the matrix $\mathbf{H}$ controls the information from news network to app network and vice versa. The knowledge transferring happens in two directions, from source to target and from target to source. We enable dual knowledge transfer across domains and let them benefit from each other. When target domain data is sparse, the target network can still learn a good representation from that of the source network through the cross connection units. It only needs to learn “residual” target representations with the reference of source representations, making the target task learning easier and hence alleviating the data sparse issues. The role of matrix $\mathbf{H}$ is similar to the scalar α_D in the sense of enabling knowledge transfer between domains.

We give a closer look at the matrix $\mathbf{H}$ for it can alleviate all three issues faced by cross-stitch unit. Firstly, the successive layers can be in *different vector space* (spaces with different dimensions) since the matrix $\mathbf{H}$ can be used to match their dimension. For example, if the l -th layer ($\mathbf{a}_{app}$ and $\mathbf{a}_{news}$) has dimension 128, and the $(l + 1)$ -th layer ($\tilde{\mathbf{a}}_{app}$ and $\tilde{\mathbf{a}}_{news}$) has dimension 64, then the matrix $\mathbf{H} \in \mathbb{R}^{64 \times 128}$. Secondly, the entries of $\mathbf{H}$ are learned from data. They are likely not to be all the same, showing that the *importances* of transferred representations are different for each neuron/position. Thirdly, we can enforce some prior on the matrix $\mathbf{H}$ to

exploit the structure of the neural architecture. The sparse structure can be enforced to adaptively *select useful representations* to transfer. Based on the cross connection units, we propose the CoNet models in the following sections, including a basic model (Sec. 3.3.1) and an adaptive variant (Sec. 3.3.1).

Basic Model

We propose the collaborative cross network (CoNet) model by adding cross connection units (see Sec. 3.3.1) and joint loss (see Eq.(3.12)) to the entire FFNN, as shown in Figure 3.2. We firstly describe a basic model in this section and then present an adaptive variant in the next section.

We decompose the model parameters into two parts, task-shared and task-specific:

$$\Theta_{app} = \{\mathbf{P}, (\mathbf{H}^l)_1^L\} \cup \{\mathbf{Q}_{app}, \theta_{f_{app}}\} \quad (3.7a)$$

$$\Theta_{news} = \{\mathbf{P}, (\mathbf{H}^l)_1^L\} \cup \{\mathbf{Q}_{news}, \theta_{f_{news}}\}, \quad (3.7b)$$

where $\mathbf{P}$ is the user embedding matrix and $\mathbf{Q}$ are the item embedding matrices with the subscript specifying the corresponding domain. The $\theta_f = \{(\mathbf{W}^l, \mathbf{b}^l)_{l=1}^L, \mathbf{h}\}$ are the connection weight matrices and biases in the L -layer FFNN where $\mathbf{h}$ is the output weight as shown in Eq.(3.4). We stack the cross connections units on the top of the shared user embeddings, enabling deep knowledge transfer. Denote by $\mathbf{W}^l$ the weight matrix connecting from the l -th to the $l+1$ -th layer (we ignore biases for simplicity), and by $\mathbf{H}^l$ the linear projection underlying the corresponding cross connections. Then two base networks are coupled by cross connections:

$$\mathbf{a}_{app}^{l+1} = \sigma(\mathbf{W}_{app}^l \mathbf{a}_{app}^l + \mathbf{H}^l \mathbf{a}_{news}^l), \quad (3.8a)$$

$$\mathbf{a}_{news}^{l+1} = \sigma(\mathbf{W}_{news}^l \mathbf{a}_{news}^l + \mathbf{H}^l \mathbf{a}_{app}^l), \quad (3.8b)$$

where the function $\sigma(\cdot)$ is the widely used rectified activation units (ReLU) [85]. We can see that $\mathbf{a}_{app}^{l+1}$ receives two information flows: one is from the *transform* gate controlled by $\mathbf{W}_{app}^l$ and one is from the *transfer* gate controlled by $\mathbf{H}^l$ (similarly for the $\mathbf{a}_{news}^{l+1}$ in source network). We call $\mathbf{H}^l$ the relationship/transfer matrix since it learns to control how much sharing is needed. To reduce model parameters and make the model compact, we use the same linear transformation $\mathbf{H}^l$ for two directions, similar to the cross-stitch networks. Actually, using different matrices for two directions did not improve results on the evaluated datasets.

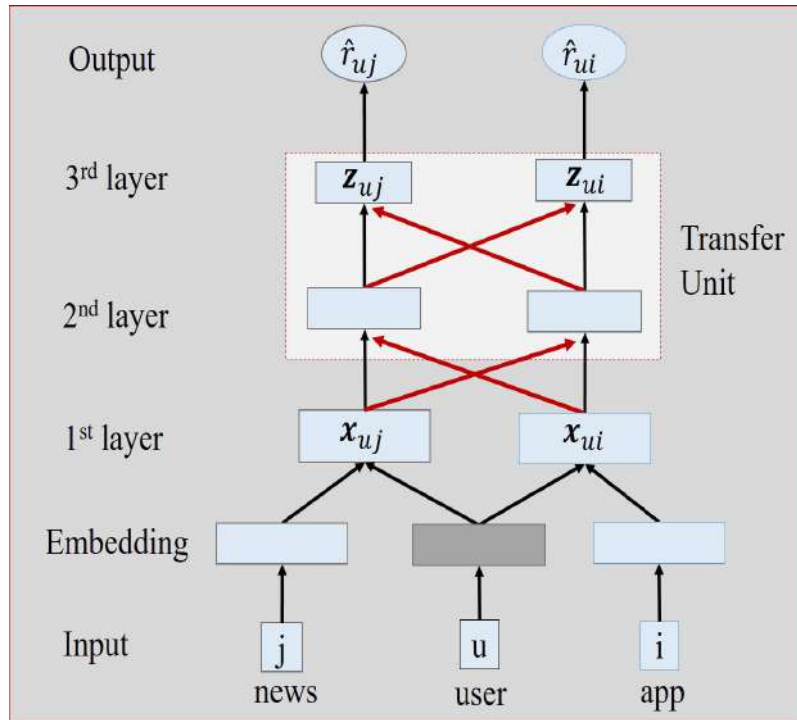


Figure 3.2: The proposed collaborative cross networks (CoNet) architecture (a version of three hidden layers and two cross units). We adopt a multilayer FFNN as the base network (grey or blue part, see Sec. 4.2.2). The red dotted lines indicate the cross connections which enable the dual knowledge transfer across domains. A cross unit is illustrated in the dotted rectangle box (see Fig. 3.1b).

Adaptive Model

As we can see, the task relationship matrices $\{\mathbf{H}^l\}$ are crucial to the proposed CoNet model. We further enforce these matrices to have some structure. The assumption is that not all representations from another network are useful. We may expect that the representations coming from other domains are sparse and selective. This corresponds to enforcing a sparse prior on the structure and can be achieved by penalizing the task relationship matrix $\{\mathbf{H}^l\}$ via some regularization. It may help the individual network to learn intrinsic representations for itself and other tasks. In other words, $\{\mathbf{H}^l\}$ adaptively controls when to transfer.

We adopt the widely used sparsity-induced regularization—least absolute shrinkage and selection operator (lasso) [113]. In detail, denote by $r \times p$ the size of matrix $\mathbf{H}^l$ (usually $r = p/2$). That is, $\mathbf{H}^l$ linearly transforms representations $\mathbf{a}_{news}^l \in \mathbb{R}^p$ in the news network and the result is as part of the input to the next layer $\tilde{\mathbf{a}}_{app}^{l+1} \in \mathbb{R}^r$ in the app network (see Eq.(3.8) and Eq.(3.6)). Denote by h_{ij} the (i, j) entry of $\mathbf{H}^l$. To induce overall sparsity, we impose the ℓ_1 -norm penalty on the entries $\{h_{ij}\}$ of $\mathbf{H}^l$:

$$\Omega(\mathbf{H}^l) = \lambda \sum_{i=1}^r \sum_{j=1}^p |h_{ij}|, \quad (3.9)$$

where hyperparameter λ controls the degree of sparsity. This corresponds to the lasso regularization. We call this sparse variant as the SCoNet model.

Other priors like low-rank ($\mathbf{H} \approx \mathbf{U}^T \mathbf{V}$) factorization are alternatives of sparse structure. And the lasso variants like group lasso and sparse group lasso are also possible. We adopt the general sparse prior and the widely used lasso regularization. The others are left for future work.

3.3.2 Objective and Optimization

Due to the nature of the implicit feedback and the task of item recommendation, the squared loss $(\hat{r}_{ui} - r_{ui})^2$ is not suitable since it is usually for rating regression/prediction. Instead, we adopt the cross-entropy loss:

$$\mathcal{L}_0 = - \sum_{(u,i) \in \mathbf{R}^+ \cup \mathbf{R}^-} r_{ui} \log \hat{r}_{ui} + (1 - r_{ui}) \log(1 - \hat{r}_{ui}), \quad (3.10)$$

where $\mathbf{R}^+$ and $\mathbf{R}^-$ are the observed interaction matrix and randomly sampled negative examples [91], respectively. This objective function has probabilistic interpretation and is the negative logarithm likelihood of the following likelihood function:

$$L(\Theta | \mathbf{R}^+ \cup \mathbf{R}^-) = \prod_{(u,i) \in \mathbf{R}^+} \hat{r}_{ui} \prod_{(u,i) \in \mathbf{R}^-} (1 - \hat{r}_{ui}), \quad (3.11)$$

where Θ are model parameters.

Now we define the joint loss function, leading to the proposed CoNet model which can be trained efficiently by back-propagation. Instantiating the *base loss* ($\mathcal{L}_0$) described in Eq. (3.10) by the *loss of app* ($\mathcal{L}_{app}$) and *loss of news* ($\mathcal{L}_{news}$) recommendation, the objective function for the CoNet model is their joint losses:

$$\mathcal{L}(\Theta) = \mathcal{L}_{app}(\Theta_{app}) + \mathcal{L}_{news}(\Theta_{news}), \quad (3.12)$$

where model parameters $\Theta = \Theta_{app} \cup \Theta_{news}$. Note that Θ_{app} and Θ_{news} share user embeddings and transfer matrices $\{\mathbf{P}, (\mathbf{H}^l)_{l=1}^L\}$. For the CoNet-sparse model, the objective function is added by the term $\Omega(\mathbf{H}^l)$ in Eq.(3.9).

The objective function can be optimized by stochastic gradient descent (SGD) and its variants like adaptive moment method (Adam) [56]. The update equations are:

$$\Theta^{new} \leftarrow \Theta^{old} - \eta \frac{\partial L(\Theta)}{\partial \Theta}, \quad (3.13)$$

where η is the learning rate. Typical deep learning library like TensorFlow¹ provides automatic differentiation and hence we omit the gradient equations $\frac{\partial L(\Theta)}{\partial \Theta}$ which can be computed by chain rule in back-propagation (BP).

3.3.3 Complexity Analysis

The model parameters Θ include:

$$\{\mathbf{P}, (\mathbf{H}^l)_{l=1}^L\} \cup \{\mathbf{Q}_{app}, (\mathbf{W}_{app}^l, b_{app}^l)_{l=1}^L, \mathbf{h}_{app}\} \cup \{\mathbf{Q}_{news}, (\mathbf{W}_{news}^l, b_{news}^l)_{l=1}^L, \mathbf{h}_{news}\}, \quad (3.14)$$

where the embedding matrices $\mathbf{P}$, $\mathbf{Q}_{app}$ and $\mathbf{Q}_{news}$ contain a large number of parameters since they depend on the input size of users and items. Typically, the number of neurons in a hidden layer is about one hundred. That is, the size of connection weight matrices and task relationship matrices is hundreds by hundreds. In total, the size of model parameters is linear with the input size and is close to the size of typical latent factors models [59] and neural CF approaches [42].

During training, we update the target network using the target domain data and update the source network using the source domain data. The learning procedure is similar to the cross-stitch networks [83]. And the cost of learning each base network is approximately equal to that of running a typical neural CF approach [42]. In total, the entire network can be efficiently trained by BP using mini-batch stochastic optimization.

¹<https://www.tensorflow.org>

Table 3.1: Datasets and Statistics.

Dataset	#Users	Target Domain			Source Domain		
		#Items	#Interactions	Density	#Items	#Interactions	Density
Mobile	23,111	14,348	1,164,394	0.351%	29,921	617,146	0.089%
Amazon	80,763	93,799	1,323,101	0.017%	35,896	963,373	0.033%

3.4 Experiments

We conduct thorough experiments to evaluate the proposed models. We show their superior performance over the state-of-the-art recommendation algorithms in a wide range of baselines and demonstrate the effectiveness of the sparse variant to select representations. We quantify the benefit of knowledge transfer by reducing training examples. Furthermore, we conduct investigations on the sensitivity to hyperparameter. We analyze the optimization efficiency to help understand the proposed models.

3.4.1 Data Sets and Evaluation Protocol

We begin the experiments by introducing the datasets, evaluation protocol, baselines, and implementation details.

Data Sets

We evaluate on two real-world cross-domain datasets. The first dataset, **Mobile**², is provided by a large internet company, i.e., Cheetah Mobile³. The information contains logs of user reading news, the history of app installation, and some metadata such as news publisher and user gender collected in one month in the US. The dataset we used contains 1,164,394 user-app installations and 617,146 user-news reading records. There are 23,111 shared users, 14,348 apps, and 29,921 news articles. We aim to improve the app recommendation by transferring knowledge from relevant news reading domain. The data sparsity is over 99.6%.

The second dataset is a public **Amazon** dataset⁴, which has been widely used to evaluate the performance of collaborative filtering approaches [41]. We use the two largest categories, Books and Movies & TV, as the cross-domain. We convert the ratings of 4-5 as positive samples. The dataset we used contains 1,323,101 user-book ratings and 963,373 user-movie ratings.

²An anonymous version can be released later.

³<http://www.cmcm.com/en-us/>

⁴<http://snap.stanford.edu/data/web-Amazon.html>

There are 80,763 shared users, 93,799 books, and 35,896 movies. We aim to improve the book recommendation by transferring knowledge from relevant movie watching domain. The data sparsity is over 99.9%. The statistics are summarized in Table 5.1. As we can see, both datasets are very sparse and hence we hope improve performance by transferring knowledge from auxiliary domains.

Evaluation Protocol

For item recommendation task, the leave-one-out (LOO) evaluation is widely used and we follow the protocol in [42]. That is, we reserve one interaction as the test item for each user. We determine hyper-parameters by randomly sampling another interaction per user as the validation/development set. We follow the common strategy which randomly samples 99 (negative) items that are not interacted by the user and then evaluate how well the recommender can rank the test item against these negative ones.

Since we aim at top-N item recommendation, the typical evaluation metrics are hit ratio (HR), normalized discounted cumulative gain (NDCG), and mean reciprocal rank (MRR), where the ranked list is cut off at $topN = 10$. HR intuitively measures whether the reserved test item is present on the top-N list, defined as:

$$HR = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \delta(p_u \leq topN),$$

where p_u is the hit position for the test item of user u , and $\delta(\cdot)$ is the indicator function. NDCG and MRR also account for the rank of the hit position, respectively defined as:

$$NDCG = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{\log 2}{\log(p_u + 1)}, \quad MRR = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{p_u}.$$

A higher value indicates better performance.

Implementation

For BPRMF, we use LightFM’s implementation⁵ which is a popular CF library. For CDCF, we adapt the official libFM implementation⁶. For MLP, we use the code released by its authors⁷. For CMF, we use a Python version reference to the original Matlab code⁸. Our methods are

⁵<https://github.com/lyst/lightfm>

⁶<http://www.libfm.org>

⁷https://github.com/hexiangnan/neural_collaborative_filtering

⁸<http://www.cs.cmu.edu/~ajit/cmff/>

Baselines	Shallow method	Deep method
Single-domain	BPRMF [105]	MLP [42]
Cross-domain	CDCF [71], CMF [111]	MLP++, CSN [83]

Table 3.2: Categorization of Baselines.

implemented using TensorFlow. Parameters are randomly initialized from Gaussian $\mathcal{N}(0, 0.01^2)$. The optimizer is Adam with initial learning rate 0.001. The size of mini batch is 128. The ratio of negative sampling is 1. As for the design of network structure, we adopt a tower pattern, halving the layer size for each successive higher layer. Specifically, the configuration of hidden layers in the base network is $[64 \rightarrow 32 \rightarrow 16 \rightarrow 8]$. This is also the network configuration of the MLP model. For CSN, it requires that the number of neurons in each hidden layer is the same. The configuration notation $[64] * 4$ equals $[64 \rightarrow 64 \rightarrow 64 \rightarrow 64]$. We investigate several typical configurations.

3.4.2 Baselines

We compare with various baselines:

BPRMF: Bayesian personalized ranking [105] is a typical latent factors CF approach which learns the user and item factors via matrix factorization and pairwise rank loss. It computes the prediction score by $\hat{r}_{ui} = \mathbf{P}_u^T \mathbf{Q}_i$ (see Eq.(3.2)). It is a shallow model and learns on the target domain only.

MLP: Multilayer perceptron [42] is a typical neural CF approach which learns user-item interaction function using neural networks. MLP corresponds to the base network as described in Section 4.2.2. It is a deep model and learns on the target domain only.

MLP++: We combine two MLPs by sharing the user embedding matrix only. This is a degenerated CoNet which has no cross connection units. It is a simple/shallow knowledge transfer approach applied to two domains.

CDCF: Cross-Domain CF with factorization machines (FM) [71] is a state-of-the-art cross-domain recommendation which extends FM [104]. It is a context-aware approach which applies factorization on the merged domains (aligned by the shared users). That is, the auxiliary domain is used as context. On the Mobile dataset, the context for a user in the target app domain is her history of reading news in the source news domain. Similarly, the context for a user in the target book domain is her history of watching movies in the source movie domain on the

Table 3.3: Comparison results of different methods on two datasets. The best results are boldfaced and the best baselines are marked with stars.

Dataset	Metric	BPRMF	CMF	CDCF	MLP	MLP++	CSN	CoNet	SCoNet	improve	paired t -test
Mobile	HR	.6175	.7879	.7812	.8405	.8445	.8458*	.8480	.8583	1.47%	$p = 0.20$
	NDCG	.4891	.5740	.5875	.6615	.6683	.6733*	.6754	.6887	2.29%	$p = 0.25$
	MRR	.4489	.5067	.5265	.6210	.6268	.6366*	.6373	.6475	1.71%	$p = 0.34$
Amazon	HR	.4723	.3712	.3685	.5014	.5050*	.4962	.5167	.5338	5.70%	$p = 0.02$
	NDCG	.3016	.2378	.2307	.3143	.3175*	.3068	.3261	.3424	7.84%	$p = 0.03$
	MRR	.2971	.1966	.1884	.3113*	.3053	.2964	.3163	.3351	7.65%	$p = 0.05$

Amazon dataset. The feature vector for the input is a sparse vector $\mathbf{x} \in \mathbb{R}^{m+n_T+n_S}$ where the non-zero entries are as follows: 1) the index for user id, 2) the index for item id (target domain), and all indices for her reading articles/watching movies (source domain). It showed better performance than other cross-domain methods like triadic (tensor) factorization [48]. It is a shallow cross-domain model.

CMF: Collective matrix factorization [111] is a multi-relation learning approach which jointly factorizes matrices of individual domains. Here, the relation is user-item interaction. On Mobile, the two matrices are $\mathbf{A}$ = “user by app” and $\mathbf{B}$ = “user by news” respectively. Similarly, they are $\mathbf{A}$ = “user by book” and $\mathbf{B}$ = “user by movie” on Amazon. The shared user factors $\mathbf{P}$ enable knowledge transfer between two domains. Then CMF factorizes matrices $\mathbf{A}$ and $\mathbf{B}$ simultaneously by sharing the user latent factors: $\mathbf{A} \approx \mathbf{P}^T \mathbf{Q}_A$ and $\mathbf{B} \approx \mathbf{P}^T \mathbf{Q}_B$. It is a shallow model and jointly learns on two domains. This can be thought of a non-deep transfer/multitask learning approach for cross-domain recommendation.

CSN: The cross-stitch network method [83], described in Section 3.2.3, is a good competitor. It is a deep multitask learning model which jointly learns two base networks. It enables knowledge transfer via a linear combination of activation maps from two networks via a shared coefficient, i.e., α_D in Eq.(3.5). This is a deep transfer/multitask learning approach for cross-domain recommendation.

3.4.3 Results

Comparing Different Approaches

In this section, we report the recommendation performance of different methods and discuss the findings. Table 3.3 shows the results of different models on the two datasets under three ranking metrics. The last two columns are the relative improvement and its paired t -test of our model vs. the best baselines. We can see that our proposed neural models are better than the

base network (MLP), the shallow cross-domain models (CMF and CDCF) learned using two domains information, and the deep cross-domain model (MLP++ and CSN) on both datasets.

On Mobile, our model achieves 4.28% improvements in terms of MRR comparing with the non-transfer MLP, showing the benefits of knowledge transfer. Note that, the way of pre-training an MLP on source domain and then transferring user embeddings to target domain as warm-up did not achieve much improvement. In fact, the improvement is so small that it can be ignored. It shows the necessity of dual knowledge transfer in a deep way. Our model improves more than 20% in terms of MRR comparing with CDCF and CMF, showing the effectiveness of deep neural approaches. Together, our neural models consistently give better performance than other existing methods. Within our models (SCoNet vs CoNet), enforcing sparse structure on the task relationship matrices are useful. Note that, the dropout technique and ℓ_2 norm penalty did not achieve these improvements. They may harm the performance in some cases. It shows the necessity of selecting representations.

On Amazon, our model achieves 7.84% improvements in terms of NDCG comparing with the best baselines (MLP++), showing the benefits of knowledge transfer. Compared to the BPRMF, the inferior performance of CMF and CDCF shows the difficulty in transferring knowledge between Amazon Books and Movies, but our models also achieve good results. Comparing MLP++ and MLP, sharing user embedding is slightly better than the base network due to shallow knowledge transfer. Within our models, enforcing sparse structure on the task relationship matrices are also useful.

CSN is inferior to the proposed CoNet models on both datasets. Moreover, it is surprising that the CSN has some difficulty in benefitting from knowledge transfer on the Amazon dataset since it is inferior to the non-transfer base network MLP. The reason is possibly that the assumptions of CSN are not appropriate: all representations from the auxiliary domain are *equally* important and are *all* useful. By using a matrix $\mathbf{H}$ rather than a scalar α_D , we can relax the first assumption. And by enforcing a sparse structure on the matrix, we also relax the second assumption.

Note that the relative improvement of the proposed model vs. the best baseline is more significant on the Amazon dataset than that on the Mobile dataset, though the Amazon is much sparser than the Mobile (see Table 5.1). One explanation is that the relatedness the book and movie domains is much larger than that between the app and news domains. This will benefit all cross-domain methods including CMF, CDCF, and CSN, since they exploit information from both two domains. Another possibility is that the noise from auxiliary domain proposes

a challenge for transferring knowledge. This shows that the proposed model is more effective since it can select useful representations from the source network and ignore the noisy ones. In the next section, we give a closer look at the impact of the sparse structure.

Impact of Sparsity: Selecting Representations to Transfer

On two real-world datasets, it both shows the usefulness of enforcing sparse structure on the task relationship matrices $\mathbf{H}$. We now quantify the contributions of the sparsity to CoNet. We investigate the impact of the sparsity by controlling the difference of architectures between CSN and CoNet. That is, we let them have the same architecture configuration. As a consequence, the performance of ablation comes from different means of knowledge transfer: scalar α_D used in CSN and sparse matrix $\mathbf{H}$ used in SCoNet.

Figure 3.3 and Figure 3.4 show the results on the Mobile and Amazon datasets under several typical architectures. We can see that the sparsity contributes to performance improvements and it is necessary to introduce the sparsity in general settings. On the Mobile data, introducing the sparsity improves the NDCG by relatively 2.29%. On the Amazon data, introducing the sparsity improves the NDCG by relatively 4.21%. These results show that it is beneficial to introduce the sparsity and to select representations to transfer on both datasets.

Benefit of Transferring: Reducing Labelled Data

Transfer learning can reduce the labor and cost of labelling data instances. In this section, we quantify the benefit of knowledge transfer by comparing with non-transfer methods. That is, we gradually reduce the number of training examples in the target domain until the performance of the proposed model is inferior to the non-transfer MLP model. The more training examples we can reduce, the more benefit we can get from transferring knowledge.

Referring to Table 5.1, there are about 50 examples per user on the Mobile dataset. We gradually reduce one and two training examples per user, respectively, to investigate the benefit of knowledge transfer. The results are shown in Table 3.4 where the rows corresponding to reduction percentage 0% are copied from Table 3.3 for clarity. The number 2.05% is approximately corresponding to reducing one training example per user. The results show that we can save the cost of labelling about 30,000 training examples by transferring knowledge from the news domain but still have comparable performance with the MLP model, a non-transfer baseline.

According to Table 5.1, there are about 16 examples per user on the Amazon dataset. With a

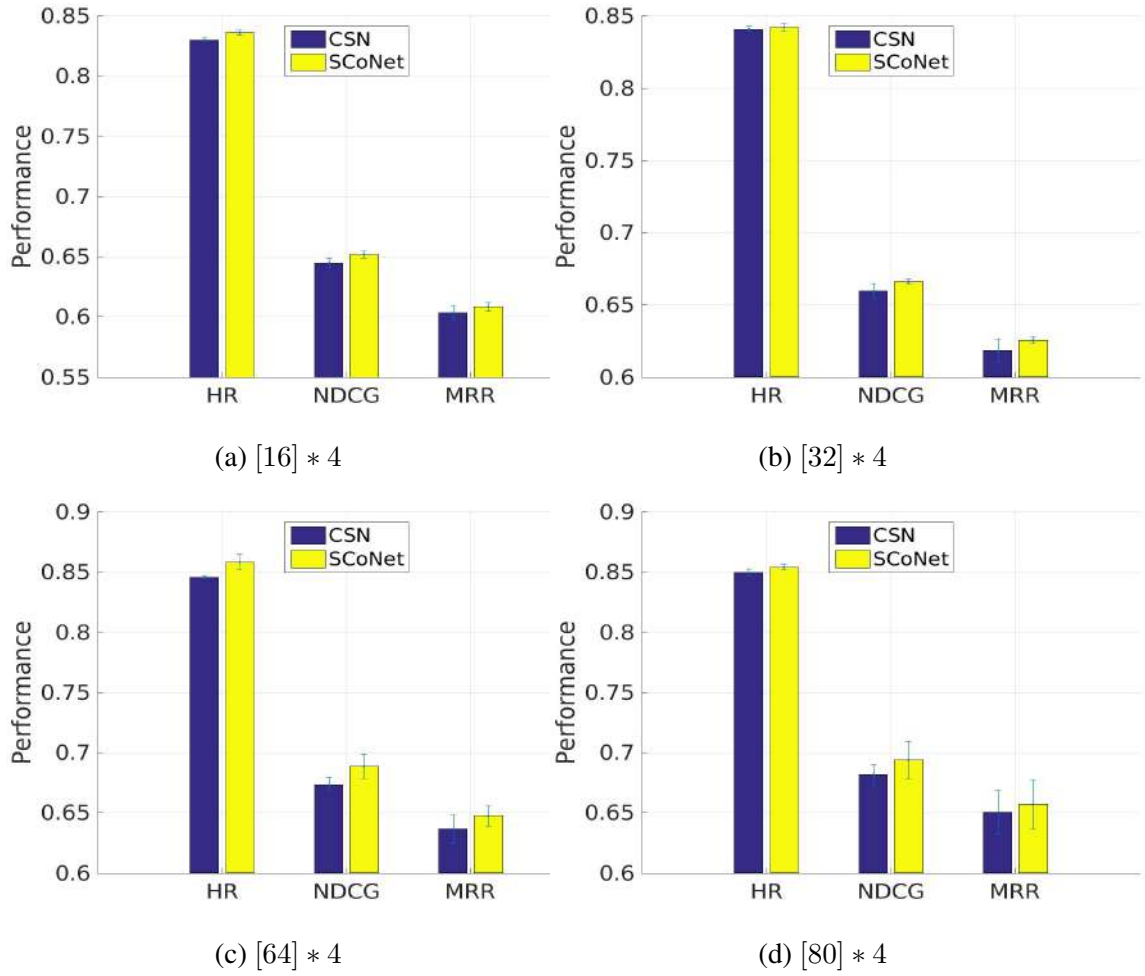


Figure 3.3: Impact of the sparsity on the Mobile dataset.

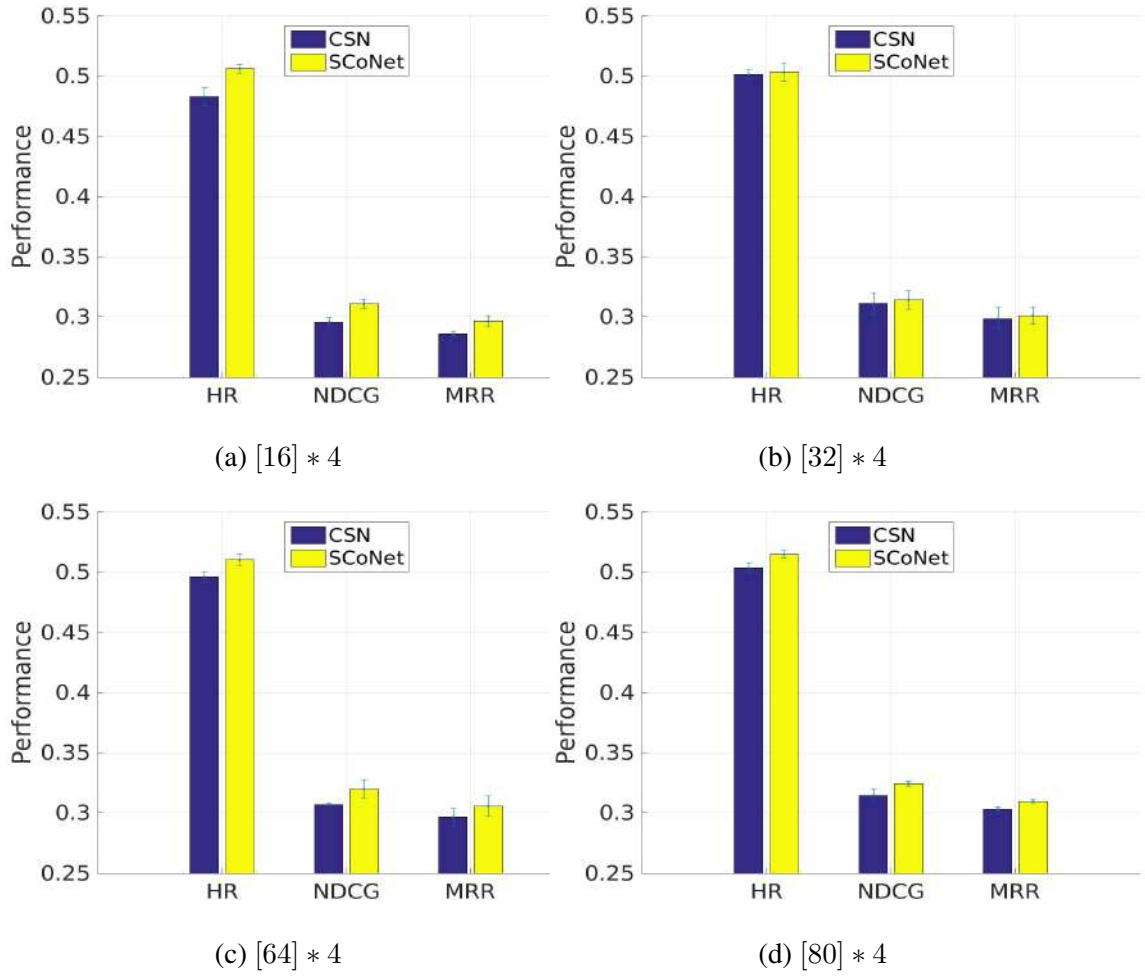


Figure 3.4: Impact of the sparsity on the Amazon dataset.

Table 3.4: The performance when reducing training examples. Results with stars are inferior to MLP.

Dataset	Method	Reduction		HR	NDCG	MRR
		percent	amount			
Mobile	MLP	0%	0	.8405	.6615	.6210
	SCoNet	0%	0	.8547	.6802	.6431
		2.05%	23,031	.8439	.6640	.6238
		4.06%	45,468	.8347*	.6515*	.6115*
Amazon	MLP	0%	0	.5014	.3143	.3113
	SCoNet	0%	0	.5338	.3424	.3351
		1.11%	12,850	.5110	.3209	.3080*
		2.18%	25,318	.4946*	.3082*	.2968*

similar setting to the Mobile dataset, the results shown in Table 3.4 indicates that we can save the cost of labelling about 20,000 training examples by transferring knowledge from movie domain. Note that the Amazon dataset is extremely sparse (the density is only 0.017%), implying that there is difficulty in acquiring many training examples. Under this scenario, our transfer models are an effective way of alleviating the issue of data sparsity and the cost of collecting data.

3.4.4 Analyses

Sensitivity Analysis on Sparsity Penalty Term

We analyze the sensitivity to the sparse penalty which controls the sparsity (λ in Eq.(3.9)). Results are shown on the Mobile data only due to space limit and we give the corresponding conclusions on the Amazon data. Figure 3.5a shows the performance varying with the penalty of sparsity enforcing on the task relationship matrices $\mathbf{H}$. On the Mobile data, the performance achieves good results at 0.1 (default) and 5.0, and it is 0.1 (default) and 1.0 on the Amazon data (not shown).

Sensitivity Analysis on Depth of Neural Networks

Table 3.5 and Table 3.6 show the performance varying with the depth of networks where results of MLP (the base model) are listed for reference. As we can see, stacking more layers are generally beneficial to performance improvements indicating the effectiveness of using neural networks for cross-domain recommendation. There is a significant improvement by using at least one hidden layers beyond the concatenated embedding module. In other words, user and item

Table 3.5: Performance Varying with Depth of Neural Models on Cheetah.

Depth (No. of hidden layers)	Metric	MLP	CoNet
[64]	HR	.7786	-
	NDCG	.5848	-
	MRR	.5240	-
[64→32]	HR	.8433	.8441
	NDCG	.6648	.6739
	MRR	.6080	.6340
[64→32→16]	HR	.8460	.8471
	NDCG	.6686	.6655
	MRR	.6121	.6224
[64→32→16→8]	HR	.8405	.8480
	NDCG	.6615	.6754
	MRR	.6210	.6373

Table 3.6: Performance Varying with Depth of Neural Models on Amazon.

Depth (No. of hidden layers)	Metric	MLP	CoNet
[64]	HR	.3636	-
	NDCG	.2277	-
	MRR	.1860	-
[64→32]	HR	.5030	.5248
	NDCG	.3114	.3280
	MRR	.2525	.3112
[64→32→16]	HR	.5123	.5154
	NDCG	.3182	.3246
	MRR	.2893	.3107
[64→32→16→8]	HR	.5014	.5167
	NDCG	.3143	.3261
	MRR	.3113	.3163

latent vectors alone are insufficient for modelling their feature interactions, and thus the necessity of transforming them with hidden layers. With knowledge transfer, our models benefit from deeper networks and do not saturate with four hidden layers while the base model (MLP) starts to overfitting. We do not try deeper networks since the network configure [64→32→16→8] can show the effectiveness of the proposed models and more deep hidden layers may face the risk of overfitting.

Optimization Performance

We analyze the optimization performance of SCoNet varying with training epochs. (One epoch means that the algorithm goes through the whole training dataset one time.) Results are shown

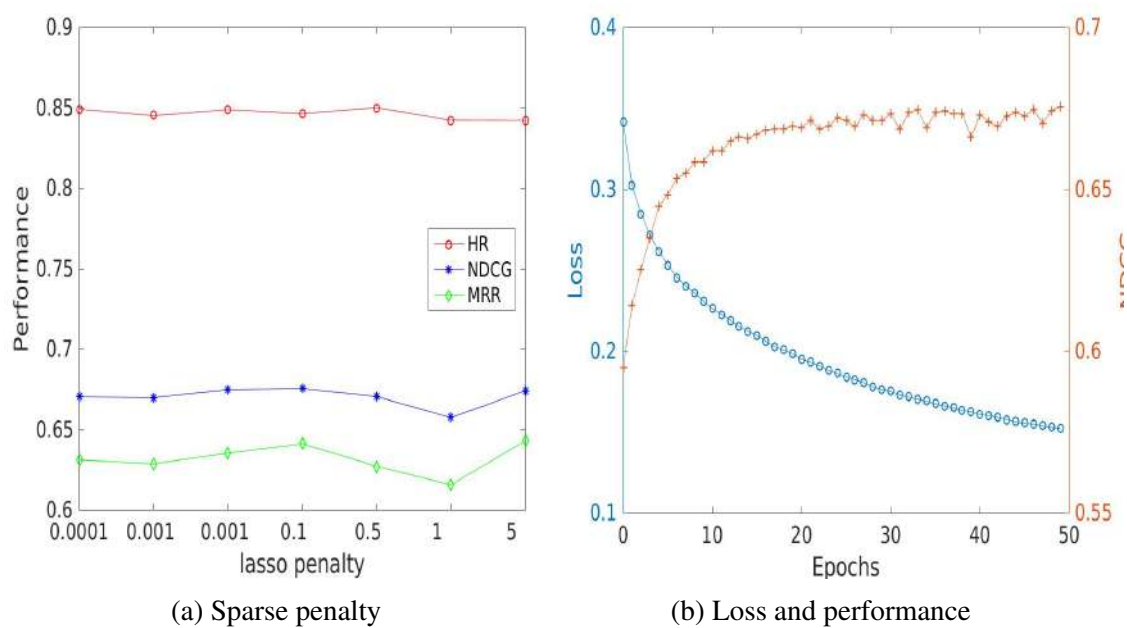


Figure 3.5: Sensitivity and Optimization

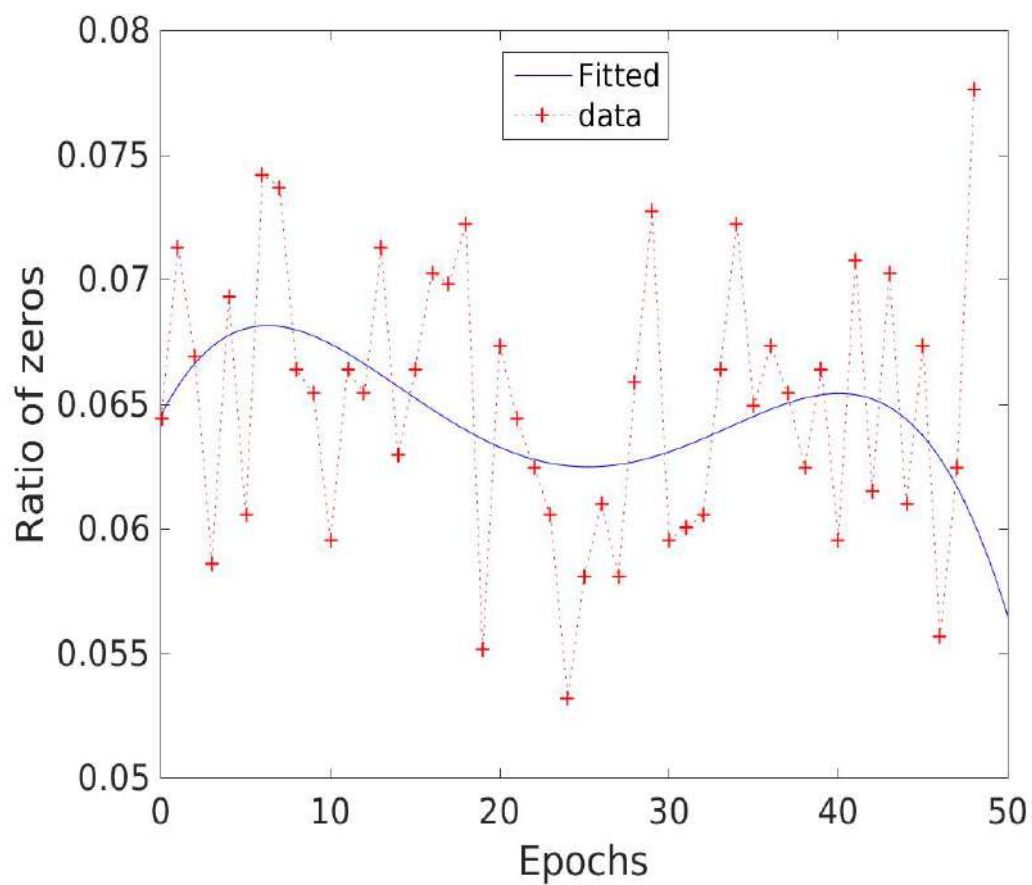


Figure 3.6: Ratio of zero entries

on the Mobile dataset only due to space limit and the trend on the Amazon dataset is similar.

We firstly show the training loss and test performance. Figure 3.5b shows the training loss (averaged/normalized over all training examples) and NDCG test performance on the test set (HR and MRR have similar trends) varying with each optimization iteration. We can see that with more iterations, the training losses gradually decrease and the recommendation performance is improved accordingly. The most effective updates are occurred in the first 15 iterations, and performance gradually improves until 30 iterations. With more iterations, SCoNet is relatively stable.

Since the sparsity of transfer matrices $(\mathbf{H}^l)_1^L$ is crucial to select representations for transferring, we show the change of zero entries over training epochs. For clarity and due to space limit, we only show the results of the first transfer matrix $\mathbf{H}^1$ which connects the first and the second hidden layers. Figure 3.6 shows the results where we use a 4-order polynomial to robustly fit the data. We can see that the matrix becomes sparser for the first 25 iterations, and the general trend is to sparsify. The average percent of zero entries in $\mathbf{H}^1$ is 6.5%. For the second and third transfer matrices, the percentage becomes 6.0% and 6.3%, respectively. Note that, the initial percent of zero entries of these three transfer matrices are 0.1%, 0.2%, and 0.0%, respectively. As we can see, the zero entries of the transfer matrices have a big increase during the optimization. In summary, sparse transfer matrices are learned and they can adaptively select partial representations to transfer across domains. And it may be better to *transfer many instead of all* representations at hand.

For the training time, our models spend about 100 seconds per epoch using one Nvidia TITAN Xp GPU. As a reference, it is 70s for MLP and 90s for CSN, which indicates that the training cost of the proposed method is comparable to (non-)transfer deep baselines.

3.5 Related Work

Recommender systems Recommender systems aim at learning user preferences on unknown items from their past history. Content-based recommendations are based on the matching between user profiles and item descriptions [98]. It is difficult to build the profile for each user when there is no/few content. Collaborative filtering (CF) alleviates this issue by predicting user preferences based on the user-item interaction behavior, agnostic to the content [20]. Latent factor models learn feature vectors for users and items mainly based on matrix factorization (MF) [59] which

has probabilistic interpretations [68, 84]. MF is also flexible to integrate text [45, 55], social relations [47, 132], and implicit feedback [46, 59]. Factorization machines can mimic MF [104]. Some hierarchical methods can reduce to factorize a specific matrix [120]. Random walk and heterogeneous networks are adapted for recommendation [110, 128]. Neural networks are proposed to push the learning of feature vectors towards non-linear representations [17, 23, 42]. CF models, however, suffer from the data sparsity issue.

Cross-domain recommendation [9] is an effective technique to alleviate sparse issue. A class of methods are based on MF applied to each domain, including collective MF (CMF) [111] with its heterogeneous variants [94] and codebook transfer [61, 63]. Active learning [145] can construct entity correspondence with limited budget. Heterogeneous cross-domain [134] and multiple source domains [74] are also proposed to account for different cases of input. These are all shallow methods and have the difficulty in learning complex (highly nonlinear) user-item interaction relationship [17, 42, 126]. We follow this research thread by using deep networks to learn the nonlinear interaction function.

Transfer and multitask learning Transfer learning (TL) aims at improving the performance of the target domain by exploiting knowledge from source domains [93]. The typical TL technique in neural networks is two-step: initialize a target network with transferred features from a pre-trained source network [90, 140]. Different from this approach, we transfer knowledge in a deep way such that two base networks benefit from each other during the learning procedure. Similar to TL, the multitask learning (MTL) is to leverage useful knowledge in multiple related tasks to help each other [10, 144]. Multi-view learning [24] is closely related to MTL. The cross-stitch network (CSN) [83] enables information sharing between two base networks. We generalize CSN by relaxing the underlying assumption, especially via the idea of selecting representations to transfer.

3.6 Conclusions

We proposed a novel deep transfer learning for cross-domain recommendation. The sparse target user-item interaction matrix can be reconstructed with the knowledge guidance from the source domain, alleviating the data sparse issue. We demonstrated the necessity of adaptively selecting representations from the auxiliary domain to transfer. It may harm the performance by transferring all of them with equal importance. We found that naive deep transfer models

may be inferior to the shallow/neural non-transfer methods in some cases. Our transfer model can reduce tens of thousands training examples by comparing with the non-transfer methods without performance degradation. This is useful when collecting data is difficult or costly. Experiments demonstrate the effectiveness of the proposed models on two large real-world datasets by comparing with shallow/deep, single/cross-domain methods. As a future work, we will integrate content information into the collaborative cross network for alleviating the cold-start problem.

Chapter 4

Deep Instance-based Knowledge Transfer in Recommendation

Collaborative filtering (CF) is the key technique for recommender systems (RSs). CF exploits user-item behavior interactions (e.g., clicks) only and hence suffers from the data sparsity issue. The research thread of transferring knowledge from auxiliary sources can improve the performance of target domain (e.g., movie recommendation) with the knowledge from the relevant source domain (e.g., book domain), leading to transfer learning methods. In real-world life, no single service can satisfy a user's all information needs. Thus it motivates us to exploit source information for RSs in this chapter. We propose a novel neural model to smoothly enable transfer meeting neural CF network method (dubbed as TransNet) for cross-domain recommendation in an end-to-end manner. TransNet selectively transfers knowledge from a source domain via a transfer network. On two real-world datasets, TransNet shows better performance in terms of three ranking metrics by comparing with various baselines. We conduct thorough analyses to understand how the transferred knowledge help the proposed model.

4.1 Introduction

Recommender systems are widely used in various domains and e-commerce platforms, such as to help consumers buy products at Amazon, watch videos on Youtube, and read articles on Google News. Collaborative filtering (CF) is among the most effective approaches based on the simple intuition that if users rated items similarly in the past then they are likely to rate items similarly in the future. Matrix factorization (MF) techniques which can learn the latent factors for users and items are its main cornerstone [59, 84]. Recently, neural networks like multilayer

perceptron (MLP) are used to learn the interaction function from data [23, 42]. MF and neural CF suffer from the data sparsity and cold-start issues.

One solution is to transfer the knowledge from relevant domains and the cross-domain recommendation techniques address such problems [9, 61, 94]. In real life, a user typically participates several systems to acquire different information services. For example, a user installs applications in an app store and reads news from a website at the same time. It brings us an opportunity to improve the recommendation performance in the target service (or all services) by learning across domains. Following the above example, we can represent the app installation feedback using a binary matrix where the entries indicate whether a user has installed an app. Similarly, we use another binary matrix to indicate whether a user has read a news article. Typically these two matrices are highly sparse, and it is beneficial to learn them simultaneously. This idea is sharpened into the collective matrix factorization (CMF) [111] approach which jointly factorizes these two matrices by sharing the user latent factors. It combines CF on a target domain and another CF on an auxiliary domain, enabling knowledge transfer [93, 144]. In terms of neural networks, given two activation maps from two tasks, cross-stitch convolutional network (CSN) [83] and its sparse variant [43] learn linear combinations of both the input activations and feed these combinations as input to the successive layers' filters, and hence enabling the knowledge transfer between two domains.

To transfer cross-domain knowledge, we propose a novel neural model, TransNet, for cross-domain recommendation in an end-to-end manner. TransNet can selectively transfer knowledge across domains by a transfer network (TNet), a novel network. A shared layer of feature interactions is stacked on the top to couple the high-level representations learned from individual networks. On real-world datasets, TransNet shows the better performance in terms of ranking metrics by comparing with various baselines. We conduct thorough analyses to understand how the transferred knowledge help TransNet.

Our contributions are summarized as follows:

- The proposed TransNet transfers the source domain using an attention mechanism which is trained in an end-to-end manner. It is the first deep model that transfers cross-domain knowledge for recommendation using attention based neural networks.
- The transfer component can selectively transfer source items with the guidance of target user-item interactions by the attentive weights. It is a novel transfer network for cross-domain recommendation.

- The proposed model can alleviate the sparsity issue including cold-user and cold-item start, and outperforms various baselines in terms of ranking metrics on two real-world datasets.

4.2 Problem Description

4.2.1 Problem Formulation

For collaborative filtering with implicit feedback, there is a binary matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$ to describe user-item interactions where each entry $r_{ui} \in \{0, 1\}$ is 1 (called observed entries) if user u has an interaction with item i and 0 (unobserved) otherwise:

$$r_{ui} = \begin{cases} 1, & \text{if user-item interaction } (u, i) \text{ exists;} \\ 0, & \text{otherwise.} \end{cases}$$

Denote the set of m -sized users by $\mathcal{U}$ and n items by $\mathcal{I}$. Usually the interaction matrix is very sparse since a user $u \in \mathcal{U}$ only consumed a very small subset of all items. Similarly for the task of item recommendation, each user is only interested in identifying top-K items. The items are ranked by their predicted scores:

$$\hat{r}_{ui} = f(u, i | \Theta), \quad (4.1)$$

where f is the interaction function and Θ denotes model parameters.

For MF-based CF approaches, the interaction function f is fixed and computed by a dot product between the user and item vectors. For neural CF, neural networks are used to parameterize function f and learn it from interaction data (see Section 4.2.2):

$$f(\mathbf{x}_{ui} | \mathbf{P}, \mathbf{Q}, \theta_f) = \phi_o(\dots(\phi_1(\mathbf{x}_{ui}))\dots), \quad (4.2)$$

where input $\mathbf{x}_{ui} = [\mathbf{P}^T \mathbf{x}_u, \mathbf{Q}^T \mathbf{x}_i] \in \mathbb{R}^{2d}$ is concatenated from that of user and item embeddings, which are projections of their one-hot encodings $\mathbf{x}_u \in \{0, 1\}^m$ and $\mathbf{x}_i \in \{0, 1\}^n$ by embedding matrices $\mathbf{P} \in \mathbb{R}^{m \times d}$ and $\mathbf{Q} \in \mathbb{R}^{n \times d}$, respectively. The output and hidden layers are computed by ϕ_o and $\{\phi_l\}$ in a neural network.

Similarly, for cross-domain recommendation, we have a target domain (e.g., news domain) user-item interaction matrix $\mathbf{R}_T \in \mathbb{R}^{m \times n_T}$ and a source domain (e.g., app domain) matrix $\mathbf{R}_S \in \mathbb{R}^{m \times n_S}$ where $m = |\mathcal{U}|$ and $n_T = |\mathcal{I}_T|$ ($n_S = |\mathcal{I}_S|$) is the size of users $\mathcal{U}$ and target items $\mathcal{I}_T$ (source items $\mathcal{I}_S$). Note that the users are shared and hence we can transfer knowledge across domains. We use u to index users, i to target items, and j to source items. Let $[j]^u = (j_1, j_2, \dots, j_s)$

be the s -sized source items that user u has interacted with in the source domain. Neural CF can be extended to leverage the source domain and then the interaction function has the form of (see Section 4.3.1):

$$f(u, i, [j]^u | \Theta). \quad (4.3)$$

For the task of item recommendation, the goal is to generate a ranked list of items for each user based on her history records, i.e., top-N recommendations. We hope improve the recommendation performance in the target domain with the help of both the content and source domain information.

A synthetic model of transfer and hybrid estimates the probability of his/her each observation conditioned on this user, the content text, and the interacted source items:

$$\hat{r}_{ui} \triangleq p(r_{ui} = 1 | u, d_{ui}, [j]^u). \quad (4.4)$$

The equation sharpens the intuition behind the synthetic model, that is, the conditional probability of whether user u will like the item i can be determined by three factors: his/her individual preferences, and his/her behavior in a related source domain ($[j]^u$). The likelihood function of the entire matrix $\mathbf{R}_T$ is then defined as:

$$p(\mathbf{R}_T) = \prod_u \prod_i p(r_{ui} | u, [j]^u). \quad (4.5)$$

We propose a novel neural model to learn the conditional probability in an end-to-end manner:

$$\hat{r}_{ui} = f(u, i, [j]^u | \Theta_f), \quad (4.6)$$

where f is the network function and Θ_f are model parameters.

The model consists of a transfer network $\mathbf{c}_{ui} = f_T(i, [j]^u | \Theta_T)$ to transfer knowledge from the source domain. A shared feature interaction layer $f_S(\mathbf{z}_{ui}, \mathbf{c}_{ui} | \Theta_S)$ where $\mathbf{z}_{ui}$ is the non-linear representations of the (u, i) interaction, is stacked on the top of the learned high-level representations from individual networks.

4.2.2 A Basic Neural CF Network

We adopt a feedforward neural network (FFNN) as the base neural CF model to parameterize the interaction function (see Eq. (4.2)). The basic network is similar to the Deep model in [15, 17] and the MLP model in [42]. The base network consists of four modules with the information flow from the input (u, i) to the output $\hat{r}_{ui}$ as follows.

Table 4.1: Model Parameters of TransNet.

Parameter	Dimension	Description
$\mathbf{P}$	$m \times d$	User embedding matrix
$\mathbf{Q}$	$n_T \times d$	Target item embedding matrix
$\mathbf{H}$	$n_S \times d$	Source item embedding matrix
$\mathbf{A}$	$L \times 2d$	Internal memory matrix
$\mathbf{C}$	$L \times 2d$	External memory matrix
$\mathbf{W}, \mathbf{b}$	$2d \times d, d$	Linear mapping weight and bias for the user-item interaction
$\mathbf{W}_z, \mathbf{W}_c$	$d \times d$	Linear mapping for outputs of individual networks
$\mathbf{h}$	$2d$	Weight of the shared layer

Input: $(u, i) \rightarrow \mathbb{1}_u, \mathbb{1}_i$

This module encodes user-item interaction indices. We adopt the one-hot encoding. It takes user u and item i , and maps them into one-hot encodings $\mathbb{1}_u \in \{0, 1\}^m$ and $\mathbb{1}_i \in \{0, 1\}^n$ where only the element corresponding to that index is 1 and all others are 0.

Embedding: $\mathbb{1}_u, \mathbb{1}_i \rightarrow \mathbf{x}_{ui}$

This module firstly embeds one-hot encodings into continuous representations $\mathbf{x}_u = \mathbf{P}^T \mathbb{1}_u$ and $\mathbf{x}_i = \mathbf{Q}^T \mathbb{1}_i$ by embedding matrices $\mathbf{P}$ and $\mathbf{Q}$ respectively, and then concatenates them as $\mathbf{x}_{ui} = [\mathbf{x}_u, \mathbf{x}_i]$, to be the input of following building blocks.

Hidden layers: $\mathbf{x}_{ui} \rightsquigarrow \mathbf{z}_{ui}$

This module takes the continuous representations from the embedding module and then transforms through several layers to a final latent representation $\mathbf{z}_{ui} = (\dots(\phi_1(\mathbf{x}_{ui})\dots))$. This module consists of hidden layers to learn nonlinear interaction between users and items.

Output: $\mathbf{z}_{ui} \rightarrow \hat{r}_{ui}$

This module predicts the score $\hat{r}_{ui}$ for the given user-item pair based on the representation $\mathbf{z}_{ui}$ from the last layer of multi-hop module. Since we focus on one-class collaborative filtering, the output is the probability that the input pair is a positive interaction. This can be achieved by a softmax layer: $\hat{r}_{ui} = \phi_o(\mathbf{z}_{ui}) = \frac{1}{1 + \exp(-\mathbf{h}^T \mathbf{z}_{ui})}$, where $\mathbf{h}$ is the parameter.

4.3 Methodology

We describe the proposed TransNet model in this section. TransNet models user preferences in the target domain by transferring knowledge from a source/auxiliary domain. TransNet learns high-level representations for source domain items such that the learned representations can estimate the conditional probability of that whether a user will like an item. This is done with a transfer network (Sec. 4.3.1), coupled by the shared embeddings on the bottom and an interaction layer on the top of it. The entire network can be trained efficiently to minimize a binary cross-entropy loss by back-propagation. We begin by describing the recommendation problem and the model formulation before introducing the network architecture.

4.3.1 TransNet

Selecting Source Items to Transfer

We introduce a transfer component to exploit the source domain knowledge. A user may participate several systems to acquire different information needs, for example, a user installs apps in an app store and reads news from other website. Cross-domain recommendation [9] is an effective technique to alleviate sparse issue where transfer learning (including multitask learning) [10, 93, 144] is a class of underlying methods. Typical methods include collective matrix factorization (CMF) [111] approach which jointly factorizes two rating matrices by sharing the user latent factors and hence it enables knowledge transfer. The cross-stitch network [83] and its sparse variant [43] enable information sharing between two base networks for each domain in a deep way. These methods treat knowledge transfer as a global process (shared global parameters) and do not match source items with the specific target item given a user.

We propose a novel transfer network (TNet) which can selectively transfer source knowledge for specific target item. Since the relationships between items are shown to be important in improving recommendation performance [58, 86, 88, 97] for single domain, *we want to capture relationships between target item and source items of a user. The central idea is to learn adaptive weights over source items specific to the given target item during the knowledge transfer.*

Given the source items $[j]^u = (j_1, j_2, \dots, j_s)$ with which the user u has interacted in the source domain, TNet learns a transfer vector $c_{ui} \in \mathbb{R}^d$ to capture the relations between the target item i and source items given the user u . The underlying observations can be illustrated in an example of improving the movie recommendation by transferring knowledge from the book

domain. When we predict the preference of a user on the movie “The Lord of the Rings,” the importance of her read books such as “The Hobbit,” and “The Silmarillion” may be much higher than those such as “Call Me by Your Name”.

The similarities between target item i and source items can be computed by their dot products:

$$a_j^{(i)} = \mathbf{x}_i^T \mathbf{x}_j, \quad j = 1, \dots, s, \quad (4.7)$$

where $\mathbf{x}_j \in \mathbb{R}^d$ is the embedding for the source item j by an embedding matrix $\mathbf{H} \in \mathbb{R}^{n_s \times d}$. This score computes the compatibility between the target item and the source items consumed by the user. For example, the similarity of target movie $i = \text{“The Lord of the Rings,”}$ with the source book $j = \text{“The Hobbit”}$ may be larger than that with the source book $j' = \text{“Call Me by Your Name”}$ (given a user u).

We normalize similarity scores to be a probability distribution over source items:

$$\alpha_j^{(i)} = \text{softmax}(a_j^{(i)}), \quad (4.8)$$

and then the transfer vector is a weighted sum of the corresponding source item embeddings:

$$\mathbf{c}_{ui} = \text{ReLU}\left(\sum_j \alpha_j^{(i)} \mathbf{x}_j\right), \quad (4.9)$$

where we introduce non-linearity on the transfer vector by activation function rectified linear unit (ReLU). Empirically we found that the activation function $\text{ReLU}(x) = \max(0, x)$ works well due to its non-saturating nature and suitability for sparse data. The transfer vector $\mathbf{c}_{ui}$ is a high-level representation, summarizing the knowledge from the source domain as the output of the TNet. TNet can selectively transfer representations from the corresponding embeddings of source items with the guidance of the target user-item interaction.

A more detailed discussion on Eq. (4.9) is in order. Eq. (4.9) sharpens the idea that the transfer component can selectively transfer source items with the guidance of target user-item interactions. This is achieved by attentive weights $\alpha_j^{(i)}$ (or $a_j^{(i)}$). When the source item j is highly relevant to the target item i given user u , then the knowledge from the source domain is easily flowing into the target domain with a high influence weight. When the source item j is irrelevant to the target item i given user u , then the knowledge from source domain is hard to flow into the target domain with a small effect weight. This selection is automatically determined by the transfer component, but this is not easily achieved by the existing cross-domain recommendation techniques like the multitask models such as collective matrix factorization [111] and collaborative cross networks [43] (a variant of cross-stitch networks [83]) which have multi-objective

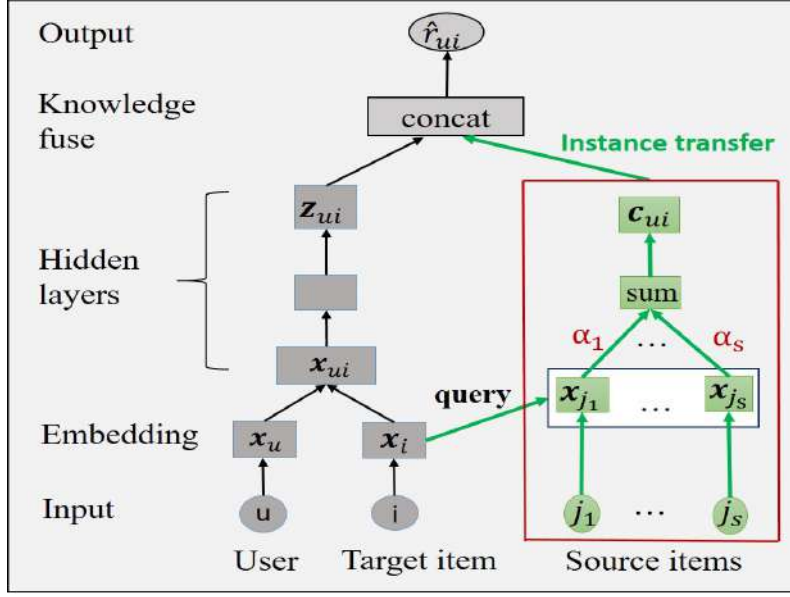


Figure 4.1: Architecture of the Proposed TransNet Model. The instance transfer component (rendered in green color) infers a variable-length transfer weight vector $\alpha = [\alpha_1, \dots, \alpha_s]$ based on the current target item i and source items $[j]^u = [j_1, \dots, j_s]$. A transfer vector c_{ui} is then computed as the weighted average, according to the weight vector α , over the source items.

optimization. Besides, we implicitly use the label information from the source domain when generating the source items for a user, while CMF and CSN explicitly exploit label information by learning to predict the labels. As a result, the transfer component benefits from the source domain knowledge in two-step: selecting instances (source items) to transfer via source domain labels and re-weighting instances with attentive weights.

Put Them Together

The architecture for the proposed TransNet model is illustrated in Figure 4.1 as a feedforward neural network (FFNN). The input layer specifies embeddings of a user u , a target item i , and the corresponding source items $[j]^u = [j_1, \dots, j_s]$. The content text d_{ui} is modelled by the memories in the MNet to produce a high-level representation o_{ui} . The source items are transferred into the transfer vector c_{ui} with the guidance of (u, i) in the TNet.

Firstly, we use a simple neural CF model (CFNet) which has one hidden layer to learn a nonlinear representation for the user-item interaction:

$$z_{ui} = \text{ReLU}(\mathbf{W}x_{ui} + \mathbf{b}), \quad (4.10)$$

where $\mathbf{W}$ and $\mathbf{b}$ are the weight and bias parameters in the hidden layer. Usually the dimension of z_{ui} is half of that x_{ui} in a typical tower-pattern architecture.

The outputs from the three individual networks can be viewed high-level features of the source domain knowledge and the user-item interaction. They come from different feature space learned by different networks. Thus, we use a shared layer on the top of the all features:

$$\hat{r}_{ui} = \frac{1}{1 + \exp(-\mathbf{h}^T \mathbf{y}_{ui})}, \quad (4.11)$$

where $\mathbf{h}$ is the parameter. And the joint representation:

$$\mathbf{y}_{ui} = [\mathbf{W}_z \mathbf{z}_{ui}, \mathbf{W}_c \mathbf{c}_{ui}], \quad (4.12)$$

is concatenated from the linear mapped outputs of individual networks where matrices $\mathbf{W}_z, \mathbf{W}_c$ are the corresponding linear mapping transformations..

4.3.2 Objective and Optimization

Due to the nature of the implicit feedback and the task of item recommendation, the squared loss $(\hat{r}_{ui} - r_{ui})^2$ may be not suitable since it is usually for rating prediction. Instead, we adopt the binary cross-entropy loss:

$$\mathcal{L} = - \sum_{(u,i) \in \mathcal{S}} r_{ui} \log \hat{r}_{ui} + (1 - r_{ui}) \log(1 - \hat{r}_{ui}), \quad (4.13)$$

where the training samples $\mathcal{S} = \mathbf{R}_T^+ \cup \mathbf{R}_T^-$ are the union of observed target interaction matrix and randomly sampled negative pairs. Usually, $|\mathbf{R}_T^+| = |\mathbf{R}_T^-|$ and we do not perform a predefined negative sampling in advance since this can only generate a fixed training set of negative samples. Instead, we generate negative samples during each epoch, enabling diverse and augmented training sets of negative examples to be used.

This objective function has a probabilistic interpretation and is the negative logarithm likelihood of the following likelihood function:

$$L(\Theta|\mathcal{S}) = \prod_{(u,i) \in \mathbf{R}_T^+} \hat{r}_{ui} \prod_{(u,i) \in \mathbf{R}_T^-} (1 - \hat{r}_{ui}), \quad (4.14)$$

where the model parameters are: $\Theta = \{\mathbf{P}, \mathbf{Q}, \mathbf{H}, \mathbf{A}, \mathbf{C}, \mathbf{W}, \mathbf{b}, \mathbf{W}_z, \mathbf{W}_c, \mathbf{h}\}$. Comparing with Eq. (5.3), instead of modeling all zero entries (i.e., the whole target matrix $\mathbf{R}_T$), we learn from only a small subset of such unobserved entries and treat them as negative samples by picking them randomly during each optimization iteration (i.e., the negative sampling technique). The objective function can be optimized by stochastic gradient descent (SGD) and its variants like adaptive moment method (Adam) [56]. The update equations are:

$$\Theta^{new} \leftarrow \Theta^{old} - \eta \frac{\partial L(\Theta)}{\partial \Theta}, \quad (4.15)$$

where η is the learning rate. Typical deep learning library like TensorFlow¹ provides automatic differentiation and hence we omit the gradient equations $\frac{\partial L(\Theta)}{\partial \Theta}$ which can be computed by chain rule in back-propagation (BP).

4.3.3 Complexity Analysis

In the model parameters Θ , the embedding matrices P , Q and H contain a large number of parameters since they depend on the input size of users and (target and source) items, and their scale is hundreds of thousands. Since the architecture follows a tower pattern, the dimension of the outputs of the individual networks is also limited within hundreds. In total, the size of model parameters is linear with the input size and is close to the size of typical latent factors models [111] and one hidden layer neural CF approaches [42].

During training, we compute the outputs of the three individual networks in parallel using mini-batch stochastic optimization which can be trained efficiently by back-propagation. TransNet is scalable to the number of the training data. It can easily update when new data examples come, just feeding them into the training mini-batch. Thus, TransNet can handle the scalability and dynamics of items and users like in an online fashion. In contrast, the topic modeling related techniques have difficulty in benefitting from these advantages to this extent.

4.4 Experiments

In this section, we conduct empirical study to answer the following questions: 1) how does the proposed TransNet model perform compared with state-of-the-art recommender systems; and 2) how do the source domain information contribute each to the proposed framework. We firstly introduce the evaluation protocols and experimental settings, and then we compare the performance of different recommender systems. We further analyze the TransNet model to understand the impact of the transfer component. We also investigate that the improved performance comes from the cold-users and cold-items to some extent.

Table 4.2: Datasets and Statistics.

Dataset	Domain	Statistics	Amount
Mobile News	Shared	#Users	15,890
	Target	#News	84,802
		#Reads	477,685
		Density	0.035%
	Source	#Apps	14,340
		#Installations Density	817,120 0.359%
Amazon Product	Shared	#Users	8,514
	Target	#Clothes (Men)	28,262
		#Ratings/ #Reviews	56,050
		Density	0.023%
	Source	#Products (Sports)	41,317
		#Ratings/ #Reviews Density	81,924 0.023%

4.4.1 Data Sets and Evaluation Protocol

Data Sets

We evaluate on two real-world cross-domain datasets. The first dataset, **Mobile**², is provided by a large internet company, i.e., Cheetah Mobile³ [69]. The information contains logs of user reading news, the history of app installation, and some metadata such as news publisher and user gender collected in one month in the US. We removed users with fewer than 10 feedbacks. The dataset we used contains 477K user-news reading records and 817K user-app installations. There are 15.8K shared users which enable the knowledge transfer between the two domains. We aim to improve the news recommendation by transferring knowledge from app domain. The data sparsity is over 99.6%.

The second dataset is a public **Amazon** dataset⁴, which has been widely used to evaluate the performance of collaborative filtering approaches [41]. We use the two categories of Amazon Men and Amazon Sports as the cross-domain [41, 45]. The original ratings are from 1 to 5 where five stars indicate that the user shows a positive preference on the item while the one stars are not. We convert the ratings of 4-5 as positive samples. The dataset we used contains 56K positive ratings on Amazon Men and 81K positive ratings on Amazon Sports. There are 8.5K shared

¹<https://www.tensorflow.org>

²An anonymous version can be released later.

³<http://www.cmcm.com/en-us/>

⁴<http://snap.stanford.edu/data/web-Amazon.html>

users, 28K Men products, and 41K Sports goods. We aim to improve the recommendation on the Men domain by transferring knowledge from relevant Sports domain. The data sparsity is over 99.7%.

The statistics of the two datasets are summarized in Table 6.1. As we can see, both datasets are very sparse and hence we hope improve performance by transferring knowledge from the auxiliary domain as well.

Evaluation Protocol

For item recommendation task, the leave-one-out (LOO) evaluation is widely used and we follow the protocol in [42]. That is, we reserve one interaction as the test item for each user. We determine hyper-parameters by randomly sampling another interaction per user as the validation/development set. We follow the common strategy which randomly samples 99 (negative) items that are not interacted by the user and then evaluate how well the recommender can rank the test item against these negative ones. Since we aim at top-K item recommendation, the typical evaluation metrics are hit ratio (HR), normalized discounted cumulative gain (NDCG), and mean reciprocal rank (MRR), where the ranked list is cut off at $topK = \{5, 10, 20\}$. HR intuitively measures whether the reserved test item is present on the top-K list, defined as:

$$HR = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \delta(p_u \leq topK), \quad (4.16)$$

where p_u is the hit position for the test item of user u , and $\delta(\cdot)$ is the indicator function. NDCG and MRR also account for the rank of the hit position, respectively defined as:

$$NDCG = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{\log 2}{\log(p_u + 1)}, \quad (4.17)$$

and

$$MRR = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{p_u}. \quad (4.18)$$

A higher value with lower cutoff indicates better performance.

Implementation

For BPRMF, we use LightFM’s implementation⁵ which is a popular CF library. For CDCF, we adapt the official libFM implementation⁶. For CMF, we use a Python version reference to the

⁵<https://github.com/lyst/lightfm>

⁶<http://www.libfm.org>

Baselines	Shallow method	Deep method
Single-Domain	BPRMF [105]	MLP [42]
Cross-Domain	CDCF [71], CMF [111]	MLP++, CSN [83], TransNet (ours)

Table 4.3: Categorization of Baselines

original Matlab code⁷. For latent factor models, we vary the number of factors from 10 to 100 with step size 10. For MLP, we use the code released by its authors⁸. The MLP++ and CSN are implemented based on MLP. Our methods are implemented using TensorFlow. Parameters are randomly initialized from Gaussian $\mathcal{N}(0, 0.01^2)$. The optimizer is Adam with initial learning rate 0.001. The size of mini batch is 128. The ratio of negative sampling is 1. The MLP and MLP++ follows a tower pattern, halving the layer size for each successive higher layer. Specifically, the configuration of hidden layers in the base MLP network is $[64 \rightarrow 32 \rightarrow 16 \rightarrow 8]$ as reference in the original paper [42]. For CSN, it requires that the number of neurons in each hidden layer is the same and the configuration is $[64] * 4$ (equals $[64 \rightarrow 64 \rightarrow 64 \rightarrow 64]$). We investigate several typical configurations $\{16, 32, 64, 80\} * 4$. The dimension of embeddings is $d = 75$.

4.4.2 Baselines

We compare with various baselines, categorized as single/cross domain, shallow/deep, and hybrid methods.

- **BPRMF**, Bayesian personalized ranking [105], is a latent factor model based on matrix factorization and pair-wise loss. It learns on the target domain only.
- **HFT**, Hidden Factors and hidden Topics [79], adopts topic distributions to learn latent factors from text reviews. It is a hybrid method.
- **CDCF**, Cross-domain CF with factorization machines (FM) [71], is a cross-domain recommender which extends FM [104]. It is a context-aware approach which applies factorization on the merged domains (aligned by the shared users). That is, the auxiliary domain is used as context. On the Mobile dataset, the context for a user in the target news domain is his/her history of app installations in the source app domain. The feature vector for the input is a sparse vector $\mathbf{x} \in \mathbb{R}^{m+n_T+n_S}$ where the non-zero entries are as follows:

⁷<http://www.cs.cmu.edu/~ajit/cmf/>

⁸https://github.com/hexiangnan/neural_collaborative_filtering

1) the index for user id, 2) the index for target news id (target domain), and all indices for his/her installed apps (source domain).

- **CMF**, Collective matrix factorization [111], is a multi-relation learning approach which jointly factorizes matrices of individual domains. Here, the relation is the user-item interaction. On Mobile, the two matrices are A = “user by news” and B = “user by app” respectively. The shared user factors P enable knowledge transfer between two domains. Then CMF factorizes matrices A and B simultaneously by sharing the user latent factors: $A \approx P^T Q_A$ and $B \approx P^T Q_B$. It is a shallow model and jointly learns on two domains. CMF is a multi-objective shallow model for cross-domain recommendation. This can be thought of a non-deep transfer/multitask learning approach for cross-domain recommendation.
- **MLP**, multilayer perceptron [42], is a neural CF approach which learns the nonlinear interaction function using neural networks. It is a deep model learning on the target domain only.
- **MLP++**: We combine two MLPs by sharing the user embedding matrix, enabling the knowledge transfer between two domains through the shared users. It is a naive knowledge transfer approach applied for cross-domain recommendation.
- **CSN**, Cross-stitch network [83], is a deep multitask learning model originally proposed for visual recognition tasks. We use the cross-stitch units to stitch two MLP networks. It learns a linear combination of activation maps from two networks and hence benefits from each other. Comparing with MLP++, CSN enables knowledge transfer also in the hidden layers besides the lower embedding matrices. CSN optimizes a multi-objective problem for cross-domain recommendation. This is a deep transfer learning approach for cross-domain recommendation.

4.4.3 Results

In this section, we report the recommendation performance of different methods and discuss the findings. The comparison results are shown in Table 4.5 and Table 4.4 respectively on the Mobile and Amazon datasets where the last row is the relative improvement of ours vs the best baseline. We have the following observations. Firstly, we can see that our proposed neural models are

Table 4.4: Results on Amazon data. The last row is the relative improvement of TransNet over the best baseline.

Method	HR@5	NDCG@5	HR@10	NDCG@10
BPRMF	.0810	.0583	.1204	.0710
MLP	.2100	.1486	.2836	.1697
CDCF	.1295	.0920	.2070	.1167
CMF	.1498	.0950	.2224	.1182
MLP++	.2263	.1626	.2992	.1862
CSN	.2340*	.1680**	.3018*	.1898*
TransNet	.2455	.1703	.3293	.1974
Improvement	4.91%	1.37%	9.11%	4.00%

Table 4.5: Results on Mobile data. The last row is the relative improvement of TransNet over the best baseline.

Method	HR@5	NDCG@5	HR@10	NDCG@10
BPRMF	.4380	.3971	.4941	.4182
MLP	.5380	.4121	.6176	.4381
CMF	.4789	.3535	.5846	.3879
CDCF	.5066	.3734	.5325	.4089
MLP++	.5524	.4284	.6319	.4535
CSN	.5551*	.4323* *	.6327*	.4574*
TransNet	.5616	.4382	.6408	.4629
Improvement	1.17%	1.36%	1.28%	1.20%

better than all baselines on the two datasets at each setting, including the base MLP network, shallow cross-domain models (CMF and CDCF), and deep cross-domain models (MLP++ and CSN). These results demonstrate the effectiveness of the proposed neural model.

On the Mobile dataset, the differences between TransNet and other methods are more pronounced for small numbers of recommended items including top-5 or top-10 where we achieve average 1.25% relative improvements over the best baseline. This is a desirable feature since we often recommend only a small number of top ranked items to consumers to alleviate the information overload issue.

Note that the relative improvement of the proposed model vs. the best baseline is more significant on the Amazon dataset than that on the Mobile dataset, obtaining average 4.85% relative improvements over the best CSN baseline, though the Amazon is sparser than the Mobile (see Table 6.1). We show the benefit of combining text content by comparing with CSN. One

explanation is that the relatedness of the Men and Sports domains is closer than that between the news and app domains. This will benefit all cross-domain methods including CMF, CDCF, MLP++, and CSN, since they exploit information from both two domains.

There is a possibility that the noise from auxiliary domain proposes a challenge for exploiting them. This shows that the proposed model is more effective since it can select useful representations from the source network.

In summary, the empirical comparison results demonstrate the superiority of the proposed neural model to exploit the text content and source domain knowledge for recommendation.

Improvements of Transfer Learning on Different Datasets

It is noted that the benefit of transfer learning on the Amazon dataset is higher than that on the Cheetah Mobile dataset where the former has a relative 4.8% improvement while the latter has a relative 1.3% improvement. One possible reason that the scenario on the Amazon dataset is a cross-domain recommendation case while the scenario on the Cheetah Mobile is a cross-dataset recommendation case. Intuitively, the relatedness between the source and target domains is closer in the former case (i.e., from the Sports category to the Clothes category) than that in the latter case (i.e., from the Apps installation to the News reading). A theory using distance between distributions is proposed in the domain adaption setting [6].

4.4.4 Analyses

Improvement on Cold Users and Items

The cold-user and cold-item problems are common issues in recommender systems. When new users enter into a system, they have no history that can be exploited by the recommender system to learn their preferences, leading to the cold-user start problem. Similarly, when latest news are released on the Google News, there are no reading records that can be exploited by the recommender system to learn users' preferences on them, leading to the cold-item start problem. In general, it is very hard to train a reliable recommender system and make predictions for users and items that have few interactions. Intuitively, the proposed model can alleviate both the cold-user and cold-item start issues. TransNet alleviates the cold-user start issue in the target domain by transferring his/her history from the related source domain. TransNet alleviates the cold-item start issue by exploiting the associated text content to reveal its properties, semantics,

and topics. We now investigate that TransNet indeed improves the performance over the cold users and items by comparing with the pure neural collaborative filtering method, MLP.

We analyse the distribution of missed hit users (MHUs) of TransNet and MLP (at cutoff 10). We expect that the cold users in MHUs of MLP can be reduced by using the TransNet model. The more amount we can reduce, the more effective that TransNet can alleviate the cold-user start issues. The results are shown in Figure 4.2 where the number of training examples can measure the “coldness” of a user. Naturally, the MHUs are most of the cold users who have few training examples. As we can see, the number of cold users in MHUs of MLP is higher than that of TransNet. If the cold users are defined as those with less than seven training examples, then TransNet reduces the number of cold users from 4,218 to 3,746 on the Amazon dataset, achieving relative 12.1% reduction. On the Mobile dataset, if the cold users are those with less than ten training examples (Mobile is denser than Amazon), then TransNet reduces the number of cold users from 1,385 to 1,145 on the Mobile dataset, achieving relative 20.9% reduction. These results show that the proposed model is effective in alleviating the cold-user start issue. The results on cold items are similar and we omit them due to the page limit.

4.5 Related Work

Collaborative filtering Recommender systems aim at learning user preferences on unknown items from their past history. Content-based recommendations are based on the matching between user profiles and item descriptions. It is difficult to build the profile for each user when there is no/few content. Collaborative filtering (CF) alleviates this issue by predicting user preferences based on the user-item interaction behavior, agnostic to the content [20]. Latent factor models learn feature vectors for users and items mainly based on matrix factorization (MF) [59] which has probabilistic interpretations [84]. Factorization machines (FM) can mimic MF [104]. To address the data sparsity, an item-item matrix (SPPM) is constructed from the user-item interaction matrix in the CoFactor model [67]. It then simultaneously factorizes the interaction matrix and the SPPMI matrix in a shared item latent space, enabling the usage of co-click information to regularize the learning of user-item matrix. In contrast with our method, We use independent unstructured text and source domain information to alleviate the data sparsity issue in the user-item matrix.

Neural networks are proposed to push the learning of feature vectors towards non-linear

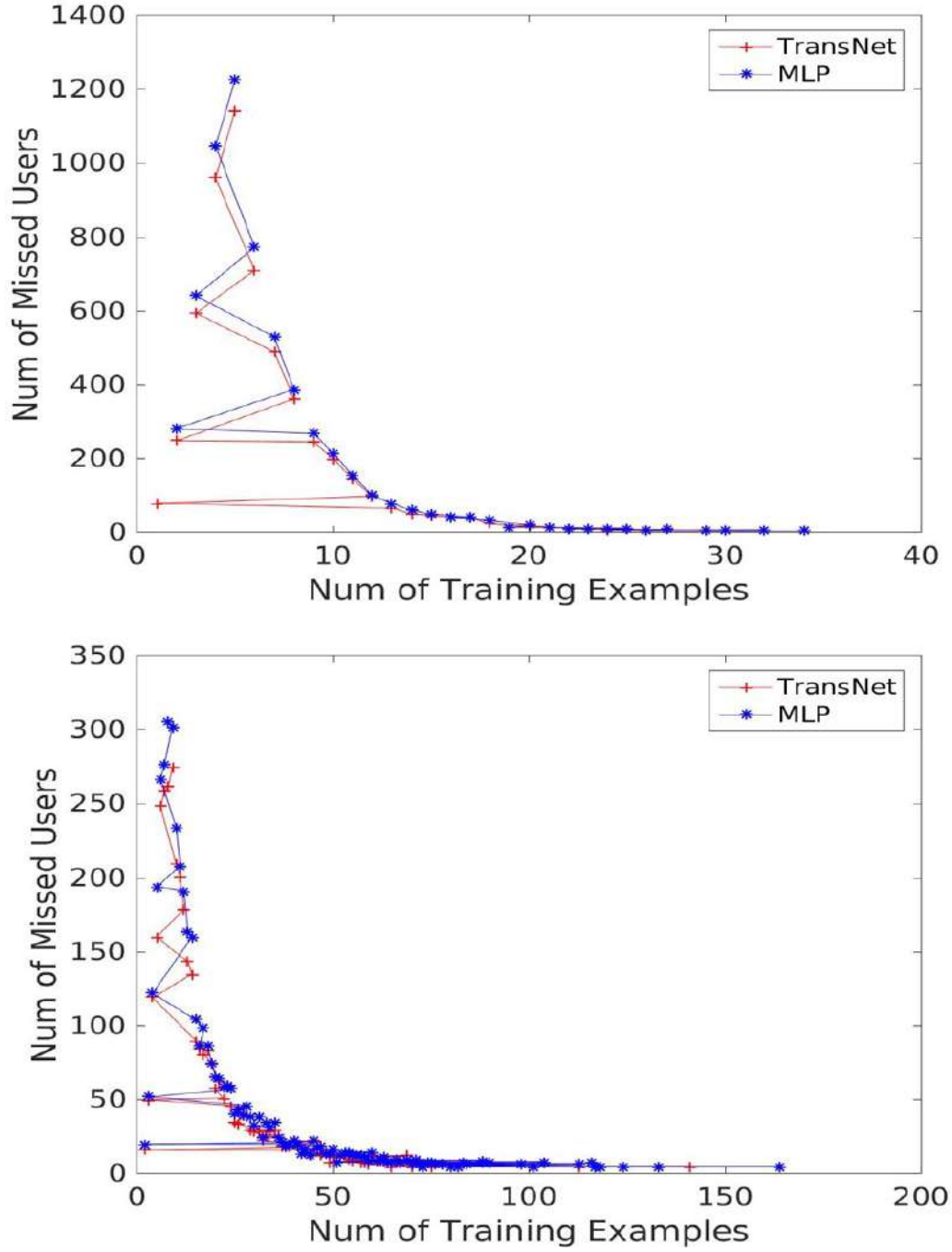


Figure 4.2: The Missed Hit Users Distribution (not normalized) Over the Number of Training Examples on the Amazon (Top) Datasets and on the Mobile (Bottom).

representations, including the neural network matrix factorization (NNMF) and multilayer perceptron (MLP) [23, 42]. The basic MLP architecture is extended to regularize the factors of users and items by social and geographical information [132]. Other neural approaches learn from the explicit feedback for rating prediction task [11, 147]. We focus on learning from the implicit feedback for top-N recommendation [127]. CF models, however, suffer from the data sparsity issue.

Cross-domain recommendation Cross-domain recommendation [9] is an effective technique to alleviate sparse issue. A class of methods are based on MF applied to each domain. Typical methods include collective matrix factorization (CMF) [111] approach which jointly factorizes two rating matrices by sharing the user latent factors and hence it enables knowledge transfer. CMF has its heterogeneous [94] variants, and codebook transfer [61]. The coordinate system transfer can exploit heterogeneous feedbacks [95, 134]. Multiple source domains [74] and multi-view learning [24] are also proposed for integrating information from several domains. Transfer learning (TL) aims at improving the performance of the target domain by exploiting knowledge from source domains [93]. Similar to TL, the multitask learning (MTL) is to leverage useful knowledge in multiple related tasks to help each other [10, 144]. The cross-stitch network [83] and its sparse variant [43] enable information sharing between two base networks for each domain in a deep way. Robust learning is also considered during knowledge transfer [40]. These methods treat knowledge transfer as a global process with shared global parameters and do not match source items with the specific target item given a user. We follow this research thread by using neural networks to selectively transfer knowledge from the source items. We introduce a transfer component to exploit the source domain knowledge.

4.6 Conclusions

It is shown that the source domain knowledge can help improve recommendation performance and can be effectively integrated under a neural architecture. The sparse target user-item interaction matrix can be reconstructed with the knowledge guidance from the source information, alleviating the data sparse issue. We proposed a novel deep neural model, TransNet, for cross-domain recommendation. TransNet smoothly enables transfer meeting neural CF. TransNet consists of a transfer component which can selectively transfer useful source items to benefit the target domain. These are achieved by the attentive weights learned automatically. TransNet shows

better performance than various baselines on two real-world datasets under different settings. The results demonstrate that our combine model outperforms the baseline that relies only on the transfer networks (CSN [83]). We quantify the amount of missed hit cold users (and items) that TransNet can reduce by comparing with the pure CF method, showing that TransNet is able to alleviate the cold-start issue.

In real world services, data sources may belong to different providers (e.g. product reviews provided by Amazon while social relations provided by Facebook). The data privacy is a big issue when we combine the multiple data sources. In future work, it is worth developing new learning techniques to learn a combined model while protecting user privacy.

Chapter 5

Deep Feature-based Knowledge Transfer in Heterogeneous User-Interest News Recommendation

We investigate how to solve the cross-corpus news recommendation for unseen users in future inference tasks where those users are not seen during the training. This is a problem where traditional content-based recommendation techniques often fail. Luckily, in real-world recommendation services, some publisher (e.g., Daily news) may have accumulated a large corpus with lots of consumers which can be used for a newly deployed publisher (e.g., Political news). To take advantage of the existing corpus, we propose a transfer learning model (dubbed as TrNews) for news recommendation to transfer the knowledge from a source corpus to a target corpus. To tackle the heterogeneity of different user interests and of different word distributions across corpora, we design a translator-based transfer-learning strategy to learn a representation mapping between source and target corpora. The learned translator can be used to generate representations for unseen users in the future. We show through experiments on real-world datasets that TrNews is better than various baselines in terms of four metrics. We also show that our translator is effective among existing transfer strategies.

5.1 Introduction

News recommendation is key to satisfying users' information need for online services. Some news articles, such as breaking news, are manually selected by publishers and displayed for all users. A huge number of news articles generated everyday make it impossible for editors

and users to read through all of them, raising the issue of information overload. Online news platforms provide a service of personalized news recommendation by learning from the past reading history of users, e.g., Google [18, 70], Yahoo [89, 114], and Bing news [73, 117].

When a new user uses the system (cold-start users) or a new article is just created (cold-start items), there are too few observations for them to train a reliable recommender system. Content-based techniques exploit the content information of news (e.g., words and tags) and hence new articles can be recommended to existing users [98]. Content-based recommendation, however, suffers from the issue of data sparsity since there is no reading history for them to be used to build a profile [96].

Transfer learning is a common technique for alleviating the issues of data sparsity [9, 69, 95]. A user may have access to many websites such as Twitter.com and Youtube.com [52, 108], and consume different categories of products such as movies and books [61]. In this case, transfer learning approaches can recommend articles to a new user in the target domain by exploiting knowledge from the relevant source domains for this new user.

A technical challenge for transfer learning approaches is that user interests are quite different across domains (corpora). For example, users do not use Twitter for the same purpose. A user may follow up on news about “Donald Trump” because she supports republican party (in the political news domain), while she may follow up account *@taylorswift13* (“Taylor Swift”) because she loves music (in the entertainment news domain). Another challenge is that the word distribution and feature space are different across domains. For example, vocabularies are different for describing political news and entertainment news. An illustration is depicted in Figure 5.1. As a result, the user profile computed from her news history is heterogeneous across domains.

Several strategies have been proposed for heterogeneous transfer learning [136]. The transferable contextual bandit (TCB) [69] learns a translation matrix to translate target feature examples to the source feature space. This linear mapping strategy is also used in collaborative cross networks (CoNet) [43] and deep dual transfer cross domain recommendation (DDTCDR) [65]. To capture complex relations between source and target domains, some nonlinear mapping strategy is considered in the embedding and mapping cross-domain recommendation (EMCDR) [77] which learns a supervised regression between source and target factors using a multilayer perceptron (MLP). Since aligned examples between source and target domains are limited, they may face the overfitting issues.

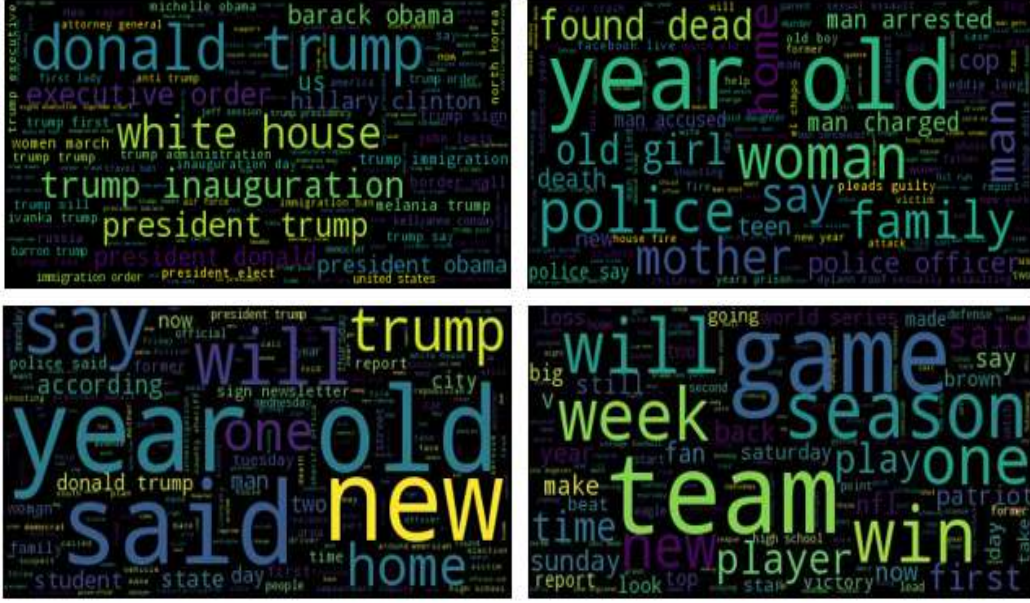


Figure 5.1: Word clouds of two news corpora. Top: Cheetah Mobile data. Bottom: MIND data. Left and right parts represent different domains (categories).

To tackle challenges of heterogeneous user interests and limited aligned data between domains, we propose a novel transfer learning model (TrNews) for cross-corpora news recommendation. TrNews builds a bridge between two base networks (one for each corpus, see Section 5.2) through the proposed translator-based transfer strategy. The translator in TrNews captures the relations between source and target domains by learning a nonlinear mapping between them (Section 5.3.1). The heterogeneity is alleviated by translating user interests across corpora. TrNews uses the translator to transfer knowledge between source and target networks. TrNews alleviates the limited data in a way of alternating training. The learned translator is used to infer the representations of unseen users in the future (Section 5.3.3). By “translating” the source representation of a user to the target domain, TrNews offers an easy solution to create unseen users’ target representations. TrNews outperforms the state-of-the-art recommendation methods on four real-world datasets in terms of four metrics (Section 5.4.2), while having an explanation advantage by allowing the visualization of the importance of each news article in the history to the future news.

5.2 Problem Description

5.2.1 Problem Formulation

Objective Function

For content-based collaborative filtering, firstly there is a binary matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$ to describe user-item interactions where each entry $r_{ui} \in \{0, 1\}$ is 1 if user u has an interaction with item i and 0 otherwise where m and n are the size of users and items respectively. Secondly, in news recommendation, or content-based recommendation, the items have affiliated content information including the news title and news abstract. Denote the content of news item i by $d_i = [w_k]_{k=1}^{n_i}$ where n_i is the length of word sequence in it and w_k is the k -th word in the sequence. Content-based collaborative filtering leverages both interaction and content information leading to the interaction function has the form of:

$$f_{CBF}(u, i, d_i | \Theta). \quad (5.1)$$

A synthetic model of interaction and content estimates the probability of his/her preferences conditioned on this user, this item, and the content text:

$$\hat{r}_{ui} \triangleq p(r_{ui} = 1 | u, i, d_i). \quad (5.2)$$

The likelihood function of the entire matrix $\mathbf{R}_T$ is then defined as:

$$p(\mathbf{R}) = \prod_u \prod_i \prod_{k \in d_i} p(r_{ui} | u, i, d_i). \quad (5.3)$$

In cross-domain news recommendation, we have a target domain (e.g., news domain) user-item interaction matrix $\mathbf{R}_T \in \mathbb{R}^{m \times n_T}$ and a source domain (e.g., app domain) matrix $\mathbf{R}_S \in \mathbb{R}^{m \times n_S}$ where $m = |\mathcal{U}|$ and $n_T = |\mathcal{I}_T|$ ($n_S = |\mathcal{I}_S|$) is the size of users $\mathcal{U}$ and target items $\mathcal{I}_T$ (source items $\mathcal{I}_S$). We use u to index users, i to target items, and j to source items. Content-based Neural CF can be extended to leverage the source domain leading to the interaction function has the form of:

$$\begin{aligned} &\text{learning target function } f_{tgt}(u, i, d_i | \Theta_{tgt}), \\ &\text{with the knowledge from the source } f_{src}(u, j, d_j | \Theta_{src}), \end{aligned} \quad (5.4)$$

where both f_{tgt} and f_{src} are a content-based neural CF model. And the parameters of the target and source models are bridged by the shared users across domains and the shared words across corpora.

Evaluation Method

For the task of item recommendation, each user is only interested in identifying top-K items. Note that, we care about the performance in the target domain. As a result, the users and items during the test are related to the target domain only. The knowledge learned from the source domain during the training phase will be exploited in the test of target domain. The items are ranked by their predicted scores:

$$\hat{r}_{ui} = f(u, i, d_i | \Theta), \quad (5.5)$$

where f is the interaction function and Θ denotes model parameters. The goal is to generate a ranked list of items for each user based on her history records, i.e., top-N recommendations. The evaluation metrics are usually AUC, Hit Ratio, NDCG, and MRR. We hope improve the recommendation performance in the target domain with the help of both the content and source domain information.

5.2.2 A Content-based Neural CF Base Model

There is a base network for each of the two domains. We introduce such a base network with a content-based neural CF model in the following. It is an attentional network which has three modules (ψ, ϕ, f) : the news encoder ψ to learn news representations, the user encoder ϕ to learn user representations, and a neural collaborative filtering module f to learn user preferences from reading behaviors.

News encoder

The news encoder module is to learn news representation from its content. The news encoder takes a news article c 's word sequence $d_c = [w_j]_{j=1}^{n_c}$ (n_c is length of c) as the input, and outputs its representation $\psi(c) \triangleq \psi(d_c) \in \mathbb{R}^D$ where D is the dimensionality. We compute the average of c 's word embeddings by:

$$\psi(d_c) = \frac{1}{|d_c|} \sum_{w \in d_c} e_w, \quad (5.6)$$

where e_w is the embedding of w .

User encoder

The user encoder module is to learn the user representation from their reading history. The user encoder takes a user's reading history $[\psi(d_i^{(u)})]_{i=1}^{n_u}$ (n_u is length of u 's history) as input, and

outputs her representation $\phi(u) \triangleq \phi([\psi(d_i^{(u)})]_{i=1}^{n_u}) \in \mathbb{R}^D$ where D is dimensionality.

In detail, given a pair of user and candidate news (u, c) , we get the user representation $\phi(u|c)$ as the weighted sum of her historical news articles' representations:

$$\phi(u|c) = \sum_{i=1}^{n_{uc}} \alpha_i^{(uc)} \psi(d_i^{(u)}). \quad (5.7)$$

The weights $\alpha_i^{(uc)}$'s are computed via attention units by: $\alpha_i^{(uc)} = a([\psi(d_i^{(u)}), \psi(d_c)])$ where a is the attention function with parameters to be learned. We use an MLP to compute it. For a specific candidate news c , we limit the history news to only those articles that are read before it. For notational simplicity, we do not explicitly specify the candidate news when referring to a user representation, i.e., $\phi(u)$ for short of $\phi(u|c)$.

Content-based Neural CF

The neural collaborative filtering module is to learn preferences from user-news interactions. The module takes concatenated representations of user and news $[\phi(u), \psi(c)]$ as input, and outputs preference score

$$\hat{r}_{uc} = f([\phi(u), \psi(c)]), \quad (5.8)$$

where f is an MLP (multi-layer perceptron).

5.3 Methodology

5.3.1 TrNews

Architecture

The architecture of TrNews is shown in Figure 5.2, which has three parts. There are a source network for the source domain S and a target network for the target domain T , respectively. The source and target networks are both an instantiation of the base network (Section 5.2). The translator enables knowledge transfer between the two networks (Section 5.3.1). We give an overview of TrNews before introducing the base network and the translator.

- **Target network** The information flow goes from the input, i.e., (user u , candidate news c_T) to the output, i.e., the preference score $\hat{r}_{uc_T}$, through the following three steps. First,

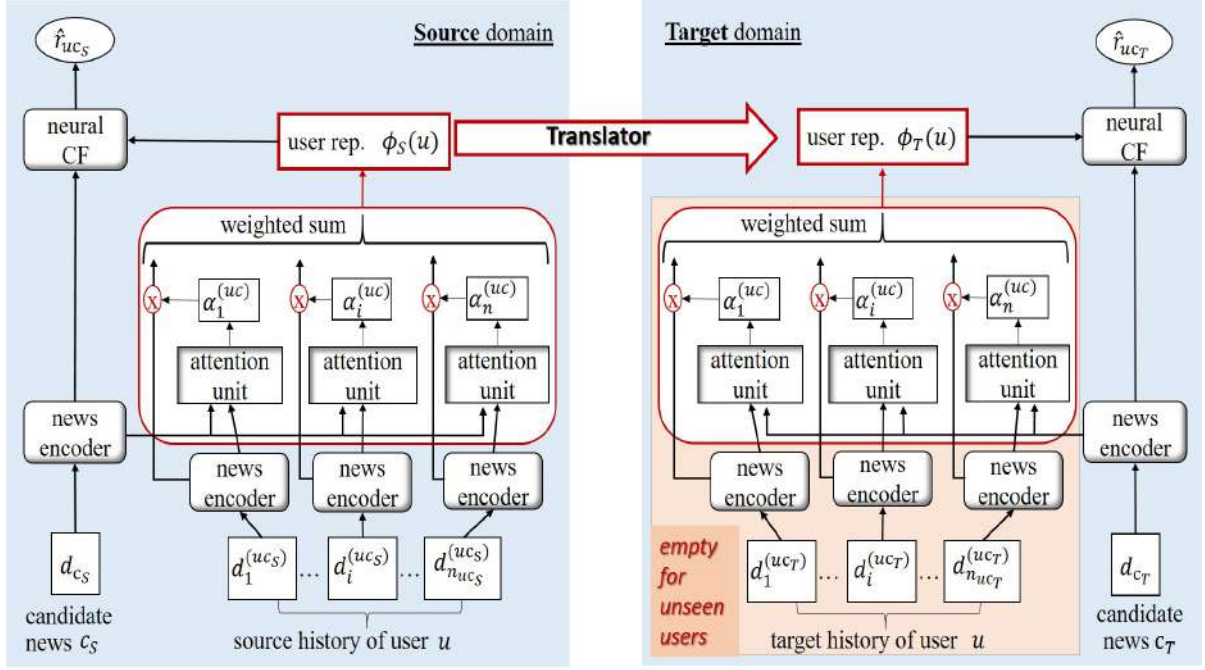


Figure 5.2: Architecture of TrNews. There is a base network for each of the two domains. The shaded gray area in the target network is empty for unseen users. The translator enables knowledge transfer between source and target networks.

the news encoder ψ_T computes the news representation from its content. The candidate news representation is:

$$\psi_T(c_T) = \psi_T(d_{c_T}) \quad (5.9)$$

where d_{c_T} is c_T 's content. The representations of historical news articles $[i]_{i=1}^{n_{uc_T}}$ of the user are $[\psi_T(d_i^{(uc_T)})]_{i=1}^{n_{uc_T}}$ where $d_i^{(uc_T)}$ is i 's content and n_{uc_T} is size of the history. Second, the user encoder ϕ_T computes the user representation from her news history by:

$$\phi_T(u) = \phi_T\left([\psi_T(d_i^{(uc_T)})]_{i=1}^{n_{uc_T}}\right). \quad (5.10)$$

Third, the neural collaborative filtering (CF) module f_T computes the preference score by:

$$\hat{r}_{uc_T} = f_T([\phi_T(u), \psi_T(c_T)]). \quad (5.11)$$

We can denote the target network by a tuple (ψ_T, ϕ_T, f_T) .

- **Source network** Similarly to the three-step computing process in target network, we compute preference score $\hat{r}_{uc_S}$ from input (u, c_S) by:

$$\hat{r}_{uc_S} = f_S([\phi_S(u), \psi_S(c_S)]) \quad (5.12)$$

with tuple (ψ_S, ϕ_S, f_S) .

- **Translator** The translator $\mathcal{F}$ learns a mapping from the user's source representation to her target representation by:

$$\mathcal{F} : \phi_S(u) \rightarrow \phi_T(u). \quad (5.13)$$

Translator

The target network suffers from the data sparsity issue of users who have no reading history. In this section, we propose a transfer learning component (i.e., the translator) to enable knowledge transfer for cross-domain news recommendation. The challenge is that user interests and word distributions are different across domains. For example, we compute the word clouds for two news corpora as shown in Figure 5.1. We can see that their word distributions are quite different and vocabularies are also different. Hence, user representations computed from their news history are heterogeneous across domains.

We build a translator, $\mathcal{F} : \phi_S(u) \rightarrow \phi_T(u)$, to learn a mapping from a user's source representation to her target representation as shown in Figure 5.3. This translator captures the relationship and heterogeneity across domains. The translator learns to approximate the target representation from the source representation.

The translator takes a user's source representation $\phi_S(u)$ as the input, and maps it to a hidden representation z_u via an encoder parameterized by θ , and then gets an approximated representation $\tilde{\phi}_S(u)$ from it via a decoder parameterized by θ' . The parameters $\Theta_{\mathcal{F}} = \{\theta, \theta'\}$ of the translator are optimized to minimize the approximation error:

$$\mathcal{L}_{\mathcal{F}} = \frac{1}{|\mathcal{U}_0|} \sum_{u \in \mathcal{U}_0} \|H\tilde{\phi}_S(u) - \phi_T(u)\|_2^2, \quad (5.14)$$

where $\mathcal{U}_0 = \mathcal{U}_S \cap \mathcal{U}_T$, and $\mathcal{U}_S$ and $\mathcal{U}_T$ are the user sets of source and target domains, respectively. H is to match the dimensions of source and target representations.

Note that, we do not minimize the approximation error between $\phi_S(u)$ and $\tilde{\phi}_S(u)$ as with the standard autoencoder because our goal is to learn a mapping from a user's source representation to her corresponding target representation. After training, the learned mapping function is then used for inferring representations of unseen users in the target domain (the inference process will be described later in Section 5.3.3). It fulfills knowledge transfer from the source to the target domain via a supervised learning process.

Extensions The translator can be generalized to multiple, say k , source domains. We learn k translators using the aligned examples from each of the source domain to the target domain and

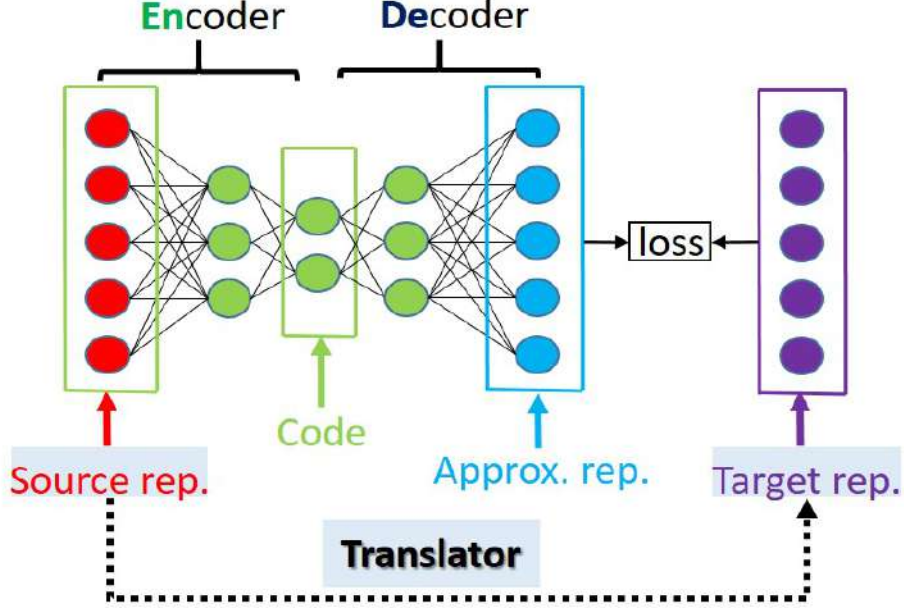


Figure 5.3: Translator. The translator enables knowledge transfer between source and target networks.

then we average (or concatenate) the k mapped representations as the final representation for the user. Another extension is to introduce denoising or stacking techniques into the translator framework, not just the MLP structure in [77].

5.3.2 Objective and Optimization

We learn TrNews in two stages. First, we train the source network using source training examples D_S and train the target network using target training examples D_T , respectively. Second, we train the translator by pairs of user representations computed on-the-fly from source and target networks. We introduce these two stages in detail.

First, TrNews optimizes the parameters associated with target network $\Theta_T = \{\theta_{\phi_T}, \theta_{\psi_T}, \theta_{f_T}\}$ and source network $\Theta_S = \{\theta_{\phi_S}, \theta_{\psi_S}, \theta_{f_S}\}$ by minimizing the joint cross-entropy loss:

$$\begin{aligned} \mathcal{L} = & - \sum_{D_T} (r_{uc_T} \log \hat{r}_{uc_T} + (1 - r_{uc_T}) \log(1 - \hat{r}_{uc_T})) \\ & - \sum_{D_S} (r_{uc_S} \log \hat{r}_{uc_S} + (1 - r_{uc_S}) \log(1 - \hat{r}_{uc_S})), \quad (5.15) \end{aligned}$$

where the two terms on the right-hand side are to optimize losses over user-news examples in the target and source domains, respectively. They are related by the word embedding matrix for the union of words of the two domains. We generate D_T and D_S as follows and take the target domain as an example since the procedure is the same for the source domain. Suppose we have

a whole news reading history for a user u , say $[d_1, d_2, \dots, d_{n_u}]$. Then we generate the positive training examples by sliding over the history sequence:

$$D_T^+ = \{([d_i]_{i=1}^{c-1}, d_c) : c = 2, \dots, n_u\}. \quad (5.16)$$

We adopt the random negative sampling technique [92] to generate the corresponding negative training examples:

$$D_T^- = \{([d_i]_{i=1}^{c-1}, d'_c) : d'_c \notin [d_1, d_2, \dots, d_{n_u}]\}, \quad (5.17)$$

that is, we randomly sample a news article from the corpus as a negative sample which is not in this user's reading history.

Second, TrNews optimizes the parameters associated with the translator $\Theta_{\mathcal{F}} = \{\theta, \theta'\}$ by Eq. (5.14). Since the aligned data is limited, we increase the training pairs by generating them on-the-fly during the training of the two networks, i.e., in an alternating way. The model learning is summarized in Algorithm 2.

Algorithm 1: Training of TrNews.

Input: $D_T, D_S, \mathcal{U}_0$

Output: Source & target networks, translator

for $iter = 1, 2, \dots, 50$ **do**

1. Train target and source networks with mini batch using D_T, D_S respectively;

2. **for** $u \in \mathcal{U}_0$ **do**

(a) Generate source representations $\phi(u_S)$ using source network;

(b) Generate target representations $\phi(u_T)$ using target network;

(c) Train the translator using pairs $(\phi(u_S), \phi(u_T))$ with mini batch;

end

3. Compute metrics on the validation set;

if *No improvement for 10 iters* **then**

Early stopping;

end

end

5.3.3 Inference for Unseen Users

For a new user in the target domain (not seen in the training set $\mathcal{U}_T^{train}$), we do not have any previous history to rely on in learning a user representation for her. That is, the shaded area of

the target network in Figure 5.2 is empty for unseen users.

TrNews estimates a new user u^* 's target representation by mapping from her source representation using the learned translator $\mathcal{F}$ by:

$$\phi_T(u^*) := \mathcal{F}(\phi_S(u^*)), \forall u^* \in \mathcal{U}_S \wedge u^* \notin \mathcal{U}_T^{train}, \quad (5.18)$$

where we compute $\phi_S(u^*)$ using u 's latest reading history in the source domain. Then we can predict the user preference for candidate news c^* by:

$$\hat{r}_{u^*c^*} = f_T([\phi_T(u^*), \psi_T(c^*)]). \quad (5.19)$$

5.4 Experiments

We evaluate the performance of TrNews (Section 5.4.2) and the effectiveness of the translator (Section 5.4.3) in this section.

5.4.1 Data Sets and Evaluation Protocols

Data Sets

We evaluate on two real-world datasets.

- The first *NY, FL, TX, & CA* are four subdatasets extracted from a large dataset provided by an internet company Cheetah Mobile [44, 69]. The information contains news reading logs of users in a large geographical area collected in January of 2017, ranging from New York (NY), Florida (FL), Texas (TX), to California (CA) based on the division of user geolocation. They are treated as four rather than a single because the user set is not overlapped among them. The top two categories (political and daily) of news are used as the cross corpora. The mean length of news articles is around 12 words while the max length is around 50 words. The mean length of user history is around 45 articles while the max length is around 900 articles.
- The second *MIND* is a benchmark dataset released by Microsoft for news recommendation [125]. We use the MIND-small version to investigate the knowledge transfer when news reading examples are not so large and it is publicly available¹. The title and abstract

¹<https://msnews.github.io/>

Data	#user	Target domain			Source domain		
		#news	#reading	#word	#news	#reading	#word
NY	14,419	33,314	158,516	368,000	23,241	139,344	273,894
FL	15,925	33,801	178,307	376,695	25,644	168,081	340,797
TX	20,786	38,395	218,376	421,586	29,797	221,344	343,706
CA	26,981	44,143	281,035	481,959	32,857	258,890	375,612
MIND	25,580	9,372	211,304	461,984	8,577	120,409	346,988

Table 5.1: Statistics of the datasets.

of news are used as the content. The clicked historical news articles are the positive examples for user. The top two categories (news and sports) of news are used as the cross corpora. The word clouds of the two datasets are shown in Figure 5.1 and the statistics are summarized in Table 5.1. The mean length of news articles is around 40 words while the max length is around 123 words. Besides, the mean length of user history is around 13 articles while the max length is around 246 articles.

Evaluation protocol

We randomly split the whole user set into two parts, training and test sets where the ratio is 9:1. Given a user in the test set, for each news in her history, we follow the strategy in [42] to randomly sample a number of negative news, say 99, which are not in her reading history and then evaluate how well the recommender can rank this positive news against these negative ones. For each user in the training set, we reserve her last reading news as the valid set. We follow the typical metrics to evaluate top- K news recommendation [2, 89, 99] which are hit ratio (HR), normalized discounted cumulative gain (NDCG), mean reciprocal rank (MRR), and the area under the ROC curve (AUC). We report the results at cut-off $K \in \{5, 10\}$.

Implementation

We use TensorFlow. The optimizer is Adam [56] with learning rate 0.001. The size of mini batch is 256. The neural CF module has two hidden layers with size 80 and 40 respectively. The size of word embedding is 128. The translator has one hidden layer on the smaller datasets and two on the larger ones. The history is the latest 10 news articles.

5.4.2 Results: Comparing Different Recommenders

In this section, we show the recommendation results by comparing TrNews with different state-of-the-art methods.

Baselines

We compare with following recommendation methods which are trained on the merged source and target datasets by aligning with shared users.

- **POP** [96] recommends the most popular news.
- **LR** [80] is widely used in ads and recommendation. The input is the concatenation of candidate news and user’s representations.
- **DeepFM** [37] is a deep neural network for ads and recommendation based on the wide & deep structure. We use second-order feature interactions of reading history and candidate news, and the input of deep component is the same as LR.
- **DIN** [148] is a deep interest network for ads and recommendation based on the attention mechanism. We use the news content for news representations.
- **TANR** [123] is a state-of-the-art deep news recommendation model using an attention network to learn the user representation. We adopt the news encoder and negative sampling the same with TrNews.

Results

We have observations from results of different recommendation methods as shown in Table 5.7.

Firstly, considering that breaking and headline news articles are usually read by every user, the POP method gets competitive performance in terms of NDCG and MRR since it ranks the popular news higher than the other news.

Secondly, the neural methods are generally better than the traditional, shallow LR method in terms of NDCG, MRR, and AUC on the four subdatasets. It may be that neural networks can learn nonlinear, complex relations between the user and the candidate news to capture user interests and news semantics. Considering that the neural representations of user and candidate news are fed as the input of LR, it gets competitive performance on MIND data.

Approach	Transfer strategy	Formulation
CST [95]	Identity mapping	$\phi_T(u) = \phi_S(u)$
TCB [69]	Linear mapping	$\phi_T(u) = H\phi_S(u)$
DDTCDR [65]		H is orthogonal
EMCDR [77]	Nonlinear mapping	$\phi_T(u) = \text{MLP}(\phi_S(u))$

Table 5.2: Different transfer learning strategies.

Finally, the proposed TrNews model achieves the best performance with a large margin improvement over all other baselines in terms of HR, NDCG, and MRR and also with an improvement in terms of AUC. It validates the necessity of accounting for the heterogeneity of user interests and word distributions across domains. This also shows that the base network is an effective architecture for news recommendation and the translator is effective to enable the knowledge transfer from the source domain to the target domain. In more detail, it is inferior by training a global model from the mixed source and target examples and then using this global model to predict user preferences on the target domain, as baselines do. Instead, it is good by training source and target networks on the source and target domains, respectively, and then learning a mapping between them, as TrNews does.

5.4.3 Results: Comparing Different Transfer Strategies

In this section, we demonstrate the effectiveness of the translator-based transfer-learning strategy.

Baselines

We replace the translator of TrNews with the transfer-learning strategies of baseline methods as summarized in Table 5.2. All baselines are state-of-the-art recommenders and capable of recommending news to cold-start users. Note that, the compared transfer-learning methods are upgraded from their original versions. We strengthen them by using the neural attention architecture as the base component. In their original versions, CST and TCB use matrix factorization (MF) while DDTCDR and EMCDR use multilayer perceptron. The neural attention architecture has shown superior performance over MF and MLP in the literature [123, 148]. As a result, we believe that the improvement will be larger if we compare with their original versions but this is obviously unfair.

Results

We have observations from results of different transfer learning strategies as shown in Table 5.8.

Firstly, the transfer strategy of identity mapping (CST) is generally inferior to the linear (TCB and DDTCDR) and nonlinear (EMCDR and TrNews) strategies. CST directly transfers the source knowledge to the target domain without adaptation and hence suffers from the heterogeneity of user interests and word distributions across domains.

Secondly, the nonlinear transfer strategy of EMCDR is inferior to the linear strategy of TCB in terms of MRR and AUC on the two smaller NY and FL datasets. This is probably because EMCDR increases the model complexity by introducing two large fully-connected layers in its MLP component. In contrast, our translator is based on the small-waist autoencoder-like architecture and hence can resist overfitting to some extent.

Finally, our translator achieves the best performance in terms of NDCG, MRR and AUC on the two smaller NY and FL datasets, and achieves competitive performance on the two larger TX and CA datasets, and achieves the best performance in terms of HR and AUC on the MIND dataset, comparing with other four transfer methods. These results validate that our translator is a general and effective transfer-learning strategy to capture the diverse user interests accurately during the knowledge transfer for the unseen users in cross-domain news recommendation.

5.4.4 Analyses

Benefit of knowledge transfer

We vary the percentage of shared users used to train the translator (see Eq. (5.14)) with:

$$\{90\%, 70\%, 50\%, 30\%\}.$$

We compare with a naive transfer strategy of CST, i.e., the way of direct transfer without adaptation. The results are shown in Figure 5.4 on the New York dataset. We can see that it is beneficial to learn an adaptive mapping during the knowledge transfer even when limited aligned examples are available to train the translator. TrNews improves relative 0.82%, 0.77%, 0.67%, 0.64% in terms of HR@10 performance over CST by varying among $\{90\%, 70\%, 50\%, 30\%\}$ respectively. So we think that the more aligned examples the translator has, the more benefits it achieves.

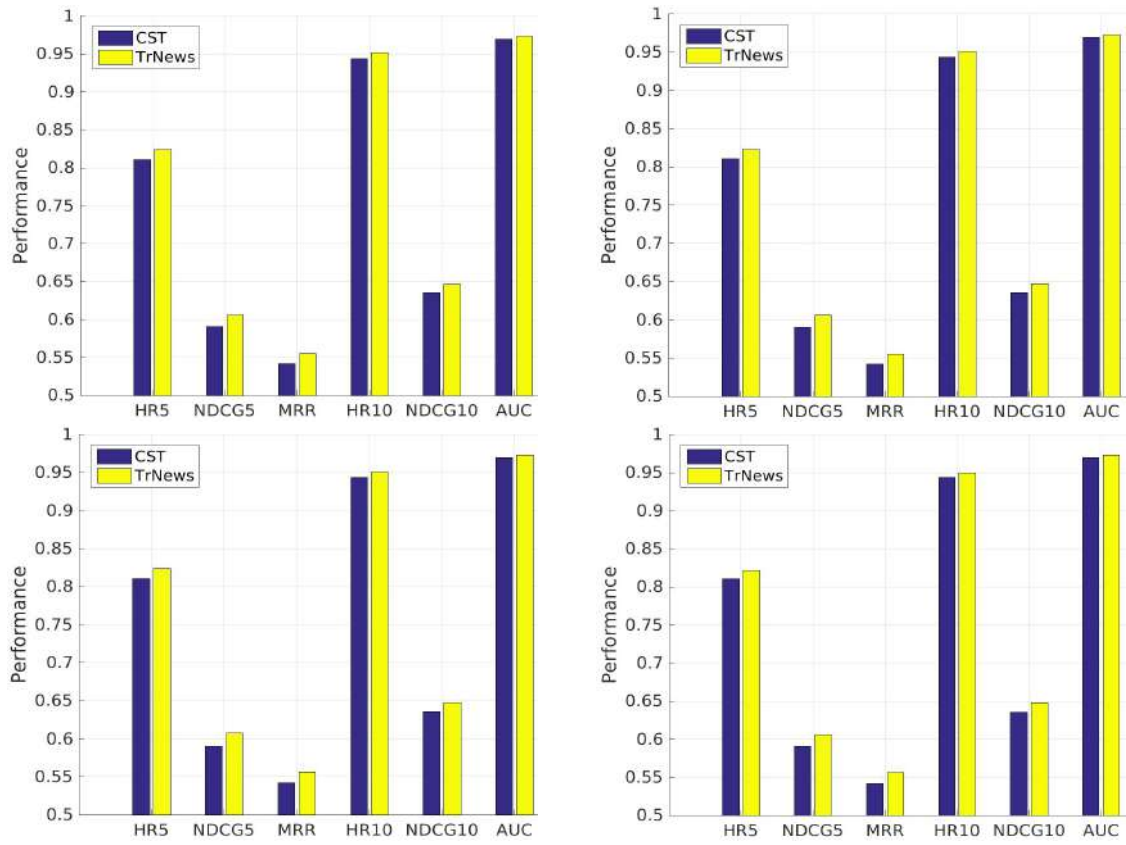


Figure 5.4: Impact of percentage (90%, 70%, 50%, 30%.) of shared users used to train the translator.

Sharing?	HR@5	HR@10	NDCG@5	NDCG@10	MRR	AUC
No	81.31	94.69	59.43	63.72	54.37	97.16
Yes	82.60	95.15	60.78	64.83	55.70	97.28

Table 5.3: Impact of sharing word embeddings between source and target domains.

Strategy	HR@5	HR@10	NDCG@5	NDCG@10	MRR	AUC
Sepa.	82.36	94.86	60.62	64.65	55.51	97.28
Alter.	82.60	95.15	60.78	64.83	55.70	97.28

Table 5.4: Training TrNews with alternating (Alter.) vs separating (Sepa.) strategies.

Impact of sharing word embeddings

We investigate the benefits of sharing word embeddings between source and target domains. There is a word embedding matrix for each of the domains and we share the columns if the corresponding words occur in both domains. Take the New York dataset as an example, the size of the intersection of their word vocabularies is 11,291 while the union is 50,263. From the results in Table 5.3 we can see that it is beneficial to share the word embeddings even when only 22.5% words are intersected between them.

Impact of alternating training

We adopt an alternating training strategy between training the two (source & target) networks and training the translator in our experiments. In this section, we compare this alternating strategy with the separating strategy which firstly trains the two networks and then trains the translator after completing the training of the two networks. That is, the training pairs of user representations for the translator are not generated on-the-fly during the training of source and target networks but generated only once after finishing their training. From the results in Table 5.4, we see that the alternating strategy works slightly better. This is probably because the aligned data between domains is limited and the alternating strategy increases the size of training pairs.

Impact of two-stage learning

We adopt a two-stage model learning between training the two (source & target) networks and training the translator in our experiments. In this section, we compare this two-stage learning

Training	HR@5	HR@10	NDCG@5	NDCG@10	MRR	AUC
end-to-end	81.85	94.63	60.58	64.74	55.68	97.03
two-stage	82.60	95.15	60.78	64.83	55.70	97.28

Table 5.5: Training TrNews in two-stage vs end-to-end.

with an end-to-end learning which jointly trains the two networks and the translator. That is, the parameters of the translator depend on the word embedding matrix and on parameters of the user encoder. From the results in Table 5.5, we see that the two-stage learning works slightly better. This is probably because the aligned data between domains is too limited to reliably update the parameters which do not belong to the parameters of the translator.

Impact of the length of the history

Since we generate the training examples by sliding over the whole reading history for each user, the length of reading history is a key parameter to influence the performance of TrNews. We investigate how the length of the history affects the performance by varying it with $\{3, 5, 10, 15, 20\}$. The results on the New York dataset are shown in Figure 5.5a. We can observe that increasing the size of the sliding window is sometimes harmful to the performance, and TrNews achieves good results for length 10. This is probably because of the characteristics of news freshness and of the dynamics of user interests. That is, the latest history matters more in general. Also, increasing the length of the input makes the training time increase rapidly, which are 58, 83, 143, 174, and 215 seconds when varying with $\{3, 5, 10, 15, 20\}$ respectively.

Impact of the embedding size

In this section, we evaluate how different choices of some key hyperparameter affect the performance of TrNews. Except for the parameter being analyzed, all other parameters remain the same. Since we compute the news and user representations using the content of words, the size of word embedding is a key parameter to influence representations of words, uses, and news articles, and hence the performance of TrNews. We investigate how embedding size affects the performance by varying it with $\{32, 64, 100, 128, 200\}$. The results on the New York dataset are shown in Figure 5.5b. We can observe that increasing the embedding size is generally not harmful to the performance until 200, and TrNews achieves good results for embedding size 128. Changing it to 200 harms the performance a little bit since the model complexity also increases.

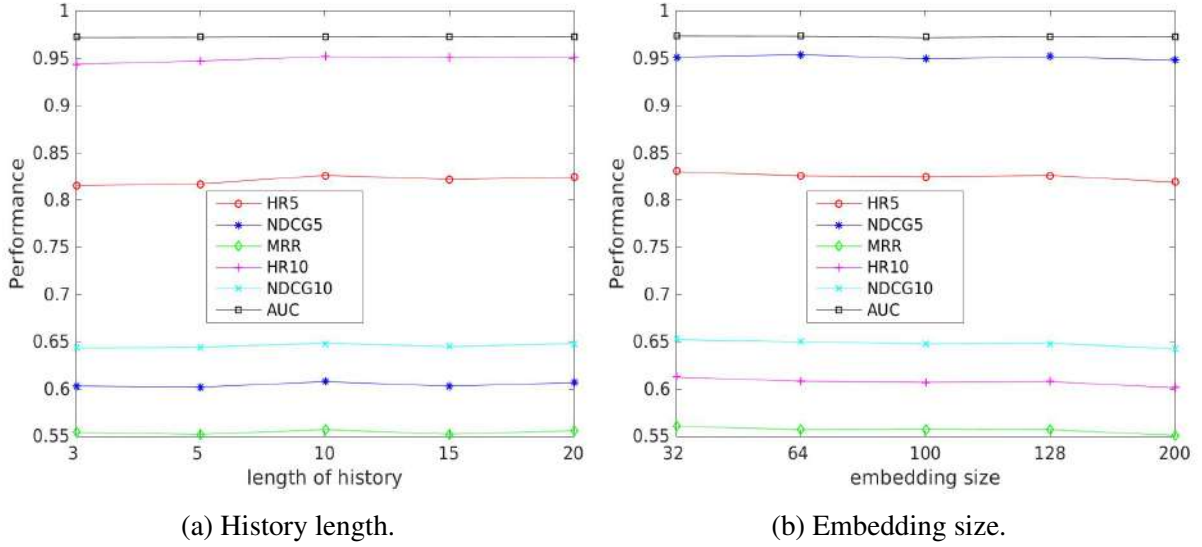


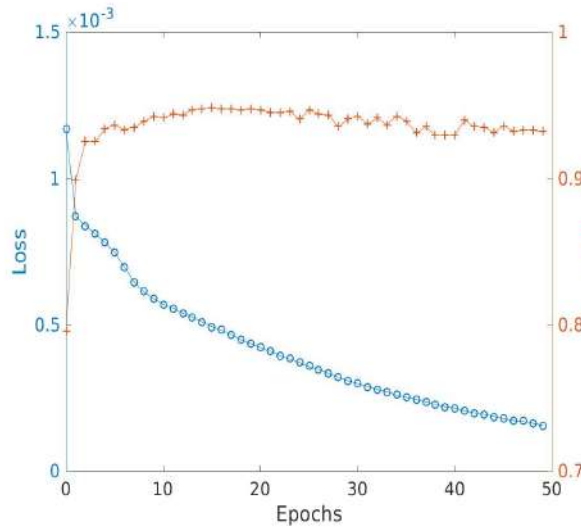
Figure 5.5: Impact of the history length (left) and embedding size (right).

Optimization performance and loss

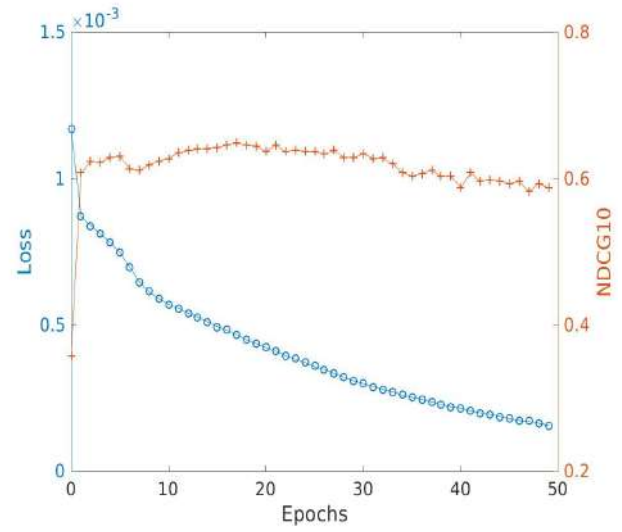
We show the optimization performance and loss over iterations on the New York dataset in Figure 5.6. We can see that with more iterations, the training losses gradually decrease and the recommendation performance is improved accordingly. The most effective updates are occurred in the first 15 iterations, and performance gradually improves until 30 iterations. With more iterations, TrNews is relatively stable. For the training time, TrNews spends 143 seconds per iteration. As a reference, it is 134s for DIN and 139s for TCB, which indicates that the training cost of TrNews is efficient by comparing with baselines. Furthermore, the test time is 150s. The experimental environment is Tensorflow 1.5.0 with Python 3.6 conducted on Linux CentOS 7 where The GPU is Nvidia TITAN Xp based on CUDA V7.0.27.

Examining user profiles

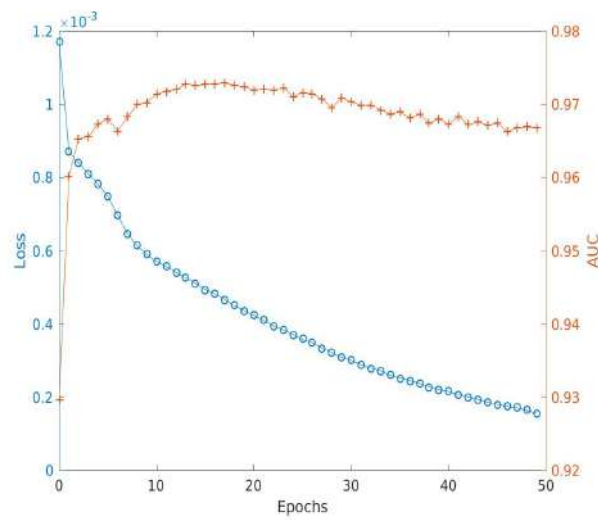
One advantage of TrNews is that it can explain which article in a user’s history matters the most for a candidate article by using attention weights in the user encoder module. Table 6.7 shows an example of interactions between some user’s history articles No. 0-9 and a candidate article No. 10, i.e., the user reads the candidate article after read these ten historical articles. We can see that the latest three articles matter the most since the user interests may remain the same during a short period. The oldest two articles, however, also have some impact on the candidate article, reflecting that the user interests may mix with a long-term characteristic. TrNews can capture these subtle short- and long-term user interests.



(a) HR@10.



(b) NDCG@10.



(c) AUC.

Figure 5.6: Performance (right Y-axis in red cross) and loss (left Y-axis in blue circle) varying with training iterations.

No.	News title	Attn. weight
0	hillary clinton makes a low-key return to washington	0.04
1	the hidden message in obama’s ‘farewell’ speech	0.12*
2	here’s why sasha obama skipped the farewell address	0.00
3	donald trump’s ‘prostitute scandal’ was filmed by cameras and recorded with microphones hidden behind the walls	0.00
4	white house official explains sasha obama’s absence at father’s farewell speech	0.00
5	irish bookie puts odds on trump’s administration, inauguration and impeachment	0.00
6	heads are finally beginning to roll at the clinton foundation	0.00
7	donald trump’s incoming administration considering white house without press corps	0.76
8	donald trump says merkel made ‘big mistake’ on migrants	0.05
9	controversial clinton global initiative closing its doors for good	0.00
10	army chief gen. bipin rawat talks about equal responsibility for women in the frontlines. we couldn’t agree more	N/A

Table 5.6: Example I: Some articles matter more while some are negligible. (No. 10 is the candidate news)

5.5 Related Work

Content recommendation Content-based recommendation exploits the content information about items (e.g., news title and article body [49, 76, 125, 129, 131], tag, vlog [32]), builds a profile for each user, and then matches users to items [72, 124, 141]. It is effective for items with content or auxiliary information but suffers from the issues of data sparsity for users. DCT [3] constructs a user-user similarity matrix from user demographic features including gender, age, occupation, and location [96]. NT-MF [52] constructs a user-user similarity matrix from Twitter texts. BrowseGraph [114] addresses the freshly news recommendation by constructing a graph using URL links between web pages. NAC [101] transfers from multiple source domains through the attention mechanism. PdMS [27] assumes that there are many recommender models available to select items for a user, and introduces a multi-armed bandit for model selection. LLAE [64] needs a social network as side information for cold-start users. Different from the aforementioned works, we aim to recommending news to unseen users by transferring knowledge from a source domain to a target domain.

Transfer learning Transfer learning aims at improving the performance of a target domain by exploiting knowledge from source domains [93]. A special setting is domain adaptation where a source domain provides labeled training examples while the target domain provides instances on which the model is meant to be deployed [34, 66]. The coordinate system transfer (CST) [95] firstly learns the principle coordinate of users in the source domain, and then transfers it to the target domain in the way of warm-start initialization. This is equivalent to an identity mapping from users' source representations to their corresponding target representations. TCB [69] learns a linear mapping to translate target feature examples to the source feature space because there are many labelled data in the source domain. This linear strategy is also used in CoNet [43] and DDTCDR [65] which transforms the source representations to the target domain by a translation matrix. Nonlinear mapping strategy [28, 77, 149] is to learn a supervised mapping function between source and target latent factors by using neural networks. SSCDR [54] extends them to the semi-supervised mapping setting. Our translator is general to accommodate these identity, linear, and nonlinear transfer-learning strategies.

5.6 Conclusions

We investigate the cross-domain news recommendation via transfer learning. The experiments on real-word datasets demonstrate the necessity of tackling heterogeneity of user interests and word distributions across domains. Our TrNews model and its translator component are effective to transfer knowledge from the source network to the target network. We also shows that it is beneficial to learn a mapping from the source domain to the target domain even when only a small amount of aligned examples are available. In future works, we will focus on preserving the privacy of the source domain when we transfer its knowledge.

NY	HR@5	HR@10	NDCG@5	NDCG@10	MRR	AUC
POP	52.96	67.66	40.34*	45.10	39.89*	77.92
LR	53.24	74.00	36.15	42.86	34.95	91.64
TANR	52.53	71.63	37.24	43.37	36.50	91.35
DeepFM	52.02	73.71	39.17	45.38	39.56	91.79
DIN	57.10*	75.66*	40.23	46.13*	38.65	92.29*
TrNews	82.60	95.15	60.78	64.83	55.70	97.28
FL	HR@5	HR@10	NDCG@5	NDCG@10	MRR	AUC
POP	52.45	66.14	39.72*	44.15	39.15*	79.33
LR	54.26*	73.90*	37.15	43.56	35.89	91.79
TANR	49.98	69.46	36.08	42.37	35.95	90.88
DeepFM	52.36	73.02	36.05	42.74	36.29	91.64
DIN	53.98	73.33	37.96	44.18*	36.96	91.86*
TrNews	81.83	94.45	62.53	66.63	58.39	97.41
TX	HR@5	HR@10	NDCG@5	NDCG@10	MRR	AUC
POP	54.21	67.87	40.62*	45.03*	39.64*	81.31
LR	55.72*	73.80*	39.24	44.97	37.78	91.74*
TANR	49.87	68.75	35.82	41.89	35.59	90.56
DeepFM	52.19	71.95	35.40	41.92	35.65	91.17
DIN	53.72	72.70	38.47	44.59	37.62	91.53
TrNews	81.50	94.67	61.76	66.11	57.49	97.21
CA	HR@5	HR@10	NDCG@5	NDCG@10	MRR	AUC
POP	58.32*	71.19	44.71*	48.86*	43.44*	83.38
LR	58.82	75.67*	42.16	47.65	40.44	92.37*
TANR	49.87	68.75	35.81	41.88	35.58	90.56
DeepFM	55.58	74.73	38.82	45.16	38.21	92.25
DIN	55.31	73.70	40.14	46.09	39.20	92.03
TrNews	81.54	94.72	61.99	66.25	57.70	97.22
MIND	HR@5	HR@10	NDCG@5	NDCG@10	MRR	AUC
POP	84.18	92.80	69.61	72.43	66.33	95.13
LR	92.69*	96.66*	85.81	87.11*	84.33*	97.92*
TANR	89.94	95.34	89.94*	83.38	79.84	97.86
DeepFM	89.16	94.78	79.36	81.19	77.12	97.63
DIN	89.28	94.88	80.16	82.03	78.22	97.63
TrNews	97.36	99.02	94.16	94.74	93.45	99.47

Table 5.7: Comparison of different recommenders.

NY	HR@5	HR@10	NDCG@5	NDCG@10	MRR	AUC
CST	81.04	94.37	59.04	63.56	54.19	96.94
TCB	82.18	94.92*	60.36*	64.46*	55.23*	97.28*
DDTCDR	82.27	94.90	59.82	63.90	54.51	97.25
EMCDR	82.44*	94.87	60.35	64.33	55.06	97.24
TrNews	82.60	95.15	60.78	64.83	55.70	97.28
FL	HR@5	HR@10	NDCG@5	NDCG@10	MRR	AUC
CST	79.29	93.91	59.03	63.60	54.74	97.07
TCB	81.51	94.83	62.06	66.33*	57.90*	97.40*
DDTCDR	81.39	94.63*	61.76	66.12	57.68	97.37
EMCDR	81.52*	94.47	62.14*	66.23	57.87	97.37
TrNews	81.83	94.45	62.53	66.63	58.39	97.41
TX	HR@5	HR@10	NDCG@5	NDCG@10	MRR	AUC
CST	78.74	94.20	58.53	63.48	54.56	96.92
TCB	80.68	94.12	61.06	65.38	56.97	97.10
DDTCDR	81.08	94.57	61.02	65.50	56.87	97.10
EMCDR	81.34*	94.72	61.78	66.11*	57.59	97.16*
TrNews	81.50	94.67*	61.76*	66.11	57.49*	97.21
CA	HR@5	HR@10	NDCG@5	NDCG@10	MRR	AUC
CST	79.92	93.71*	60.19	64.63	55.97	97.12
TCB	80.90*	93.71*	62.32	66.45	58.35	97.36
DDTCDR	80.22	93.47	61.42	65.72	57.44	97.25
EMCDR	80.53	93.33	62.04*	66.18	58.11*	97.30*
TrNews	81.54	94.72	61.99	66.25*	57.70	97.22
MIND	HR@5	HR@10	NDCG@5	NDCG@10	MRR	AUC
CST	96.93	98.49	93.83	94.34	93.09	99.32
TCB	97.41*	98.94	94.21*	94.72	93.43	99.41
DDTCDR	97.38	98.99	94.25	94.78	93.49	99.46*
EMCDR	97.42	99.01*	94.16	94.68	93.35	99.45
TrNews	97.36	99.02	94.16	94.74*	93.45*	99.47

Table 5.8: Comparison of different transfer strategies.

Chapter 6

Adversarial Knowledge Transfer in Recommendation

Transfer learning is an effective technique to improve a target recommender system with the knowledge from a source domain. Existing research focuses on the recommendation performance of the target domain while ignores the privacy leakage of the source domain. The transferred knowledge, however, may unintendedly leak private information of the source domain. For example, an attacker can accurately infer user demographics from their historical purchase provided by a source domain data owner. In this chapter, we address the above privacy-preserving issue by learning a privacy-aware neural representation by improving target performance while protecting source privacy. The key idea is to simulate the attacks during the training for protecting unseen users' privacy in the future, modeled by an adversarial game, so that the transfer learning model becomes robust to attacks. Experiments show that the proposed PrivNet model can successfully disentangle the knowledge benefitting the transfer from leaking the privacy.

6.1 Introduction

Recommender systems (RSs) are widely used in everyday life ranging from Amazon products [116, 148] and YouTube videos [15, 32] to Twitter microblogs [51] and news feeds [76, 117]. RSs estimate user preferences on items from their historical interactions. RSs, however, cannot learn a reliable preference model if there are too few interactions in the case of new users and items, i.e., suffering from the data sparsity issues.

Transfer learning is an effective technique for alleviating the issues of data sparsity by exploiting the knowledge from related domains [69, 95]. We may infer user preferences on

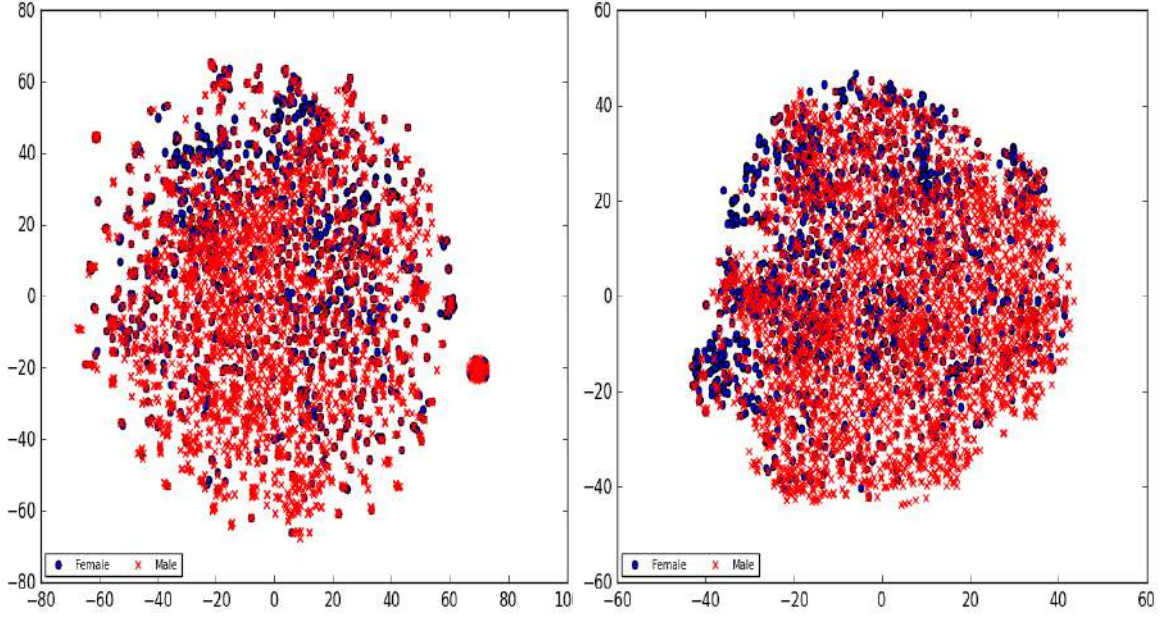


Figure 6.1: t-SNE projection of transferred representations of users with (left) and without (right) training of PrivNet on the MovieLens-Gender dataset. (see Section 6.4.5 for details)

videos from their Tweet texts [52], from movies to books [61], and from news to apps [43, 44]. These behaviors across domains are different views of the same user and may be driven by some inherent user interests [25].

There is a privacy concern when the source domain shares their data to the target domain due to the ever-increasing user data abuse and privacy regulations [102, 137]. Private information contains those attributes that users do not want to disclose, such as gender and age [53]. They can be used to train better recommendation models by alleviating the data sparsity issues to build better user profiles [15, 146]. Previous work [5, 122] shows that an attacker can accurately infer a user’s gender, age, and occupation from their ratings, recommendation results, and a small amount of users who reveal their demographics.

Privacy-leakage Example On a typically widely used movie recommendation benchmark, MovieLens¹, there are 50 movies rated by Female only (e.g., Country Life (1994)), while 350 movies rated by Male only (e.g., Time Masters (1982)). It implies that the occurrence of a rating, regardless of its numeric value (true or noisy), leaks the user privacy on revealing their genders. When these source rating examples are transferred to a target domain, the third party (the target domain or the malicious attacker) can infer the user privacy of the source domain from the rating instances, even though the source domain didn’t explicitly provide any user private attributes to the target domain.

¹<https://grouplens.org/datasets/movielens/>

A technical challenge for protecting user privacy in transfer learning is that the transferred knowledge has dual roles: usefulness to improve target recommendation and uselessness to infer source user privacy.

Another challenge is that the recommender in the target domain does not know the attackers and has no control over it during the test. The goal of the recommender model is to recommend ranked items to users such that any potential adversary cannot infer users' private attributes (e.g., age, gender and occupation). However, a challenge is that the recommendation system does not know the malicious attacker's model. The attacker can iteratively adapt its model regarding to existing recommender since the attacker can get the recommended results from the recommender (the recommended results are publicly visible to users) but not vice versa. In other words, the attacker is in the dark place which is not visible to the target recommender while the target recommender is in the light place which is visible to the attacker.

In this chapter, we propose a novel model (PrivNet) to achieve the two goals by learning privacy-aware transferable knowledge such that it is useful for improving recommendation performance in the target domain while it is useless to infer private information of the source domain. The key idea is to simulate the attack during the training for protecting unseen users' privacy in the future. The privacy attacker and the recommender are naturally modeled by an adversarial learning game. The main contributions are two-fold:

- PrivNet is the first to address the privacy protection issues, i.e., protecting source user private attributes while improving the target performance, during the knowledge transfer in neural recommendation.
- PrivNet achieves a good tradeoff between the utility and privacy of the source information through evaluation on real-world datasets by comparing with strategies of adding noise (i.e., differential privacy) and perturbing ratings.

6.2 Problem Description

6.2.1 Problem Formulation

We have two domains, a source domain S and a target domain T . User sets in two domains are shared, denoted by $\mathcal{U}$ (of size $m = |\mathcal{U}|$). Denote item sets in two domains by $\mathcal{I}_S$ and $\mathcal{I}_T$ (of size $n_S = |\mathcal{I}_S|$ and $n_T = |\mathcal{I}_T|$), respectively. For the target domain, a binary matrix $\mathbf{R}_T \in \mathbb{R}^{m \times n_T}$

describes the user-item interactions, where the entry $r_{ui} \in \{0, 1\}$ equals 1 if user u has an interaction with item i and 0 otherwise. Similarly, for the source domain, we have $\mathbf{R}_S \in \mathbb{R}^{m \times n_S}$ and the entry $r_{uj} \in \{0, 1\}$. We reserve i and j to index the target and source items, respectively. Let $\mathbf{Y}^p \in \mathbb{R}^{m \times c_p}$ denote the p -th user private attribute (e.g., p ='Gender') matrix where each entry $y_{u,p}$ is the value of the p -th private information for user u (e.g., $y_{u,p}$ ='Male') and there are c_p choices. Denote all n private attributes data by $\mathbf{Y} = \{\mathbf{Y}^p\}_{p=1}^n$ (e.g., Gender, Age). We can define the problem as follows:

PROBLEM: Privacy-aware transfer learning in recommendation.

INPUT: $\mathbf{R}_T, \mathbf{R}_S, \mathbf{Y}$.

OUTPUT: Generate a ranked list of items for users in the target domain.

REQUIRE: An attacker is difficult to infer the source user private attributes from the knowledge transferred to the target domain.

ASSUMPTION: Some users $\mathcal{U}^{pub} \subset \mathcal{U}$ share their private information with the public profile.

6.2.2 Recommender

In this section, we introduce a novel transfer-learning recommender which has three parts, a source network for the source domain, a target network for the target domain, and a knowledge transfer unit between the two domains.

Target network

The input is a pair of (user, item) and the output is their matching degree. The user is represented by their w -sized historical items $[i_1, \dots, i_w]$. First, an item embedding matrix $\mathbf{A}_T$ projects the discrete item indices to the d -dimensional continuous representations: $\mathbf{x}_i$ and $\mathbf{x}_{i_*}$ where $*$ $\in [1, 2, \dots, w]$. Second, the user representation $\mathbf{x}_u$ is computed by the user encoder module based on an attention mechanism by querying their historical items with the predicted item:

$$\mathbf{x}_u = \sum_{i_*} \alpha_{i_*} \mathbf{x}_{i_*}, \quad (6.1)$$

where $\alpha_{i_*} = \mathbf{x}_i^T \mathbf{x}_{i_*}$ (normalized: $\sum \alpha_{i_*} = 1$). Third, a multilayer perceptron (MLP) f_T parameterized by ϕ_T is used to compute target preference score (the notation $[\cdot, \cdot]$ denotes concatenation):

$$\hat{r}_{ui} = P(r_{ui}|u, i; \theta_T) = f_T([\mathbf{x}_u, \mathbf{x}_i]), \quad (6.2)$$

where $\theta_T = \{\mathbf{A}_T, \phi_T\}$ is the model parameter.

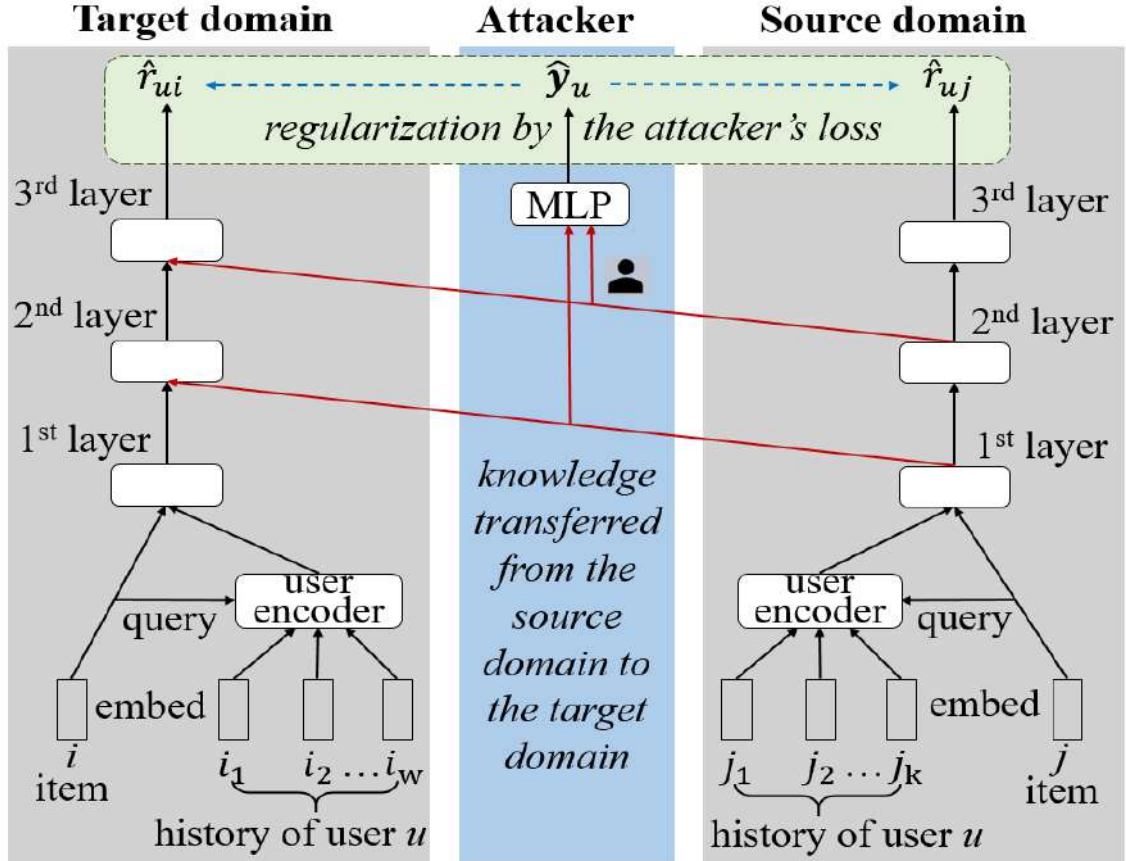


Figure 6.2: Architecture of PrivNet (a version of three layers). It has two components: the recommender (the left & right parts, see Section 6.2.2) is a representation-based transfer learning model where the red arrows indicate the representations transferred from the source domain to the target domain in a multilayer way. The privacy attacker (the middle part, see Section 6.3.1) marked by an avatar infers user privacy from the transferred representations. PrivNet (see Section 6.3.1) exploits the knowledge from the source domain with regularization from the adversary loss of the attacker indicated by the dotted box.

Source network

Similar to the three-step computing process in the target network, we compute the source preference score by:

$$\hat{r}_{uj} = P(r_{uj}|u, j; \theta_S) = f_S([\mathbf{x}_u, \mathbf{x}_j]), \quad (6.3)$$

where $\theta_S = \{\mathbf{A}_S, \phi_S\}$ is the model parameter with item embedding matrix $\mathbf{A}_S$ and multilayer perceptron ϕ_S .

Transfer unit

The transfer unit implements the knowledge transfer from the source to the target domain. Since typical neural networks have more than one layer, say L , the representations are transferred in a multilayer way. Let $\mathbf{x}_{u|\#}^\ell$ where $\# \in \{S, T\}$ be user u 's source/target representation in the ℓ -th layer ($\ell = 1, 2, \dots, L-1$) where $\mathbf{x}_{u|S}^1 = [\mathbf{x}_u, \mathbf{x}_j]$ and $\mathbf{x}_{u|T}^1 = [\mathbf{x}_u, \mathbf{x}_i]$. The transferred representation is computed by projecting the source representation to the space of target representations with a translation matrix $\mathbf{H}^\ell$:

$$\mathbf{x}_{u|trans}^\ell = \mathbf{H}^\ell \mathbf{x}_{u|S}^\ell, \quad (6.4)$$

With the knowledge from the source domain, the target network learns a linear combination of the two input activations from both networks and then feeds these combinations as input to the successive layer's filter. In detail, the $(\ell + 1)$ -th layer's input of the target network is computed by:

$$\mathbf{W}_T^\ell \mathbf{x}_{u|T}^\ell + \mathbf{x}_{u|trans}^\ell, \quad (6.5)$$

where $\mathbf{W}_T^\ell$ is the connection weight matrix in the ℓ -th layer of the target network. The total transferred knowledge is concatenated by all layers's representations:

$$\mathbf{x}_{u|trans} = [\mathbf{x}_{u|trans}^\ell]_{\ell=1}^{L-1}. \quad (6.6)$$

Objective

The recommender minimizes the negative logarithm likelihood:

$$\mathcal{L}(\theta) = - \sum_{D_T} \log P(r_{ui}|u, i; \theta_T) - \sum_{D_S} \log P(r_{uj}|u, j; \theta_S), \quad (6.7)$$

where $\theta = \{\theta_T, \theta_S, \{\mathbf{H}^\ell\}_{\ell=1}^{L-1}\}$, D_T and D_S are target and source training examples, respectively.

We introduce how to generate them in Section 6.3.2.

6.3 Methodology

The architecture of PrivNet is shown in Figure 6.2. It has two components, a recommender and an attacker. We present an attack against the recommender (Section 6.3.1). We propose PrivNet to protect source user privacy during the knowledge transfer (Section 6.3.1).

6.3.1 PrivNet

Attacker

The recommender can fulfil the Problem 1 (see Section 6.2) if there is no attacker existing. A challenge for the recommender is that it does not know the attacker models in advance. To address this challenge, we add an attacker component during the training to simulate the attacks for the test. By integrating a simulated attacker into the recommender, it can deal with the unseen attacks in the future. In this section, we introduce an attacker to infer the user private information from the transferred knowledge. In the next Section 6.3.1, we will introduce an adversarial recommender by exploiting the simulated attacker to regularize the recommendation process in order to fool the adversary so that it can protect the privacy of unseen users in the future.

The attacker model predicts the private user attribute from their source representation sent to the target domain:

$$\begin{aligned}\hat{y}_{u,p} &= P(y_{u,p} | \mathbf{x}_{u|trans}; \theta_p) \\ &= f_p(\mathbf{x}_{u|trans}; \theta_p),\end{aligned}\tag{6.8}$$

where $\hat{y}_{u,p}$ is the predicted value of user u 's p -th private attribute and $p = 1, \dots, n$. The f_p is the prediction model parameterized by θ_p . Note that, an attacker can use any prediction model and here we use an MLP due to its nonlinearity and generality.

For all n private user attributes, the attacker model minimizes the multitask loss:

$$\mathcal{L}(\Theta) = -\frac{1}{n} \sum_p \sum_{D_p} \log P(y_{u,p} | \mathbf{x}_{u|trans}; \theta_p),\tag{6.9}$$

where $\Theta = \{\theta_p\}_{p=1}^n$ and D_p is training examples for the p -th attribute. We introduce how to generate them in Section 6.3.2.

Put It All Together

So far, we have introduced a recommender to exploit the knowledge from a source domain and a privacy attacker to infer user private information from the transferred knowledge. To fulfill the

Problem 1 in Section 6.2, we need to achieve two goals: improving the target recommendation and protecting the source privacy. In this section, we propose a novel model (PrivNet) by exploiting the attacker component to regularize the recommender.

Since we have two rival objectives (i.e., target quality and source privacy), we adopt the adversarial learning technique [36] to learn a privacy-aware transfer model. The generator is a privacy attacker which tries to accurately infer the user privacy, while the discriminator is an recommender which learns user preferences and deceives the adversary. The recommender of PrivNet minimizes:

$$\tilde{\mathcal{L}}(\theta) = \mathcal{L}(\theta) - \lambda \mathcal{L}(\Theta), \quad (6.10)$$

where the hyperparameter λ controls the influence from the attacker component. PrivNet seeks to improve the recommendation quality (the first term on the right-hand side) and fools the adversary by maximizing the loss of the adversary (the second term,). The adversary has no control over the transferred knowledge, i.e., $\mathbf{x}_{u|trans}$. Losses of the two components are interdependent but they optimize their own parameters. PrivNet is a general framework since both the recommender and the attacker can be easily replaced by their variants. PrivNet reduces to privacy-agnostic transfer model when $\lambda = 0$.

We also reformulate the objective function of recommendation systems in a min-max game way as minimizing the private attacker's gain as well as the recommendation loss simultaneously:

$$\underbrace{\min_{\theta=\{\theta_T, \theta_S, \{\mathbf{H}^\ell\}_{\ell=1}^{L-1}\}} \left(\mathcal{L}(\theta) - \lambda \overbrace{\max_{\Theta=\{\theta_p\}_{p=1}^n} \mathcal{L}(\Theta)}^{\text{private attribute attacker}} \right)}_{\text{privacy-aware transfer learning in recommendation system}} \quad (6.11)$$

We further clarify the PrivNet model from the perspective of the above min-max optimization objective. Firstly, PrivNet tries to learn the transferred knowledge in the source domain by optimizing the parameters in the source space. Secondly, the attacker can happen anywhere including in the target domain, that is the target domain may try to recover the private attributes in the source domain from the transferred knowledge. Together, PrivNet is to achieve such a goal: learning a privacy-aware transfer representation which has the best transfer gain from the space of the least privacy leakage. The hyperparameter controls the balance on whether PrivNet prefers more transfer gain benefiting target domain or prefers less privacy leakage for source domain.

Algorithm 2: Training PrivNet.

Input: Target data D_T , source data D_S , privacy data D_{priv} , hyperparameter λ

Output: PrivNet

for number of training iterations **do**

1. Accumulate (user, attributes) with a mini-batch $(\mathcal{U}_b, \mathcal{Y}_b)$ from D_{priv}
2. Feed users $\mathcal{U}_b$ and their history in D_S into the source network (see Sec. 6.2.2) so as to generate the transferred knowledge $\mathcal{X}_{b|S}$
3. Update Θ using examples $(\mathcal{X}_{b|S}, \mathcal{Y}_b)$ via gradient descent over $\mathcal{L}(\Theta)$.
4. Update θ using mini-batch examples from D_S and D_T with adversary loss via gradient descent over $\tilde{\mathcal{L}}(\theta)$.

end

The gradient-based updates can use any standard gradient-based learning rule. Deep learning library (e.g., TensorFlow) can automatically calculate gradients.

6.3.2 Objective and Optimization

Generating Training Examples

We generate D_T and D_S as follows and take the target domain as an example since the procedure is the same for the source domain. Suppose we have a whole item interaction history for some user u , say $[i_1, i_2, \dots, i_l]$. Then we generate the positive training examples by sliding over the sequence of the history:

$$D_T^+ = \{([i_w]_{w=1}^{c-1}, i_c) : c = 2, \dots, l\}. \quad (6.12)$$

We adopt the random negative sampling technique [92] to generate the corresponding negative training examples:

$$D_T^- = \{([i_w]_{w=1}^{c-1}, i'_c) : i'_c \notin [i_1, i_2, \dots, i_l]\}. \quad (6.13)$$

As the same with [5, 122], we assume that some users $\mathcal{U}^{pub} \subset \mathcal{U}$ share their private attributes with the public profile. Then we have the labelled privacy data:

$$D_{priv} = \{D_p\}_{p=1}^n \text{ where } D_p = \{(u, y_{u,p}) : u \in \mathcal{U}^{pub}\}. \quad (6.14)$$

Model Learning

The training process of PrivNet is illustrated in Algorithm 2.

Lines 1-3 are to optimize the privacy part related parameter, i.e., Θ in $\mathcal{L}(\Theta)$. On line 1, it creates a mini-batch size examples from data D_{priv} . Each example contains a user and their

corresponding private attributes $(u, \{y_{u,p}\}_{p=1}^n)$. On line 2, it feeds users and their historical items in the source domain to the source network so as to generate transferred knowledge $\mathbf{x}_{u|trans}$. On line 3, the transferred knowledge and their corresponding private attributes $(\mathbf{x}_{u|trans}, \{y_{u,p}\}_{p=1}^n)$ are used to train the privacy attacker component by descending its stochastic gradient using the mini-batch examples:

$$\nabla_{\Theta} \mathcal{L}(\Theta).$$

Line 4 is to optimize the recommender part related parameter, i.e., θ by descending its stochastic gradient with adversary loss using mini-batch examples:

$$\nabla_{\theta} \tilde{\mathcal{L}}(\theta).$$

6.3.3 Complexity Analysis

The parameter complexity of PrivNet is the addition of its recommender component and the privacy component. The embedding matrices of the recommender dominate the number of parameters as they vary with the input. As a result, the parameter complexity of PrivNet is $\mathcal{O}(d \cdot (n_S + n_T))$ where d is the embedding dimension, and n_S and n_T are the number of items in the source and target domains respectively.

The learning complexity of PrivNet divides into two parts: the forward prediction and backward parameter update. The forward prediction of PrivNet is the addition of its recommender component and two times of the privacy component since the recommender component needs the loss from the privacy component. The complexity of backward parameter update is the addition of its recommender component and the privacy component since they optimize their own parameters.

6.4 Experiments

In this section, we conduct experiments to evaluate both recommendation performance and privacy protection of PrivNet.

6.4.1 Datasets and Evaluation Protocols

Datasets

We evaluate on the following real-world datasets.

Data	#user	Target domain		Source domain		Private attribute
		#item	#rating	#item	#rating	
Foursquare	29,515	28,199	357,553	28,407	467,810	G
MovieLens	5,967	2,049	274,115	1,484	299,830	G, A

Table 6.1: Statistics of datasets. (G=Gender, A=Age)

- *Foursquare (FS)* It is a public available data on user-venue checkins [135]. The source and target domains are divided by the checkin’s time, i.e., dealing with the covariate shift issues where the distribution of the input variables change between the old data and the newly collected one. The private user attribute is Gender.
- *MovieLens (ML)* It is a public available data on user-movie ratings [39]. We reserve those ratings over three stars as positive feedbacks. The source and target domains are divided by the movie’s release year, i.e., transferring from old movies to the new ones. The private user attributes are Gender and Age. Following [5], we categorize Age into three groups: over-45, under-35, and between 35 and 45.

The statistics are summarized in Table 6.1 and we can see that all of the datasets have more than 99% sparsity. It is expected that the transfer learning technique is helpful to alleviate the data sparsity issues in these real-world recommendation services.

Evaluation Metric

For privacy evaluation, we follow the protocol in [53] to randomly sample 80% of users as the training set and treat the remaining users as the test set. The users in the training set has publicly shown their private information while the users in the test set keep it private. We split a small data from the training set as the validation set where the ratio is train:valid:test=7:1:2. For privacy metrics, we compute Precision, Recall, and F1-score in a weighted² way which are suitable for imbalanced data distribution [26]. We report results for each private attribute. We first calculate metrics for each label, and then compute their average weighted by support (the number of true instances for each label). A lower value indicates better privacy protection.

For recommendation evaluation, we follow the leave-one-out strategy in [42], i.e., reserving the latest one interaction as the test item for each user, then randomly sampling a number of (e.g.,

²Note, the weighted F1 values are not necessarily equal to the harmonic mean of the corresponding Precision and Recall values.

Hyperparameter	Setting
train:valid:test	7:1:2
user representation size	80
item representation size	80
history length cutoff (#items)	10
neural collaborative filtering layers	[80, 64]
attention unit layers	[80, 64]
number of transfer layers	1
negative sampling ratio for training	1
test positive:negative	1:99
clip norm	5
batch size	128
bias init	0
weight init	Glorot uniform
embedding init	Glorot uniform
learning rate	5e-4
optimizer	Adam
activation function	sigmoid
total epochs (with early stopping)	50

Table 6.2: Setting of hyperparameters.

99) negative items that are not interacted by the user. We evaluate how well the recommender can rank the test item against these negative ones. We split a small data from the training set as the validation set where the ratio is train:valid:test=7:1:2. For recommendation metrics, we compute hit ratio (HR), normalized discounted cumulative gain (NDCG), mean reciprocal rank (MRR), and AUC for top- K (default $K = 10$) item recommendation [29]. A higher value indicates better recommendation.

Implementation

All methods are implemented using TensorFlow. Parameters are initialized by default. The optimizer is the adaptive moment estimation with learning rate 5e-4. The size of mini-batch is 128 with negative sampling ratio 1. The embedding size is 80 while the MLP has one hidden layer with size 64. The history size is 10. λ is 1 in Eq. (6.10). The noise level is 10%. The number of dummy items are 5. The privacy related metrics are computed by Python scikit-learn library. The setting of hyperparameters used to train our model and the baselines is summarized in Table 6.2.

Methods	Knowledge transfer	Privacy protection (+strategy)
BPRMF (Rendle et al, 2009) [105]	✗	✗
MLP (He et al, 2017) [42]	✗	✗
CSN (Misra et al, 2016) [83]	✓	✗
CoNet (Hu et al, 2018) [43]	✓	✗
BlurMe (Weinsberg et al, 2012) [122]	✗	✓ (+perturbation)
LDP (Bassily and Smith, 2015) [4]	✗	✓ (+noise)
PrivNet (ours)	✓	✓ (+adversary)

Table 6.3: Categorization of comparing methods.

6.4.2 Baseline

We compare PrivNet with various kinds of baselines as summarized in Table 6.3.

The following methods are privacy-agnostic.

- *BPRMF*: Bayesian personalized ranking [105] is a latent factors approach which learns user and item factors via matrix factorization.
- *MLP*: Multilayer perceptron [42] is a neural CF approach which learns the user-item interaction function using neural networks.
- *CSN*: The cross-stitch network [83] is a deep transfer learning model which couples the two basic networks via a linear combination of activation maps using a translation scalar.
- *CoNet*: Collaborative cross network [43] is a deep transfer learning method for cross-domain recommendation which learns linear combination of activation maps using a translation matrix.

The following methods are privacy-aware.

- *BlurMe*: This method [122] perturbs a user’s profile by adding dummy items to their history. It is a representative of the perturbation-based technique to recommend items while protect private attributes.
- *LDP*: Local differential privacy [4] modifies user-item ratings by adding noise to them based on the differential privacy. It is a representative of the noise-based technique to recommend items while protect private attributes. Note, the original LDP and BlurMe are single-domain models which are also used as comparing baselines in [5].

Dataset	Metric	BPRMF	MLP	CSN	CoNet	BlurMe	LDP	PrivNet
Foursquare	HR	36.5	47.0	52.7	53.4*	52.6	44.5	54.3
	NDCG	22.0	31.5	35.9	36.3*	35.4	29.9	36.8
	MRR	17.6	31.9	35.0*	35.3	32.1	27.1	33.4
MovieLens	HR	53.0	77.4	82.7	77.1	85.7	85.8*	86.0
	NDCG	37.0	50.5	55.7	50.7	69.7	69.9	69.9
	MRR	32.0	44.5	49.3	44.6	65.9	65.9	65.7*

Table 6.4: Comparison results of different methods on recommendation performance. The bold face indicates the best result while the star mark indicates the second best.

To be fair and to investigate the influence of privacy-preserving strategies, we replace the adversary strategy of PrivNet with the strategy of LDP (adding noise) and BlurMe (perturbing ratings), and keep the other components the same.

6.4.3 Results: Comparing Recommendation Performance

The results of different methods on recommendation are summarized in Table 6.4. A higher value indicates better recommendation performance.

Comparing with the privacy-agnostic methods (BPRMF, MLP, CSN, and CoNet), PrivNet is superior than them with a large margin on the MovieLens dataset. This shows that PrivNet is effective in recommendation while it protects the source private attributes. Since these four methods represent a wide range of typical recommendation methods (matrix factorization, neural CF, transfer learning), we can see that the architecture of PrivNet is a reasonable design for recommender systems.

Comparing with the privacy-aware methods (LDP and BlurMe), we can see that LDP significantly degrades recommendation performance with a reduction about six to ten percentage points on the Foursquare dataset. This shows that LDP suffers from the noisy source information since it harms the usefulness of the transferred knowledge to the target task. For BlurMe, we can see that BlurMe still degrades recommendation performance on the Foursquare dataset, for example with relative 4.0% performance reduction in terms of MRR. This shows that BlurMe suffers from the perturbed source information since it harms the usefulness of the transferred knowledge to the target task.

Among the privacy-aware methods, PrivNet achieves the best recommendation performance in terms of all HR, NDCG, and MRR on the Foursquare dataset, and the best in terms of HR

Dataset	Metric	LDP	BlurMe	PrivNet
Foursquare	Precision	64.7	73.2	66.8
	Recall	75.2	75.3	71.1
	F1	66.0	66.7	68.1
MovieLens-G	Precision	73.4	69.4	70.9
	Recall	75.4	71.7	72.5
	F1	73.6	70.1	62.0
MovieLens-A	Precision	63.8	54.6	55.4
	Recall	65.5	58.1	57.9
	F1	61.4	54.2	46.3

Table 6.5: Comparison results on privacy protection. The bold face indicates the best result (the lower the better).

on the MovieLens dataset. It shows that PrivNet is better for improving the usefulness of the transferred knowledge by comparing with LDP and BlurMe.

In summary, PrivNet is effective in transferring the knowledge, showing that the adversary strategy of PrivNet achieves state-of-the-art performance by comparing with the strategies of adding noise (LDP) and perturbing ratings (BlurMe).

6.4.4 Results: Comparing Privacy Protection

The results of different methods on privacy inference are summarized in Table 6.5 (Note, there are no results for the four privacy-agnostic methods). A lower value indicates better privacy protection.

Comparing PrivNet and BlurMe, we can see that the perturbation method by adding dummy items still suffers from privacy inference attacks in terms of Precision and Recall on the Foursquare dataset, and in terms of F1 on the MovieLens dataset. The reason may be that the attacker can effectively distinguish the true profiles from the dummy items. That is, it can accurately learn from the true profiles while ignore the dummy items. Comparing PrivNet and LDP, we can see that adding noise to ratings still suffers from privacy inference attacks in terms of Recall on the Foursquare dataset, and in terms of all three metrics on the MovieLens dataset. It implies that the occurrence of a rating, regardless of its numeric value (true or noisy), leaks the user privacy. That is, the binary event of excluding or including an item in a user’s profile is a signal for user privacy inference nearly as strong as numerical ratings. In particular, there are 50 movies rated by Female only (e.g., Country Life (1994)) while 350 by Male only (e.g., Time

Adversary loss?	Foursquare			MovieLens-Gender			MovieLens-Age		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
No	73.2	75.4	68.5	73.6	75.6	73.5	60.8	65.3	61.4
Yes	66.8	71.1	68.1	70.9	72.5	62.0	55.4	57.9	46.3

Table 6.6: Necessity of adversary loss to regularize the recommender (lower value better privacy protection).

Masters (1982)). Adding noise to these ratings may not influence the inference of Gender for these users very much.

PrivNet achieves nearly half the best results on privacy protection in terms of three evaluation metrics on the two datasets. It has significantly lower F1 scores in comparison to all baselines on the MovieLens dataset. It is effective to hide private information during the knowledge transfer. By simulating the attacks during the training, PrivNet is prepared against the malicious attacks for unseen users in the future. In summary, PrivNet is an effective source privacy-aware transfer model such that it makes the malicious attackers more difficult to infer the source user privacy during the knowledge transfer, compared with the strategies of adding noise (LDP) and perturbing ratings (BlurMe).

6.4.5 Analyses

Clustering on Transferable Representations

Figure 6.1 shows t-SNE projections of 4,726 users' transferred representations on the MovieLens-Gender dataset. These user vectors are computed from the user encoder as shown in Figure 6.2. We can see that the vectors are more mixed distributed among male and female users with the training of PrivNet. In contract, the vectors for female users are clustered on the top-left corner while male users are on the bottom-right without the training of PrivNet ($\lambda = 0$, see Section 6.4.5). To quantify the difference, we perform K-means clustering on the user vectors where $K=2$, and calculate the V-measure [107] which assesses the degree of overlap between the 2 clusters and the Gender groups. The measure is 0.0119 and 0.0027 respectively for without and with training of PrivNet. Note that a lower measure is better since we do not want to the two classes to be easily separable.

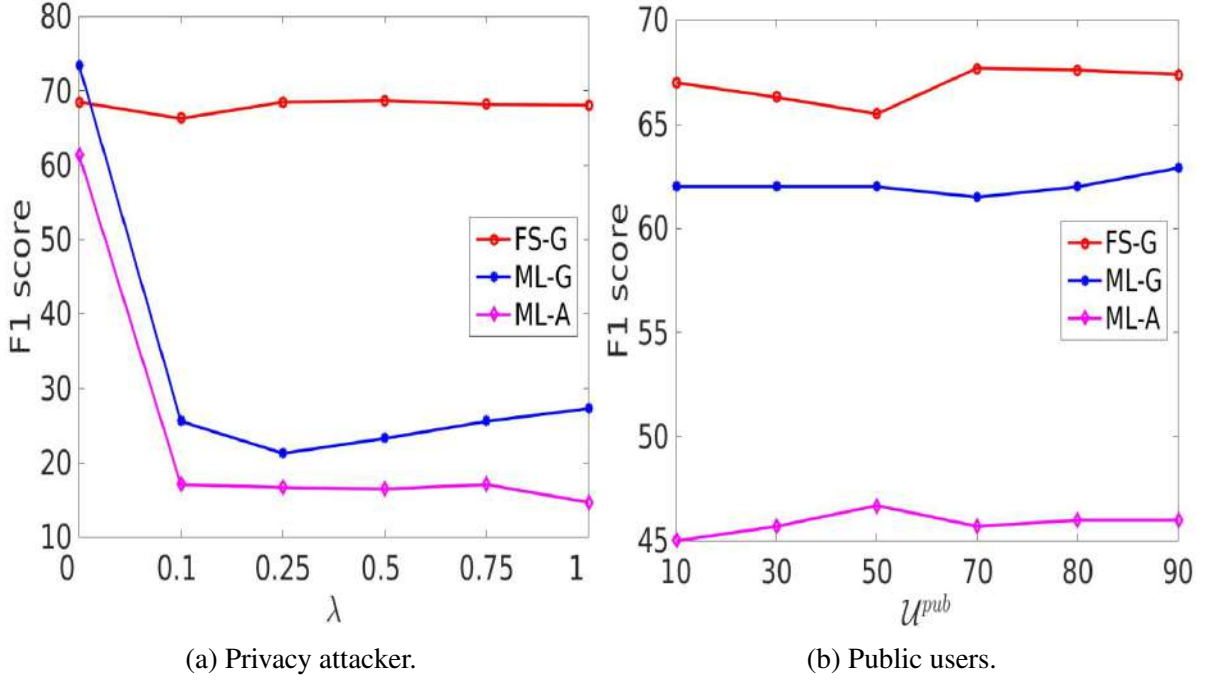


Figure 6.3: Impact of privacy component and public users. (FS-G: Foursquare-Gender, ML-G: MovieLens-Gender, ML-A: MovieLens-Age)

Model Ablation

The key component of PrivNet is the adversary loss used to regularize the recommender. We remove this component to show its necessity to protect the private attributes by setting the $\lambda = 0$ in Eq. (6.10). The results are summarized in Table 6.6. As we expect, PrivNet without adversary loss is most vulnerable to privacy attacks since it has no privacy defense. There is a significant drop in terms of all three privacy-related metrics without this model component.

Impact of Privacy Component

We vary the λ (see Eq. (6.10)) of privacy component with $\{0, 0.1, 0.25, 0.5, 0.75, 1.0\}$ to show the its impact on privacy protection and recommendation (where $\lambda = 0$ corresponds to without privacy attack component, see also Table 6.6). Figure 6.3a shows the impact on privacy protection. The privacy inference generally becomes more difficult with the increase of λ , showing that the privacy inference component of PrivNet is a key factor for protecting the user privacy in the source domain. In particular, all results of $\lambda \neq 0$ are better than that of $\lambda = 0$ in hiding the private information. Privacy inference results, however, are subtle among different private attributes and evaluation metrics. On the Foursquare dataset, F1 decreases at first (until λ to 0.1), then it increases. On the MovieLens-Gender dataset, the F1 score decreases at first (until λ to

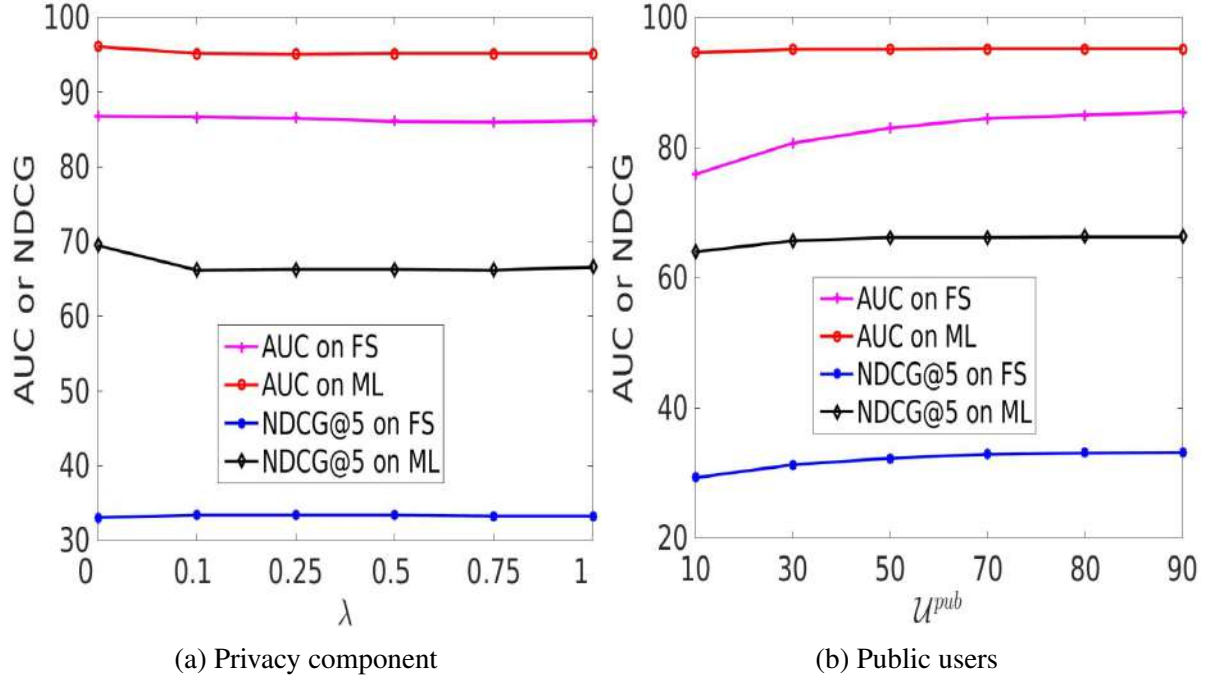


Figure 6.4: Parameter sensitivity for recommendation.

0.25) and then it increases. It means that the private information is obscured more successfully in the beginning but less in the end. The reason may be that the model overfits by increasing the value of λ and leads to an inaccurate estimation of privacy inference. On the MovieLens-Age dataset, the F1 score consistently decreases with the increase of λ .

Figure 6.4a shows the impact on recommendation performance. The recommendation performance decreases with λ increasing from 0 to 0.1 on the MovieLens dataset, showing that increasing the impact of privacy inference component harms the recommendation quality to some extent.

Impact of Public Users

We vary the percentage of public users $\mathcal{U}^{pub}$ (see Section 6.2) with $\{10, 30, 50, 70, 80, 90\}$. Figure 6.3b shows the impact on the privacy inference. It is surprising that the privacy inference does not become more easy with the increase of public users. On the Foursquare dataset, it infers inaccurately until the percentage increases to 50% and then accurately until to 80% in terms of F1. This shows that the adversary strategy of PrivNet is effective to protect unseen users' privacy when only a small number of users (e.g., 10%) reveal their profiles for the training. On the MovieLens dataset, it infers inaccurately after 50% until to 80% in terms of F1.

Figure 6.4b shows the impact on recommendation performance. Since the amount of public

No.	Movie	Genre	Attn weight
0	Chicken Run	Animation, Children, Comedy	0.127
1	X-Men	Action, Sci-Fi	0.069
2	Mission: Impossible	Action, Adventure, Mystery	0.001
3	Titan A.E.	Adventure, Animation, Sci-Fi	0.059
4	The Perfect Storm	Action, Adventure, Thriller	0.056
5	Gone in 60 Seconds	Action, Crime	0.053
6	Schindler's List	Drama, War	0.098
7	The Shawshank Redemption	Drama	0.331
8	The Matrix	Action, Sci-Fi, Thriller	0.062
9	Shakespeare in Love	Comedy, Romance	0.140
10	Howards End	Drama	N/A

Table 6.7: Example: Capturing short-/long-term user interests and high-level category relationship among items. No. 10 is the candidate new articles to be recommended while No. 0 to No. 9 are the historical news articles. Attn weight: Attention weight.

users controls how much knowledge is shared between the source and target domains, the recommendation performance improves with the increasing amount of public users. In summary, PrivNet is favourable in practice since it can achieve a good tradeoff on the utility and privacy when only a small amount of users reveal their profiles to the public.

Case Study

One advantage of PrivNet is that it can explain which item in a user's history matters the most for a candidate item by using the attention weights. Table 6.7 shows an example of interactions between a user's historical movies (No. 0~9) and the candidate movie (No. 10). We can see that the latest movie matters a lot since the user interests may remain the same during a short period. The oldest movie, however, also has some impact on the candidate movie, reflecting that the user interests may mix with a long-term characteristic. PrivNet can capture these subtle short-/long-term user interests. Furthermore, the movie (No. 7) belonging to the same genre as the candidate movie matters the most. PrivNet can also capture this high-level category relationship.

6.5 Related Work

Transfer learning in recommendation Transfer learning in recommendation [9] is an effective technique to alleviate the data sparsity issue in one domain by exploiting the knowledge from

other domains. Typical methods apply matrix factorization [95, 111, 133] and representation learning [29, 75, 77, 132, 143] on each domain and share the user (item) factors, or learn a cluster level rating pattern [61, 142]. Transfer learning is to improve the target performance by exploiting knowledge from auxiliary domains [13, 24, 30, 93, 144]. One transfer strategy (two-stage) is to initialize a target network with transferred representations from a pre-trained source network [90, 140]. Another transfer strategy (end-to-end) is to transfer knowledge in a mutual way such that the source and target networks benefit from each other during the training, with examples including the cross-stitch networks [83] and collaborative cross networks [43]. These transfer learning methods have access to the input or representations from source domain. Therefore, it raises a concern on privacy leaks and provides an attack possibility during knowledge transfer.

Privacy-preserving techniques Existing privacy-preserving techniques mainly belong to three research threads. One thread adds noise (e.g., differential privacy [22]) to the released data or the output of recommender systems [53, 81, 82, 119, 121]. One thread perturbs user profiles such as adding (or deleting/changing) dummy items to the user history so that it hides the user’s actual ratings [100, 122]. Adding noise and perturbing ratings may still suffer from privacy inference attacks when the attacker can successfully distinguish the true profiles from the noisy/perturbed ones. Furthermore, they may degrade performance since data is corrupted. The privacy-preserving stacking technique [38, 139] is proposed to enhance a logistic regression model by ensemble learning such that the two goals are achieved: protecting privacy and improving performance. Furthermore, a real-world cross-organization health data is used to evaluate the transfer learning integrated with privacy guarantee where the user privacy is of a significant concern in such cases. Another thread uses adversary loss [5, 106] to formulate the privacy attacker and the recommender system as an adversarial learning problem. However, they face the data sparsity issues. A recent work [103] trains linear classifiers to predict a protected attribute and then remove it by projecting the representation on its null-space. Since the noise is introduced in the differential privacy, it faces the performance deterioration issue.

Federated learning Since the datasets are usually existing in different platforms, different organizations, and different websites, the data privacy and abuse are raised when we try to fuse them to train a recommender system. The federated learning (FL) is recently proposed as an effective solution for tackling such issue by allowing knowledge to be shared and not compromising user privacy. We refer readers to [137] for general concepts of federated learning

where a comprehensive survey is provided. Some recommendation methods have adopted federated learning so as to protect the personal data without affecting performance [12, 14, 118]. Here we focus on the applications of federating learning into recommendation in terms of three aspects: horizontal, vertical, and transferable federated learning. Horizontal FL, or sample-based FL, is applicable in the recommendation scenario where datasets have same/overlapped attributes (e.g. items) but different users. When we treat each user as an example and want to exploit all users feedbacks on their local devices, then horizontal FL scene is investigated in the Google Android application [57]. Vertical FL, or feature-based FL, is applicable in the recommendation scenario where datasets have shared/overlapped users but different attributes. [31, 112] proposed a model which adopts vertical FL into matrix factorization recommendation technique. In their setting, the users are shared for both domains while an auxiliary domain has extra information for users called user-associated attributes. Since this model needs to know the alignment of users between domains, the user identity faces the risk of privacy leakage. Another drawback of these methods is that they suffer from efficiency and scalability due to high cost of encryption computation and communication over channels.

6.6 Conclusion

We presented an attack scenario to infer the private user attributes from the transferred knowledge in recommendation, raising the issues of source privacy leakage beyond target performance. To protect user privacy in the source domain, a privacy-aware transfer model (PrivNet) is proposed beyond improving the performance in the target domain. It is effective in terms of recommendation performance and privacy protection, achieving a good trade-off between the utility and privacy of the transferred knowledge. In future works, we want to relax the assumption that the private user attributes need to provide in advance in order to train the privacy inference component for protecting unseen users.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Deep transfer learning in recommendation is a novel and exciting research area, which can greatly improve user information need as well as bring in much profit for real industry applications. In this thesis, we have conducted the following works:

1. We survey two typical recommendation techniques, i.e., shallow and deep learning based methods, and three typical transfer learning approaches, i.e., model-based transfer, instance-based transfer and feature-based transfer.
2. We divide the deep knowledge transfer in recommendation into two plug-in parts, one is the base network and the other is transfer unit. We describe two kinds of base networks and three kinds of transfer units. They can be easily matched with each other.
3. We propose three new problem settings including cross-dataset, cross-domain, and hybrid sources recommendation, and then design our deep transfer learning solutions correspondingly,
 - Deep model-based transfer for cross-dataset recommendation via Collaborative Cross Network (CoNet),
 - Deep instance-based transfer for cross-domain recommendation via TransNet,
 - Deep feature-based transfer for hybrid sources recommendation via TrNews.
4. We are the very first to protect source user privacy during the knowledge transfer from the source domain to the target domain, with focus not only on improving target performance

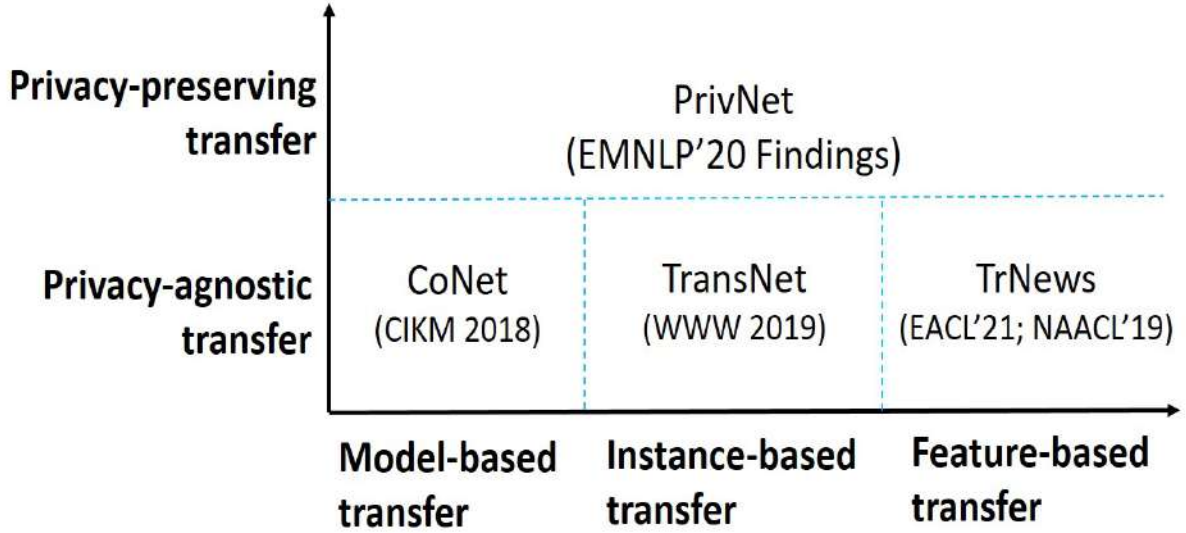


Figure 7.1: Summary of Our Deep and Adversarial Transfer Learning Models in Recommendation.

but also on privacy-aware recommendation. We introduce a practical attacker scene and then design an adversarial learning algorithm to model the recommender and the attacker,

- Adversarial transfer learning for protecting private attributes in recommendation by learning a privacy-aware transferable representation via PrivNet,

According to the first dimension of “what to transfer” and “how to transfer” in transfer learning, and the second dimension of privacy-preserving transfer and privacy-agnostic transfer, we summarize our work as shown in Figure 7.1.

7.2 Future Work

In this section, we illustrate several promising potential directions for research within the field of deep and adversarial transfer learning in recommendation. The first one is on negative transfer. As we have shown in the deep feature-based transfer learning (TrNews), the transfer model may face the challenge of negative transfer where a simple architecture may be better than a complex one. A detailed investigation is needed to be done since the risk of overfitting is a confounding factor. The second one is on privacy protection. In our current work, we need to define what attribute is to be protected. That is, we differentiate the importance of different attributes. In reality, we may not know the relative importance of different attributes, or all attributes are needed to be protected. More deeper work is worth conducting for these new scenarios.

7.3 List of Publications

The list of publications during pursuing my PhD degree is given as follows.

Conference:

1. "TrNews: Heterogeneous User-Interest Transfer Learning for News Recommendation",
Guangneng Hu & Qiang Yang,
European Chapter of the Association for Computational Linguistics (EACL) 2021, Pages 734–744, Association for Computational Linguistics, Online
2. "PrivNet: Safeguarding Private Attributes in Transfer Learning for Recommendation",
Guangneng Hu & Qiang Yang,
Conference on Empirical Methods in Natural Language Processing (EMNLP) 2020 Findings, Pages 4506–4516, Association for Computational Linguistics, Online
3. "Transfer Meets Hybrid: A Synthetic Approach for Cross-Domain Collaborative Filtering with Text",
Guangneng Hu, Yu Zhang & Qiang Yang,
The Web Conference (WWW) 2019, Pages 2822–2829, Association for Computing Machinery, San Francisco, USA
4. "CoNet: Collaborative Cross Networks for Cross-Domain Recommendation",
Guangneng Hu, Yu Zhang & Qiang Yang,
Conference on Information and Knowledge Management (CIKM) 2018, Pages 667–676, Association for Computing Machinery, Torino, Italy
5. "Integrating Reviews into Personalized Ranking for Cold Start Recommendation",
Guangneng Hu & Xin-Yu Dai,
Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD) 2017, Pages 708–720, Springer International Publishing, Jeju, South Korea
6. "Personalized Neural Embeddings for Collaborative Filtering with Text",
Guangneng Hu,

North American Chapter of the Association for Computational Linguistics (NAACL) 2019,
Pages 2082–2088, Association for Computational Linguistics, Minneapolis, USA

7. "Dual Side Deep Context-aware Modulation for Social Recommendation",
Bairan Fu, Wenming Zhang, **Guangneng Hu**, Xinyu Dai, Shujian Huang & Jiajun Chen,
The Web Conference (WWW) 2021, Association for Computing Machinery, Online

Journal:

1. "Collaborative Filtering with Topic and Social Latent Factors Incorporating Implicit Feedback",
Guangneng Hu, Xin-Yu Dai, Fengyu Qiu, Rui Xia, Tao Li, Shujian Huang & Jiajun Chen,
ACM Transactions on Knowledge Discovery from Data (TKDD) 2018, Volume 12, No. 2,
Article 23

Bibliography

- [1] Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749.
- [2] An, M., Wu, F., Wu, C., Zhang, K., Liu, Z., and Xie, X. (2019). Neural news recommendation with long-and short-term user representations. In *Proceedings of Annual Meeting of Association for Computational Linguistics*, pages 336–345.
- [3] Barjasteh, I., Forsati, R., Masrour, F., Esfahanian, A.-H., and Radha, H. (2015). Cold-start item and user recommendation with decoupled completion and transduction. In *Proceedings of ACM Conference on Recommender Systems*.
- [4] Bassily, R. and Smith, A. (2015). Local, private, efficient protocols for succinct histograms. In *ACM STOC*.
- [5] Beigi, G., Mosallanezhad, A., Guo, R., Alvani, H., et al. (2020). Privacy-aware recommendation with private-attribute protection using adversarial learning. In *ACM WSDM*.
- [6] Ben-David, S., Blitzer, J., Crammer, K., Pereira, F., et al. (2007). Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137.
- [7] Bennett, J., Lanning, S., et al. (2007). The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. Citeseer.
- [8] Berkovsky, S., Kuflik, T., and Ricci, F. (2007). Cross-domain mediation in collaborative filtering. In *UMAP*.
- [9] Cantador, I., Fernández-Tobías, I., Berkovsky, S., and Cremonesi, P. (2015). Cross-domain recommender systems. In *Recommender Systems Handbook*.

- [10] Caruana, R. (1997). Multitask learning. *Machine Learning*.
- [11] Catherine, R. and Cohen, W. (2017). Transnets: Learning to transform for recommendation. In *ACM RecSys*.
- [12] Chai, D., Wang, L., Chen, K., and Yang, Q. (2020). Secure federated matrix factorization. *IEEE Intelligent Systems*.
- [13] Chen, C., Zhang, M., Wang, C., Ma, W., Li, M., Liu, Y., and Ma, S. (2019). An efficient adaptive transfer neural network for social-aware recommendation. In *ACM SIGIR*.
- [14] Chen, F., Dong, Z., Li, Z., and He, X. (2018). Federated meta-learning for recommendation. *arXiv:1802.07876*.
- [15] Cheng, H.-T., Koc, L., Harmsen, J., et al. (2016). Wide & deep learning for recommender systems. In *ACM Recsys Workshop*.
- [16] Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*.
- [17] Covington, P., Adams, J., and Sargin, E. (2016). Deep neural networks for youtube recommendations. In *RecSys*.
- [18] Das, A. S., Datar, M., Garg, A., and Rajaram, S. (2007). Google news personalization: scalable online collaborative filtering. In *Proceedings of World Wide Web*, pages 271–280.
- [19] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.
- [20] Deshpande, M. and Karypis, G. (2004). Item-based top-n recommendation algorithms.
- [21] Doersch, C. and Zisserman, A. (2017). Multi-task self-supervised visual learning. In *IEEE CVPR*.
- [22] Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*.
- [23] Dziugaite, G. and Roy, D. (2015). Neural network matrix factorization.

- [24] Elkahky, A., Song, Y., and He, X. (2015a). A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *WWW*.
- [25] Elkahky, A., Song, Y., and He, X. (2015b). A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *WWW*.
- [26] Fawcett, T. (2006). An introduction to roc analysis. *Pattern recognition letters*.
- [27] Felício, C. Z., Paixão, K. V., Barcelos, C. A., and Preux, P. (2017). A multi-armed bandit model selection for cold-start user recommendation. In *Proceedings of Conference on User Modeling, Adaptation and Personalization*, pages 32–40.
- [28] Fu, W., Peng, Z., Wang, S., Xu, Y., and Li, J. (2019). Deeply fusing reviews and contents for cold start users in cross-domain recommendation systems. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 94–101.
- [29] Gao, C., Chen, X., Feng, F., Zhao, K., et al. (2019a). Cross-domain recommendation without sharing user-relevant data. In *WWW*.
- [30] Gao, C., He, X., Gan, D., Chen, X., Feng, F., Li, Y., Chua, T.-S., and Jin, D. (2019b). Neural multi-task recommendation from multi-behavior data. In *IEEE ICDE*.
- [31] Gao, D., Tan, B., Ju, C., Zheng, V. W., and Yang, Q. (2020). Privacy threats against federated matrix factorization. *IJCAI 2020 Workshop on Federated Learning for Data Privacy and Confidentiality*.
- [32] Gao, W., Tian, Y., Huang, T., and Yang, Q. (2010). Vlogging: A survey of videoblogging technology on the web. *ACM Computing Surveys*.
- [33] Ge, W. and Yu, Y. (2017). Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1086–1095.
- [34] Glorot, X., Bordes, A., and Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of international conference on machine learning*.
- [35] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

- [36] Goodfellow, I., Pouget, J., Mirza, M., Xu, B., et al. (2014). Generative adversarial nets. In *NIPS*.
- [37] Guo, H., Tang, R., Ye, Y., Li, Z., and He, X. (2017). Deepfm: a factorization-machine based neural network for ctr prediction. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 1725–1731.
- [38] Guo, X., Yao, Q., Kwok, J., Tu, W., Chen, Y., Dai, W., and Yang, Q. (2020). *Privacy-Preserving Stacking with Application to Cross-organizational Diabetes Prediction*, pages 269–283. Springer International Publishing.
- [39] Harper, F. and Konstan, J. (2016). The movielens datasets: History and context. *ACM TIST*.
- [40] He, M., Zhang, J., Yang, P., and Yao, K. (2018). Robust transfer learning for cross-domain collaborative filtering using multiple rating patterns approximation. In *WSDM '18*, pages 225–233. ACM.
- [41] He, R. and McAuley, J. (2016). Vbpr: visual bayesian personalized ranking from implicit feedback. In *AAAI*.
- [42] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017). Neural collaborative filtering. In *WWW*.
- [43] Hu, G., Zhang, Y., and Yang, Q. (2018a). Conet: Collaborative cross networks for cross-domain recommendation. *CIKM*.
- [44] Hu, G., Zhang, Y., and Yang, Q. (2019). Transfer meets hybrid: a synthetic approach for cross-domain collaborative filtering with text. In *The World Wide Web Conference*, pages 2822–2829.
- [45] Hu, G.-N. and Dai, X.-Y. (2017). Integrating reviews into personalized ranking for cold start recommendation. In *PAKDD*.
- [46] Hu, G.-N., Dai, X.-Y., Qiu, F.-Y., Xia, R., Li, T., Huang, S.-J., and Chen, J.-J. (2018b). Collaborative filtering with topic and social latent factors incorporating implicit feedback. *ACM Transactions on Knowledge Discovery from Data*.
- [47] Hu, G.-N., Dai, X.-Y., Song, Y., Huang, S.-J., and Chen, J.-J. (2015). A synthetic approach for recommendation: Combining ratings, social relations, and reviews. In *IJCAI*.

- [48] Hu, L., Cao, J., Xu, G., Cao, L., Gu, Z., and Zhu, C. (2013). Personalized recommendation via cross-domain triadic factorization. In *WWW*.
- [49] Hu, L., Xu, S., Li, C., Yang, C., Shi, C., Duan, N., Xie, X., and Zhou, M. (2020). Graph neural news recommendation with unsupervised preference disentanglement. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4255–4264. Association for Computational Linguistics.
- [50] Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *IEEE ICDM*.
- [51] Huang, H., Zhang, Q., Gong, Y., and Huang, X. (2016). Hashtag recommendation using end-to-end memory networks with hierarchical attention. In *Proceedings of International Conference on Computational Linguistics*, pages 943–952.
- [52] Huang, Y.-Y. and Lin, S.-D. (2016). Transferring user interests across websites with unstructured text for cold-start recommendation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 805–814.
- [53] Jia, J. and Gong, N. (2018). Attriguard: A practical defense against attribute inference attacks via adversarial machine learning. In *USENIX Security*.
- [54] Kang, S., Hwang, J., Lee, D., and Yu, H. (2019). Semi-supervised learning for cross-domain recommendation to cold-start users. In *Proceedings of ACM International Conference on Information and Knowledge Management*.
- [55] Kim, D., Park, C., Oh, J., Lee, S., and Yu, H. (2016). Convolutional matrix factorization for document context-aware recommendation. In *ACM RecSys*.
- [56] Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization.
- [57] Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.
- [58] Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM.

- [59] Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*.
- [60] Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*.
- [61] Li, B., Yang, Q., and Xue, X. (2009a). Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In *Twenty-First international joint conference on artificial intelligence*.
- [62] Li, B., Yang, Q., and Xue, X. (2009b). Transfer learning for collaborative filtering via a rating-matrix generative model. In *Proceedings of the 26th annual international conference on machine learning*, pages 617–624.
- [63] Li, B., Zhu, X., Li, R., Zhang, C., Xue, X., and Wu, X. (2011). Cross-domain collaborative filtering over time. In *IJCAI*.
- [64] Li, J., Jing, M., Lu, K., Zhu, L., Yang, Y., and Huang, Z. (2019a). From zero-shot learning to cold-start recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4189–4196.
- [65] Li, P. and Tuzhilin, A. (2020). Ddtcdr: Deep dual transfer cross domain recommendation. In *Proceedings of ACM International Conference on Web Search and Data Mining*.
- [66] Li, Y., Baldwin, T., and Cohn, T. (2019b). Semisupervised stochastic multi-domain learning using variational inference. In *Proceedings of Annual Meeting of Association for Computational Linguistics*.
- [67] Liang, D., Altosaar, J., Charlin, L., and Blei, D. M. (2016). 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.
- [68] Liu, B., Fu, Y., Yao, Z., and Xiong, H. (2013). Learning geographical preferences for point-of-interest recommendation. In *ACM SIGKDD*.
- [69] Liu, B., Wei, Y., Zhang, Y., Yan, Z., and Yang, Q. (2018). Transferable contextual bandit for cross-domain recommendation.

- [70] Liu, J., Dolan, P., and Pedersen, E. R. (2010). Personalized news recommendation based on click behavior. In *Proceedings of international conference on Intelligent user interfaces*, pages 31–40.
- [71] Loni, B., Shi, Y., Larson, M., and Hanjalic, A. (2014). Cross-domain collaborative filtering with factorization machines. In *European conference on information retrieval*.
- [72] Lops, P., De Gemmis, M., and Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*.
- [73] Lu, Z., Dou, Z., Lian, J., Xie, X., and Yang, Q. (2015). Content-based collaborative filtering for news topic recommendation. In *Twenty-ninth AAAI conference on artificial intelligence*.
- [74] Lu, Z., Zhong, E., Zhao, L., Xiang, E., Pan, W., and Yang, Q. (2013). Selective transfer learning for cross domain recommendation. In *SIAM International Conference on Data Mining*.
- [75] Ma, M., Ren, P., Lin, Y., Chen, Z., Ma, J., and Rijke, M. d. (2019a). Pi-net: A parallel information-sharing network for shared-account cross-domain sequential recommendations. In *ACM SIGIR*.
- [76] Ma, Y., Zong, L., Yang, Y., and Su, J. (2019b). News2vec: News network embedding with subnode information. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.
- [77] Man, T., Shen, H., Jin, X., and Cheng, X. (2017). Cross-domain recommendation: an embedding and mapping approach. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 2464–2470.
- [78] Marx, V. (2013). The big challenges of big data. *Nature*, 498(7453):255–260.
- [79] McAuley, J. and Leskovec, J. (2013). Hidden factors and hidden topics: understanding rating dimensions with review text. In *ACM RecSys*.
- [80] McMahan, B., Holt, G., Sculley, D., Young, M., Ebner, D., Grady, J., Nie, L., Phillips, T., Davydov, E., Golovin, D., et al. (2013). Ad click prediction: a view from the trenches. In *Proceedings of ACM SIGKDD international conference on Knowledge discovery and data mining*.

- [81] McSherry, F. and Mironov, I. (2009). Differentially private recommender systems: Building privacy into the netflix prize contenders. In *ACM SIGKDD*.
- [82] Meng, X., Wang, S., Shu, K., Li, J., et al. (2018). Personalized privacy-preserving social recommendation. In *AAAI*.
- [83] Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. (2016). Cross-stitch networks for multi-task learning. In *IEEE CVPR*.
- [84] Mnih, A. and Salakhutdinov, R. (2008). Probabilistic matrix factorization. In *NIPS*.
- [85] Nair, V. and Hinton, G. (2010). Rectified linear units improve restricted boltzmann machines. In *ICML*.
- [86] Nguyen, T. and Takasu, A. (2018). Npe: Neural personalized embedding for collaborative filtering. In *IJCAI*.
- [87] Nikolaenko, V., Ioannidis, S., Weinsberg, U., Joye, M., et al. (2013). Privacy-preserving matrix factorization. In *ACM CCS*.
- [88] Ning, X. and Karypis, G. (2011). Slim: Sparse linear methods for top-n recommender systems. In *2011 11th IEEE International Conference on Data Mining*, pages 497–506. IEEE.
- [89] Okura, S., Tagami, Y., Ono, S., and Tajima, A. (2017). Embedding-based news recommendation for millions of users. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- [90] Oquab, M., Bottou, L., Laptev, I., and Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724.
- [91] Pan, R., Zhou, Y., Cao, B., Liu, N., Lukose, R., Scholz, M., and Yang, Q. (2008a). One-class collaborative filtering. In *IEEE ICDM*.
- [92] Pan, R., Zhou, Y., Cao, B., Liu, N. N., Lukose, R., Scholz, M., and Yang, Q. (2008b). One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 502–511. IEEE.

- [93] Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.
- [94] Pan, W., Liu, N., Xiang, E., and Yang, Q. (2011). Transfer learning to predict missing ratings via heterogeneous user feedbacks. In *IJCAI*.
- [95] Pan, W., Xiang, E. W., Liu, N. N., and Yang, Q. (2010). Transfer learning in collaborative filtering for sparsity reduction. In *Twenty-fourth AAAI conference on artificial intelligence*.
- [96] Park, S.-T. and Chu, W. (2009). Pairwise preference regression for cold-start recommendation. In *Proceedings of ACM conference on Recommender systems*, pages 21–28.
- [97] Paterek, A. (2007). Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8.
- [98] Pazzani, M. and Billsus, D. (2007). Content-based recommendation systems. In *The adaptive web*.
- [99] Peng, H., Liu, J., and Lin, C.-Y. (2016). News citation recommendation with implicit and explicit semantics. In *Proceedings of Annual Meeting of Association for Computational Linguistics*.
- [100] Polat, H. and Du, W. (2003). Privacy-preserving collaborative filtering using randomized perturbation techniques. In *IEEE ICDM*.
- [101] Rafailidis, D. and Crestani, F. (2019). Neural attentive cross-domain recommendation. In *Proceedings of ACM SIGIR International Conference on Theory of Information Retrieval*, pages 165–172.
- [102] Ramakrishnan, N., Keller, B., Mirza, B., Grama, A., and Karypis, G. (2001). Privacy risks in recommender systems. *IEEE Internet Computing*.
- [103] Ravfogel, S., Elazar, Y., Gonen, H., Twiton, M., and Goldberg, Y. (2020). Null it out: Guarding protected attributes by iterative nullspace projection. *ACL*.
- [104] Rendle, S. (2012). Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology*.

- [105] Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2009). Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*.
- [106] Resheff, Y., Elazar, Y., Shahar, M., and Shalom, O. (2019). Privacy and fairness in recommender systems via adversarial training of user representations. In *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods*.
- [107] Rosenberg, A. and Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP*.
- [108] Roy, S. D., Mei, T., Zeng, W., and Li, S. (2012). Socialtransfer: cross-domain transfer learning from social streams for media applications. In *Proceedings of ACM international conference on Multimedia*, pages 649–658.
- [109] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.
- [110] Shi, C., Zhang, Z., Luo, P., Yu, P., Yue, Y., and Wu, B. (2015). Semantic path based personalized recommendation on weighted heterogeneous information networks. In *ACM CIKM*.
- [111] Singh, A. and Gordon, G. (2008). Relational learning via collective matrix factorization. In *ACM SIGKDD*.
- [112] Tan, B., Liu, B., Zheng, V., and Yang, Q. (2020). A federated recommender system for online services. In *Fourteenth ACM Conference on Recommender Systems*, pages 579–581.
- [113] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B*.
- [114] Trevisiol, M., Aiello, L. M., Schifanella, R., and Jaimes, A. (2014). Cold-start news recommendation with domain-dependent browse graph. In *Proceedings of ACM Conference on Recommender systems*, pages 81–88.
- [115] Voigt, P. and Von dem Bussche, A. (2017). The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*.
- [116] Wan, M., Ni, J., Misra, R., and McAuley, J. (2020). Addressing marketing bias in product recommendations. In *ACM WSDM*.

- [117] Wang, H., Zhang, F., Xie, X., and Guo, M. (2018a). Dkn: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1835–1844. International World Wide Web Conferences Steering Committee.
- [118] Wang, J., Tang, Q., Arriaga, A., and Ryan, P. (2019). Novel collaborative filtering recommender friendly to privacy protection. In *IJCAI*.
- [119] Wang, J. and Zhou, Z. (2020). Differentially private learning with small public data. In *AAAI*.
- [120] Wang, P., Guo, J., Lan, Y., Xu, J., Wan, S., and Cheng, X. (2015). Learning hierarchical representation model for nextbasket recommendation. In *ACM SIGIR*.
- [121] Wang, Y., Gu, Q., and Brown, D. (2018b). Differentially private hypothesis transfer learning. In *ECML-PKDD*.
- [122] Weinsberg, U., Bhagat, S., Ioannidis, S., and Taft, N. (2012). Blurme: Inferring and obfuscating user gender based on ratings. In *ACM RecSys*.
- [123] Wu, C., Wu, F., An, M., Huang, Y., and Xie, X. (2019a). Neural news recommendation with topic-aware news representation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 1154–1159.
- [124] Wu, C., Wu, F., An, M., Qi, T., Huang, J., Huang, Y., and Xie, X. (2019b). Neural news recommendation with heterogeneous user behavior. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.
- [125] Wu, F., Qiao, Y., Chen, J.-H., Wu, C., Qi, T., Lian, J., Liu, D., Xie, X., Gao, J., Wu, W., and Zhou, M. (2020). MIND: A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3597–3606. Association for Computational Linguistics.
- [126] Wu, S., Ren, W., Yu, C., Chen, G., Zhang, D., and Zhu, J. (2016a). Personal recommendation using deep recurrent neural networks in netease. In *IEEE ICDE*.
- [127] Wu, Y., DuBois, C., Zheng, A., and Ester, M. (2016b). Collaborative denoising auto-encoders for top-n recommender systems. In *ACM WSDM*.

- [128] Xiang, L., Yuan, Q., Zhao, S., Chen, L., Zhang, X., Yang, Q., and Sun, J. (2010). Temporal recommendation on graphs via long-and short-term preference fusion. In *ACM SIGKDD*.
- [129] Xiao, W., Zhao, H., Pan, H., Song, Y., Zheng, V. W., and Yang, Q. (2019). Beyond personalization: Social content recommendation for creator equality and consumer satisfaction. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.
- [130] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.
- [131] Yan, R., Lapata, M., and Li, X. (2012). Tweet recommendation with graph co-ranking. In *Proceedings of Annual Meeting of Association for Computational Linguistics*.
- [132] Yang, C., Bai, L., Zhang, C., Yuan, Q., and Han, J. (2017a). Bridging collaborative filtering and semi-supervised learning: A neural approach for poi recommendation. In *ACM SIGKDD*.
- [133] Yang, C., Yan, H., Yu, D., Li, Y., and Chiu, D. M. (2017b). Multi-site user behavior modeling and its application in video recommendation. In *ACM SIGIR*.
- [134] Yang, D., He, J., Qin, H., Xiao, Y., and Wang, W. (2015). A graph-based recommendation across heterogeneous domains. In *ACM CIKM*.
- [135] Yang, D., Qu, B., and Cudré, P. (2019a). Privacy-preserving social media data publishing for personalized ranking-based recommendation. *IEEE TKDE*.
- [136] Yang, Q., Chen, Y., Xue, G.-R., Dai, W., and Yu, Y. (2009). Heterogeneous transfer learning for image clustering via the social web. In *Proceedings of Annual Meeting of Association for Computational Linguistics*.
- [137] Yang, Q., Liu, Y., Chen, T., and Tong, Y. (2019b). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19.
- [138] Yang, Z., Salakhutdinov, R., and Cohen, W. (2017c). Transfer learning for sequence tagging with hierarchical recurrent networks.

- [139] Yao, Q., Guo, X., Kwok, T. Y., Tu, W., Chen, Y., Dai, W., and Yang, Q. (2019). Privacy-preserving stacking with application to cross-organizational diabetes prediction. In *IJCAI International Joint Conference on Artificial Intelligence*, page 4114.
- [140] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *NIPS*.
- [141] Yu, Y., Wan, X., and Zhou, X. (2016). User embedding for scholarly microblog recommendation. In *Proceedings of Annual Meeting of Association for Computational Linguistics*.
- [142] Yuan, F., Yao, L., and Benatallah, B. (2019). Darec: Deep domain adaptation for cross-domain recommendation via transferring rating patterns. In *IJCAI*.
- [143] Zhang, F., Yuan, N. J., Lian, D., Xie, X., and Ma, W.-Y. (2016). 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.
- [144] Zhang, Y. and Yang, Q. (2017). A survey on multi-task learning.
- [145] Zhao, L., Pan, S., Xiang, E., Zhong, E., Lu, Z., and Yang, Q. (2013). Active transfer learning for cross-system recommendation. In *AAAI*.
- [146] Zhao, X., Guo, Y., He, Y., Jiang, H., et al. (2014). We know what you want to buy: a demographic-based system for product recommendation on microblogs. In *ACM SIGKDD*.
- [147] Zheng, L., Noroozi, V., and Yu, P. (2017). Joint deep modeling of users and items using reviews for recommendation. In *ACM WSDM*.
- [148] Zhou, G., Zhu, X., Song, C., Fan, Y., Zhu, H., Ma, X., Yan, Y., Jin, J., Li, H., and Gai, K. (2018). Deep interest network for click-through rate prediction. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1059–1068.
- [149] Zhu, F., Wang, Y., Chen, C., Liu, G., Orgun, M., and Wu, J. (2018). A deep framework for cross-domain and cross-system recommendations. In *International Joint Conference on Artificial Intelligence*.